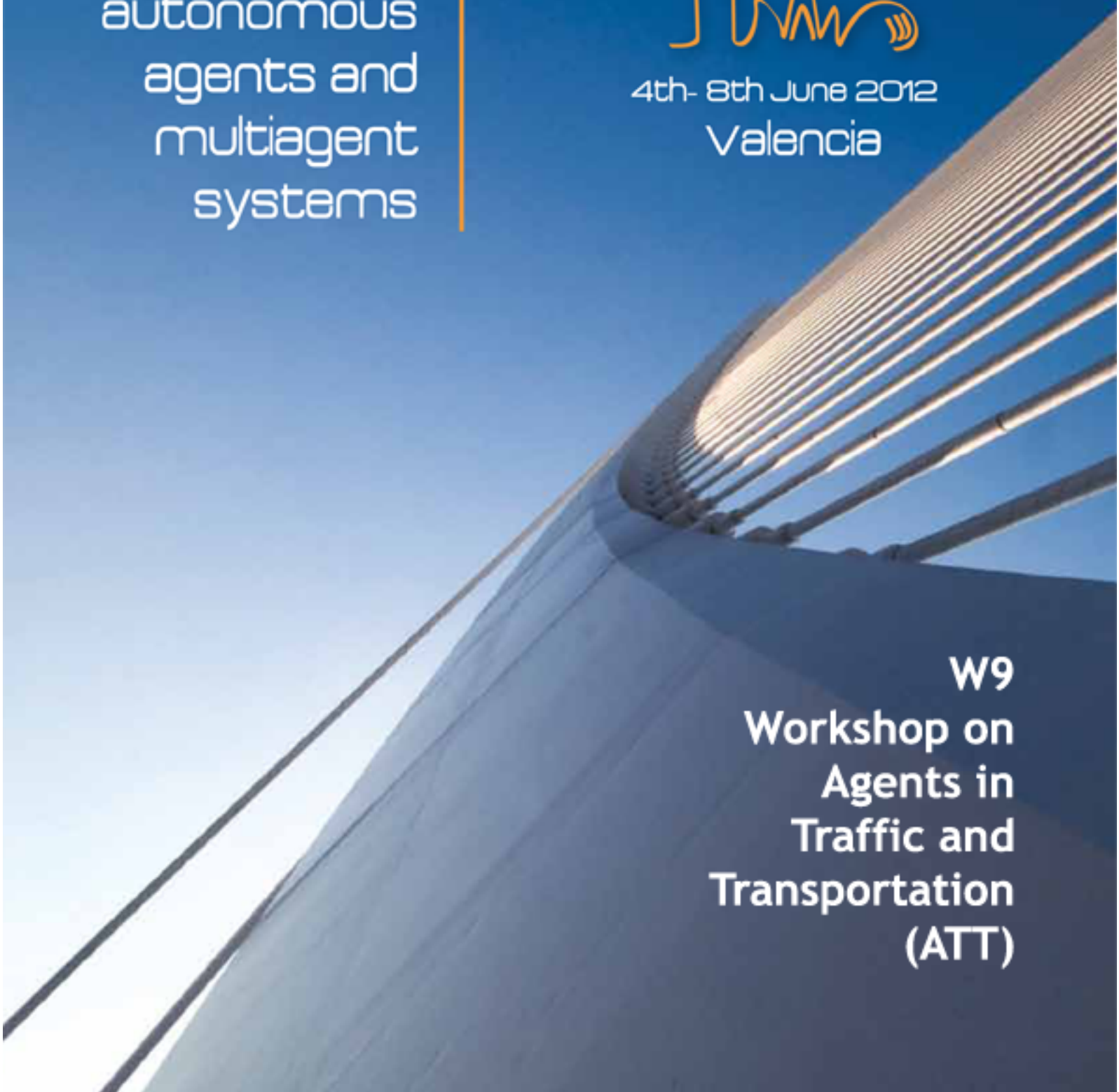
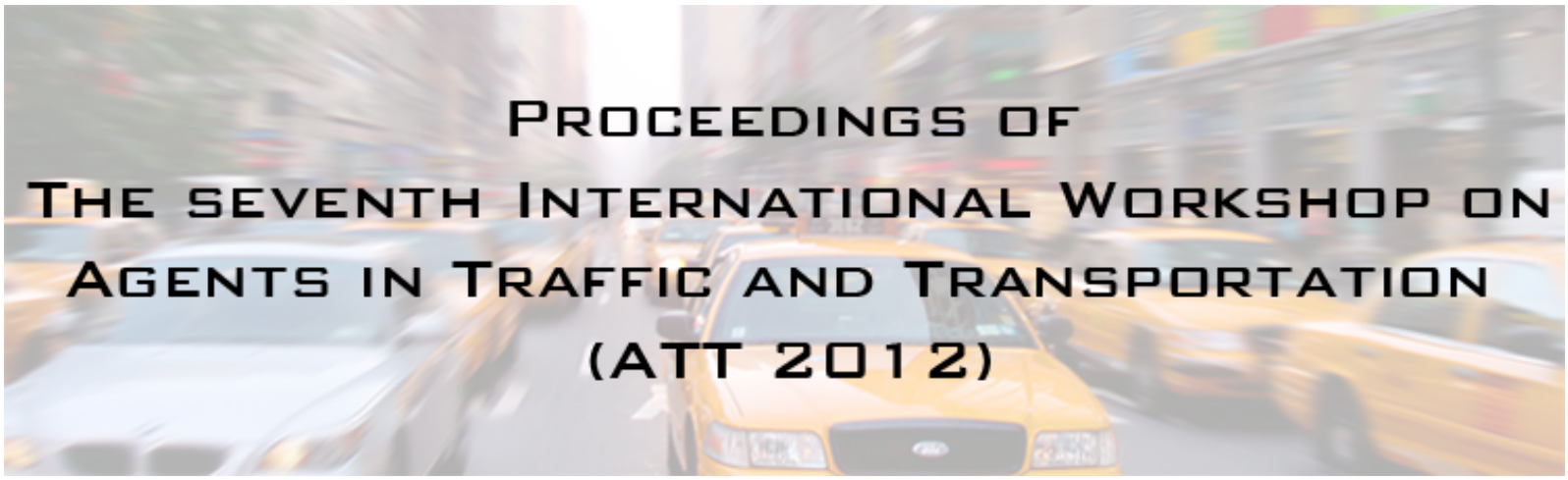
eleventh
international
conference
on
autonomous
agents and
multiagent
systems

AAMAS 2012

4th- 8th June 2012
Valencia



W9
Workshop on Agents in Traffic and Transportation (ATT)



held at AAMAS 2012
June 5th, Valencia, Spain

Matteo Vasirani
Franziska Klügl
Eduardo Camponogara
Hiromitsu Hattori
(editors)

Preface

Traffic and transportation became one of the most vivid application areas for multiagent and agent technology. Traffic and transportation systems are not only spatially distributed, but also made up by subsystems with a high degree of autonomy. Consequently, many applications in this domain can be adequately modelled as autonomous agents and multiagent systems.

This is the seventh of a well established series of workshops since 2000. The international workshop series on "Agents in Traffic and Transportation" (ATT) provides a forum for discussion for researchers and practitioners from the fields of artificial intelligence, multiagent systems and transportation engineering. The series aims at bringing researchers and practitioners together in order to set up visions on how agent technology can be used to model, simulate, and manage large-scale complex transportation systems, both at micro and at macro level.

This seventh edition of ATT was held together with the International Conference on Autonomous Agents and Multiagent Systems (AAMAS), in Valencia (Spain) on June 5, 2012. Previous editions were: Barcelona, together with Autonomous Agents in 2000; Sydney, together with ITS 2001; New York, together with AAMAS 2004; Hakodate, together with AAMAS 2006; Estoril, together with AAMAS 2008; Toronto, together with AAMAS 2010.

This edition of the workshop attracted the submission of 26 high-quality papers. All papers were thoroughly reviewed by at least three renowned experts in the field. Based on the reviewers' reports, and the unavoidable space and time constraints associated with the workshop, it was possible to select only 15 of these submissions as full papers, leading to an overall acceptance rate of 58%. In the process, a number of good and interesting papers had to be rejected.

The present workshop proceedings cover a broad range of topics related to Agents in Traffic and Transportation, tackling the use of tools and techniques based on multiagent simulation, learning, planning and data fusion, to name just a few. We hope you will enjoy it! Finally, we owe a big "Thank you" to all people – authors, reviewers, hosts and chairs of the AAMAS conference – who dedicated their time and energy to make this edition of ATT a success.

Valencia, June 2012 Matteo Vasirani, Franziska Klügl, Eduardo Camponogara and Hiromitsu Hattori.

Programme Committee

Adrian Agogino (NASA Ames Research Center, USA)
Sachiyo Arai (Chiba University, Japan)
Ana Bazzan (UFRGS, Brazil)
Itzhak Benenson (Tel-Aviv University, Israel)
Ladislau Bölöni (University of Central Florida, USA)
Vicent Botti (TU Valencia, Spain)
Brahim Chaib-draa (Université Laval, Canada)
Shih-Fen Cheng (Singapore Management University, Singapore)
Paul Davidsson (University of Malmö, Sweden)
Alexis Drogoul (IRD, Vietnam)
Juergen Dunkel (FHH Hannover, Germany)
The Duy Bui (Vietnam National University, Vietnam)
Alberto Fernández (University Rey Juan Carlos, Spain)
Hideki Fujii (University of Tokio, Japan)
Tom Holvoet (KU Leuven, Belgium)
Peter Jarvis (Xerox PARC, USA)
Ronald van Katwijk (TNO, Netherlands)
Ryszard Kowalczyk (Swinburne University, Australia)
Satoshi Kurihara (Osaka University, Japan)
Maite López-Sánchez (University of Barcelona, Spain)
Marin Lujak (University Rey Juan Carlos, Spain)
Rene Mandiau (Université de Valenciennes, France)
Jörg MÃijller (TU Clausthal, Germany)
Kai Nagel (TU Berlin, Germany)
Yuu Nakajima (Kyoto University, Japan)
Itsuki Noda (IRD, Japan)
Sascha Ossowski (University Rey Juan Carlos, Spain)
Omer Rana (Cardiff University, UK)
Rosaldo Rosetti (Universidade do Porto, Portugal)
Andreas Schadschneider (University of Cologne, Germany)
José Reynaldo A. Setti (University of São Paulo, Brazil)
Harry Timmermans (TU Eindhoven, Netherlands)
Sabine Timpf (University of Augsburg, Germany)
Kagan Tumer (Oregon State University, USA)
Giuseppe Vizzari (University of Milano-Bicocca, Italy)
Mathijs de Weerdt (TU Delft, Netherlands)
Li Weigang (University of Brasilia, Brazil)
Tomohisa Yamashita (AIST, Japan)

Additional Reviewers

Jomi Fred Hübner (Federal University of Santa Catarina, Brazil)
Christine Mumford (Cardiff University, UK)
Fredrik Ohlin (University of Malmö, Sweden)
Panagiotis Papadopoulos (Cardiff University, UK)
Federico Pecora (Örebro University, Sweden)
Scott Proper (Oregon State University, USA)
Stijn Vandael (KU Leuven, Belgium)
Logan Yliniemi (Oregon State University, USA)

Organizers

Matteo Vasirani (University Rey Juan Carlos, Spain)
Franziska Klügl (Örebro University, Sweden)
Eduardo Camponogara (Federal University of Santa Catarina, Brazil)
Hiromitsu Hattori (Kyoto University, Japan)

Schedule

9:00 – 10:20 Session 1: Coordination and Control

10:20 – 10:50 Coffee Break

10:50 – 13:00 Session 2: Planning and Routing

13:00 – 14:20 Lunch Break

14:20 – 16:00 Session 3: Pedestrians and Drivers

16:00 – 16:30 Coffee Break

16:30 – 17:45 Session 4: Data and Simulation

17:45 – 18:30 Closing Session

Urban Traffic Network Control by Distributed Satisficing Agents

Marcelo Lopes de Lima*
Department of Automation and Systems Engineering
Federal University of Santa Catarina
Florianópolis, SC 88040-900, Brazil
mllima@das.ufsc.br

Eduardo Camponogara†
Department of Automation and Systems Engineering
Federal University of Santa Catarina
Florianópolis, SC 88040-900, Brazil
camponog@das.ufsc.br

ABSTRACT

This work aims to present a distributed sliding-horizon control technique, applied to the control of an urban traffic network, where the control agents follow a satisficing approach coordinating themselves to obtain a solution that is satisfactory for all agents. The coordination mechanism finds, in a distributed way, the analytic center of the region where all agents are satisfied. We show that the analytic center is also Pareto optimal. Our approach is compared to the centralized one where, instead of a coordination mechanism, the solution is based on a fixed and ad hoc adjustment of the relative importance of the agents.

Categories and Subject Descriptors

G.1.6 [**Mathematics of Computing**]: Numerical Analysis—*Optimization*; I.2.11 [**Computing Methodologies**]: Artificial Intelligence—*Distributed Artificial Intelligence*; J.7 [**Computer Applications**]: Computers in Other Systems

General Terms

Performance, Design, Theory

Keywords

satisficing control, satisficing theory, multiagent control system, urban traffic control

1. INTRODUCTION

Traffic congestion, delays, and emissions of pollutants are recurring issues in dealing with urban traffic control and management [11]. Efforts to mitigate these problems are so diverse as the improvement and expansion of the existing traffic infrastructure, the implementation of control policies with priority for public transport, and the deployment of real-time traffic control systems. Examples of such control systems are PRODYN [6], OPAC [7], RHODES [10], and feedback control strategies based on the linear-quadratic regulator [4] and sliding-horizon control [3]. The feedback control strategies are inherently robust as has been observed in field studies [5]. Despite the increased performance of such control systems, the signaling control of traffic lights is carried out mostly using fixed-time control, in part due to its simplicity and mainly because of its low implementation and maintenance cost.

The present work aims to simplify the design, installation, and reconfiguration of sliding-horizon control techniques by carrying out sensing and control in a distributed manner, and also to improve the control performance using a *satisficing* multiagent system. In the satisficing approach, the agents try to attain a performance at least greater than their minimum specified level of performance but also, and very important, they coordinate themselves to obtain a solution that is satisfactory to all agents. The satisficing approach also allows negotiation between the agents when their minimum level of performance can not be simultaneously achieved. Although essentially different in its methods, our approach has a philosophical trace to Satisficing Theory [12] and Satisficing Control [8].

The contribution of this paper to the problem of traffic light control is twofold: one is the definition of a minimum level of performance and a negotiation policy that permit the agents to coordinate themselves, and the other is a mechanism for coordination. The coordination mechanism is to find, in a distributed way, the analytic center of the region where all agents are satisfied. We show that the analytic center is also Pareto optimal. Our approach is compared to the centralized one where, instead of a coordination mechanism, the solution is based on a fixed and ad hoc adjustment of the relative importance of the agents.

2. TRAFFIC DYNAMIC MODEL

In general, urban traffic networks are formed by junctions connected by road links where traffic lights may be used to coordinate the conflicting traffic flows. Among other possibilities [4], the traffic lights set the percentage of green time allocated to each link.

For this work, we will consider the network in Figure 1 with

*Supported by Petróleo Brasileiro S/A (Petrobras).

†Supported by CNPq/Brazil under grant #471405/2011-6.

eight junctions and one traffic light at each link approaching a junction. We can see that the network of Figure 1 can easily be represented by the directed graph of Figure 2, where the nodes are the junctions $m \in \mathcal{M}$ and the links $(i,j) \in \mathcal{E} \subset \mathcal{M} \times \mathcal{M}$ are the arcs connecting the nodes. The state variable $x_{i,j}$ represents the number of vehicles (queue) in the link from node j to the affected node i.

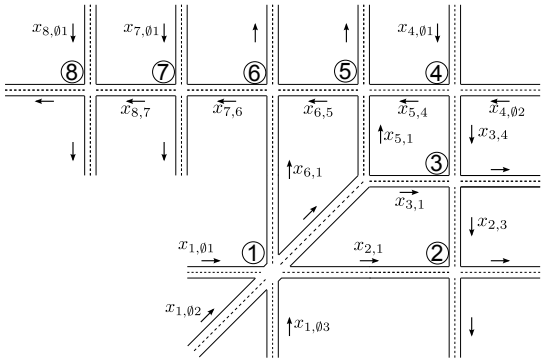


Figure 1: Example of a traffic network. The state variable $x_{i,j}$ is the number of vehicles (queue) of junction i affected by junction j.

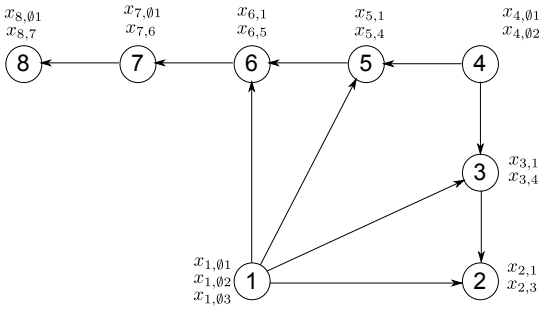


Figure 2: Graph for the traffic network example.

The interaction between nodes can be generalized assuming a generic node m and a set of input nodes $I(m) = \{i_1, ..., i_I\}$ and output nodes $O(m) = \{j_1, ..., j_O\}$ as in Figure 3. For example, node 3 has as input nodes $I(3) = \{1, 4\}$ and as output node $O(3) = \{2\}$. We divide the input nodes in internal and external nodes, for example $I(7) = I_{\mathrm{I}}(7) \cup I_{\mathrm{E}}(7)$ where $I_{\mathrm{I}}(7) = \{6\}$ is internal and $I_{\mathrm{E}}(7) = \{\emptyset 1\}$ is external.

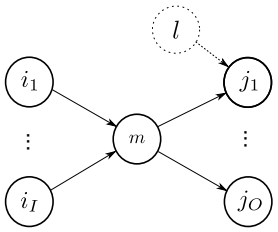


Figure 3: Illustration of input and output nodes of a node m.

The mathematical model chosen to describe the dynamics of the vehicle queues is based on the model known as store-and-forward [4] and is given by:

$$\mathbf{x}_m(k+1) = A_m \mathbf{x}_m(k) + \sum_{i \in I(m) \cup \{m\}} B_{m,i} \mathbf{u}_i(k) \qquad (1)$$

where the vector $\mathbf{x}_m(k) = (x_{m,i_1}(k), ..., x_{m,i_I}(k))$ are the queues of junction m influenced by the green time signals $\mathbf{u}_i(k) = (u_{i,i_1}(k), .., u_{i,i_I}(k))$ at instant k.

In this model, the matrix A_m is the identity, matrix $B_{m,m}$ expresses the discharge of queues $\mathbf{x}_m$ as a function of green times $\mathbf{u}_m$, and matrices $B_{m,i}$, $i \in I(m)$, represent how queues $\mathbf{x}_m$ build up as queues $\mathbf{x}_i$ are emptied by $\mathbf{u}_i$ green times. Matrices $B_{m,i}$, $i \in I(m) \cup \{m\}$, are functions of the physical characteristics of the traffic network. For the example, consider node 3 for which its matrices are:

$$B_{3,3} = T_3 \begin{bmatrix} -\frac{S_{3,1}}{C_3} & 0 \\ 0 & -\frac{S_{3,4}}{C_3} \end{bmatrix}$$

$$B_{3,1} = T_3 \begin{bmatrix} \rho_{3,1,\emptyset 1} \cdot \frac{S_{1,\emptyset 1}}{C_1} & \rho_{3,1,\emptyset 2} \cdot \frac{S_{1,\emptyset 2}}{C_1} & \rho_{3,1,\emptyset 3} \cdot \frac{S_{1,\emptyset 3}}{C_1} \\ 0 & 0 & 0 \end{bmatrix}$$

$$B_{3,4} = T_3 \begin{bmatrix} 0 & 0 \\ \rho_{3,4,\emptyset 1} \cdot \frac{S_{4,\emptyset 1}}{C_4} & \rho_{3,4,\emptyset 2} \cdot \frac{S_{4,\emptyset 2}}{C_4} \end{bmatrix}$$

where T_3 is the sample time (in seconds), $S_{i,j}$ is the saturation flow on the link from j to i (in vehicles per second), $\rho_{m,i,j}$ is the rate at which vehicles from link j to i enter link i to m, and C_i (in seconds) is the cycle time of junction i as explained below. Notice that the entries in $B_{3,3}$ are negative, indicating queue discharge as a function of green time signals $\mathbf{u}_3$.

The concept of cycle time is illustrated in Figure 4. Each cycle is composed by stages meaning a particular traffic light configuration. In the example of Figure 4, after stage 3, stage 1 repeats starting another cycle. From one stage to another there is a lost time added to avoid interference between stages. The sum of all green times plus lost times in a junction gives the cycle time for that junction.

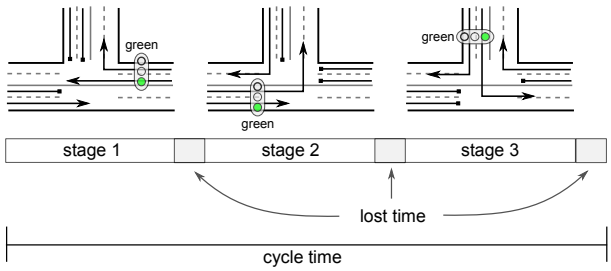


Figure 4: Illustration of the cycle time.

Three constraints are imposed to the junctions:

Constraint 1: The sum of the green times $u_{m,i}$ and lost time $L_{m,i}$ must be equal to the cycle time C_m of the junction m to which they belong,

$$\sum_{i \in I(m)} u_{m,i} + L_{m,i} = C_m, \quad \forall m \in \mathcal{M}$$

Constraint 2: The green times can not be negative,

$$\mathbf{u}_m \geq 0, \quad \forall m \in \mathcal{M}$$

Constraint 3: The states are always nonnegative,

$$\mathbf{x}_m \geq 0, \quad \forall m \in \mathcal{M}$$

3. DISTRIBUTED SATISFICING CONTROL

We propose a distributed approach to control the green time of the traffic lights, where a satisfactory global solution for the entire network is obtained from the specification of the agents. A convenient arrangement is to allocate one agent to each subsystem. In other words, for each node $m \in \mathcal{M} = \{1, \ldots, M\}$, the set of all nodes, we associate an agent $\mathcal{A}_m$ belonging to the agent set $\mathcal{A} = \{\mathcal{A}_1, \ldots, \mathcal{A}_M\}$.

Our agents apply a sliding-horizon control scheme where agent $\mathcal{A}_m$, $m \in \mathcal{M}$, calculates a plan of actions for N_m^{u} periods ahead of the current time so that the evolution in N_m^{p} periods of its states is satisfactory given a criterion. N_m^{u} and N_m^{p} are called control and prediction horizons respectively. Given the initial state $\mathbf{x}_m(0)$ of subsystem m, the predicted states are given by the following equation:

$$\begin{cases} \widetilde{\mathbf{x}}_m = \widetilde{A}_m \mathbf{x}_m(0) + \sum\limits_{i \in I(m) \cup \{m\}} \widetilde{B}_{m,i} \widetilde{\mathbf{u}}_i \\ \widetilde{\mathbf{y}}_m = \widetilde{C}_m \widetilde{\mathbf{x}}_m \\ \mathbf{y}_m(0) = C_m \mathbf{x}_m(0) \end{cases}$$

with

$$\widetilde{\mathbf{x}}_m = \begin{bmatrix} \mathbf{x}_m(1) \\ \mathbf{x}_m(2) \\ \vdots \\ \mathbf{x}_m(N_m^{\mathrm{p}}) \end{bmatrix}, \widetilde{\mathbf{y}}_m = \begin{bmatrix} \mathbf{y}_m(1) \\ \mathbf{y}_m(2) \\ \vdots \\ \mathbf{y}_m(N_m^{\mathrm{p}}) \end{bmatrix}, \widetilde{\mathbf{u}}_i = \begin{bmatrix} \mathbf{u}_i(0) \\ \mathbf{u}_i(1) \\ \vdots \\ \mathbf{u}_i(N_m^{\mathrm{u}} - 1) \end{bmatrix},$$

$$\widetilde{A}_m = \begin{bmatrix} A_m \\ (A_m)^2 \\ \vdots \\ (A_m)^{N_m^{\mathrm{p}}} \end{bmatrix}, \widetilde{C}_m = \begin{bmatrix} C_m & & \\ & \ddots & \\ & & C_m \end{bmatrix},$$

$$\widetilde{B}_{m,i} = \begin{bmatrix} I & \mathbf{0} & \cdots & \mathbf{0} \\ A_m & I & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ (A_m)^{N_m^{\mathrm{p}}-1} & (A_m)^{N_m^{\mathrm{p}}-2} & \cdots & (A_m)^{N_m^{\mathrm{p}}-N_m^{\mathrm{u}}} \end{bmatrix} B_{m,i}$$

where vector $\widetilde{\mathbf{x}}_m$ is the predicted states N_m^{p} periods ahead, and vector $\widetilde{\mathbf{u}}_m = (\mathbf{u}_m(0), \mathbf{u}_m(1), \ldots, \mathbf{u}_m(N_m^{\mathrm{u}} - 1))$ constitutes the *action plan* of agent $\mathcal{A}_m$, N_m^{u} periods ahead.

The prediction model can be restated in a compact form as:

$$\widetilde{\mathbf{x}}_m = \widetilde{A}_m \mathbf{x}_m(0) + \widetilde{B}_m \widetilde{\mathbf{v}}_m \tag{2}$$

where $\widetilde{B}_m = [\widetilde{B}_{m,m} \quad \widetilde{B}_{m,i_1} \quad \cdots \quad \widetilde{B}_{m,i_I}]$ and the vector $\widetilde{\mathbf{v}}_m = (\widetilde{\mathbf{u}}_m, \widetilde{\mathbf{u}}_{i_1}, \cdots, \widetilde{\mathbf{u}}_{i_I}) = (\widetilde{\mathbf{u}}_m, \widetilde{\mathbf{u}}_{I(m)})$ is called the *plan profile* of agent $\mathcal{A}_m$, N_m^{u} periods ahead.

3.1 Agent Behavior

Our goal through the following subsections is to model an appropriate behavior for the agents so as to accomplish their objectives and restrictions and to guarantee, in a distributed way, individual and global specified criteria of performance.

The satisficing theory [13, 12] proposes that the objectives of the agents should be evaluated using two indexes, one related to the goals (which therefore should be maximized) and the other related to the cost of energy or resources (which therefore should be minimized). These indexes will serve as a basis for establishing the *behavior* of the agents.

DEFINITION 1. Selectability *(f_S) is the index of the usefulness of the actions with respect to the objective.*

DEFINITION 2. Rejectability *(f_R) is the index of the cost associated with the actions.*

While the selectability function is normally concave since the usefulness of the actions should be maximized, the rejectability function is typically convex because the cost of the actions should be minimized.

Selectability and rejectability are the building blocks for the utilities of the types of agents that will be considered in the sequel, namely *selfish* and *satisficing* agents.

3.2 Selfish Agents

Defining the symmetric matrices $\widetilde{Q}_m \succeq 0$ (positive semi-definite) and $\widetilde{R}_m \succ 0$ (positive definite) as:

$$\widetilde{Q}_m = \begin{bmatrix} Q_m & & \\ & \ddots & \\ & & Q_m \end{bmatrix}, \ \widetilde{R}_m = \begin{bmatrix} R_m & & \\ & \ddots & \\ & & R_m \end{bmatrix}$$

of appropriated dimensions, we define the selectability and the rejectability of agent $\mathcal{A}_m$ by the following functions:

$$f_{S,m} = -\widetilde{\mathbf{x}}_m' \widetilde{Q}_m \widetilde{\mathbf{x}}_m \tag{3}$$

$$f_{R,m} = \widetilde{\mathbf{u}}_m' \widetilde{R}_m \widetilde{\mathbf{u}}_m \tag{4}$$

The selfish utility of agent $\mathcal{A}_m$ is given by:

$$f_m(\widetilde{\mathbf{v}}_m, \mathbf{x}_m(0)) = f_{S,m} - \alpha_m f_{R,m}, \ \alpha_m > 0$$

that depends on its plan profile and on its initial state (initial queues). From now on we will omit the dependence of the utility on the initial state.

The maximization of the selfish utility expresses the desire of the agent to maintain the queues and the green times close to zero, the stable equilibrium.

3.3 Satisficing Agents

In a cost versus benefit analysis, any action that results in an acceptable selectability compared to the rejectability is a defensible choice that belongs to the satisficing set.

DEFINITION 3. Satisficing Set*: the region S of the domain where the difference between selectability and an adjusted rejectability results more than a minimum acceptable level of utility, formally*

$$S = \{\widetilde{\mathbf{v}} | f(\widetilde{\mathbf{v}}) = f_S(\widetilde{\mathbf{v}}) - \alpha \cdot f_R(\widetilde{\mathbf{v}}) \geq \beta\}$$

with $\alpha \in [0, \infty)$ denoting the sensitivity to cost with respect to the benefit and $\beta \in \mathbb{R}$ being the minimum acceptable level of utility, or satisfaction. *Any solution that belongs to the satisficing set is a* satisficing solution.

The objective of the satisficing agent $\mathcal{A}_m$ is to find a satisficing solution, where its selfish utility f_m results greater than a minimum level of satisfaction β_m. For the urban traffic application, we define

$$\beta_m = N_m^{\mathrm{p}}(-\mathbf{x}_m^{\mathrm{s}'} Q_m \mathbf{x}_m^{\mathrm{s}}) \tag{5}$$

where $\mathbf{x}_m^{\mathrm{s}}$ are the maximal, but still satisfactory, average queues of junction m. This definition of β_m expresses the desire of the agent to maintain the average queues less than the satisfactory maximal queue.

3.4 Coordination of the Satisficing Agents

In a multiagent system, the agents should coordinate themselves to reach a satisfactory collective solution.

The classical way to define a global utility H for a set of M agents is through a scalarization approach [1] according to which the interests of selfish agents are aggregated as a function

$$H(\widetilde{\mathbf{v}}) = \sum_{m=1}^{M} w_m f_m(\widetilde{\mathbf{v}}_m), \qquad w_m > 0, \forall m \in \mathcal{M}$$

of the vector $\widetilde{\mathbf{v}} = (\widetilde{\mathbf{u}}_1, \ldots, \widetilde{\mathbf{u}}_M)$. The decision process is centralized and defined by the following problem over $\widetilde{\mathbf{v}}$:

$$P^{\mathrm{C}} : \begin{cases} \text{Maximize} & H(\widetilde{\mathbf{v}}) = \sum\limits_{m=1}^{M} w_m f_m(\widetilde{\mathbf{v}}_m) \\ \text{subject to} & \widetilde{\mathbf{v}} \in \widetilde{\mathcal{D}} \end{cases} \tag{6}$$

where $\widetilde{\mathcal{D}}$ is a generic convex domain. One characteristic of this problem is that any optimal solution $\widetilde{\mathbf{v}}^{\star}$ to problem P^{C} is Pareto optimal. Another characteristic is that the adjustment of w_m defines a particular solution in the Pareto set.

In a distributed system, on the other hand, a solution is reached by the interactions of the agents. Selfish agents produce a Nash point that is normally not Pareto. Instead, we will apply the satisficing agents which solve the following satisficing problem:

$$P_m^{\mathrm{S}} : \begin{cases} \text{find} & \widetilde{\mathbf{u}}_m \in \widetilde{\mathcal{D}}_m \\ \text{such that:} & f_m(\widetilde{\mathbf{v}}_m) \geq \beta_m \\ & f_j(\widetilde{\mathbf{v}}_j) \geq \beta_j, \forall j \in O(m) \end{cases} \tag{7}$$

where each agent $\mathcal{A}_m$ tries to find a satisficing solution for itself and for the affected agents.

The interactions of the satisficing agents will produce a solution in the jointly satisficing set,

$$S \triangleq \{\widetilde{\mathbf{v}} : f_m(\widetilde{\mathbf{v}}_m) \geq \beta_m, \forall m \in \mathcal{M}\}$$

Notice that in this case we have not just one solution but a set of possible solutions.

THEOREM 1. *The analytic center $\widetilde{\mathbf{v}}^{\dagger} = (\widetilde{\mathbf{u}}_1^{\dagger}, \cdots, \widetilde{\mathbf{u}}_M^{\dagger})$ of the satisficing set $S = \{\widetilde{\mathbf{v}} : f_m(\widetilde{\mathbf{v}}_m) \geq \beta_m, \forall m \in \mathcal{M}\}$ is Pareto optimal and equivalent to the centralized solution with $w_m = 1/(f_m(\widetilde{\mathbf{v}}_m^{\dagger}) - \beta_m)$.*

We call coordination the process by which the agents, in a distributed manner, try to find a jointly satisficing solution and, in particular, the analytic center of the satisficing set.

The constrained analytic center of the satisficing set is obtained in two phases. In phase I a feasible solution $\widetilde{\mathbf{v}}^{\mathrm{s}}$ for which $f_m(\widetilde{\mathbf{v}}^{\mathrm{s}}) \geq \beta$ for all $m \in \mathcal{M}$ is calculated and used in phase II for the computation of the analytic center $\widetilde{\mathbf{v}}^{\dagger}$ of the satisficing set.

The phase I problem of agent $\mathcal{A}_m$ is of the form:

$$\begin{aligned} F_m^{\mathrm{I}}(\widetilde{\mathbf{u}}_{\neg m}) : \min_{\widetilde{\mathbf{u}}_m} F_m^{\mathrm{I}}(\widehat{\mathbf{u}}_m | \widetilde{\mathbf{u}}_{\neg m}) = & \sum_{j \in \widetilde{O}(m)} s_j \\ \text{s.t. : } & f_m(\widetilde{\mathbf{u}}_m | \widetilde{\mathbf{u}}_{\neg m}) \geq \beta_m - s_m \\ & f_j(\widetilde{\mathbf{u}}_m | \widetilde{\mathbf{u}}_{\neg m}) \geq \beta_j - s_j, \ \forall j \in O(m) \\ & s_j \geq 0, \ \forall j \in \tilde{O}(m) \\ & \widetilde{\mathbf{u}}_m \in \widetilde{\mathcal{D}}_m \end{aligned} \tag{8}$$

where $\widehat{\mathbf{u}}_m = (\widetilde{\mathbf{u}}_m, \widetilde{\mathbf{s}}_m)$, $\widetilde{\mathbf{s}}_m = (s_j : \forall j \in \widetilde{O}(m))$, $\widetilde{O}(m) = O(m) \cup \{m\}$ and $\widetilde{\mathbf{u}}_{\neg m}$ denotes the action plan of the agents but agent $\mathcal{A}_m$. Let $\widehat{\mathbf{v}}^{\mathrm{I}} = (\widehat{\mathbf{u}}_1^{\mathrm{I}}, \ldots, \widehat{\mathbf{u}}_M^{\mathrm{I}})$ be an optimal solution to the set of problems $\{F_m^{\mathrm{I}}\}_{m \in \mathcal{M}}$. If $F_m^{\mathrm{I}}(\widehat{\mathbf{v}}^{\mathrm{I}}) = 0$ for all $m \in \mathcal{M}$, then the satisficing set is nonempty. If $F_m^{\mathrm{I}}(\widehat{\mathbf{v}}^{\mathrm{I}}) > 0$ for any m, then there does not exist a simultaneously satisficing solution for all the agents, in which case $\{m \in \mathcal{M} : s_m > 0\}$ is the subset of agents that cannot be satisfied.

Phase II solves, starting from the solution found in phase I, the set $\{F_m\}_{m \in \mathcal{M}}$ of problems given by:

$$\begin{aligned} F_m(\widetilde{\mathbf{u}}_{\neg m}) : \ \min_{\widetilde{\mathbf{u}}_m} F_m(\widetilde{\mathbf{u}}_m | \widetilde{\mathbf{u}}_{\neg m}) = & \\ - \sum_{j \in \widetilde{O}(m)} & \log(f_j(\widetilde{\mathbf{u}}_j | \widetilde{\mathbf{u}}_{\neg j}) - \beta_j) \\ \text{subject to } \ \widetilde{\mathbf{u}}_m & \in \widetilde{\mathcal{D}}_m \end{aligned} \tag{9}$$

where the functions $F_m(\widetilde{\mathbf{u}}_m | \widetilde{\mathbf{u}}_{\neg m})$, called log barrier functions [1], force the solution to the analytic center of the satisficing set.

Phase I and II problems can be solved by the agent set $\mathcal{A}$ using a distributed interior-point method. The convergence analysis presented in [2] ensures that the iterative solution of the set of problems in phase I and II converges to the analytic center when only nonneighboring agents iterate in parallel.

In every cycle, the coordination of the agents is achieved by the calculation of the analytic center of the problem set $\{P_m^{\mathrm{S}}\}_{m \in \mathcal{M}}$. According to Theorem 1, the analytic center results in a Pareto solution. This Pareto solution is equivalent to that obtained by a centralized solution *if w_m where not fixed*. However, the centralized approach uses fixed weights that are adjusted in an ad hoc manner.

3.5 Negotiation

From the definition of the satisficing set it can be seen that a smaller sensitivity α and/or a smaller satisfaction β lead to bigger satisficing sets. So, when the agents can not be simultaneously satisfied, they have to negotiate adjusting their sensibility to cost or adjusting their minimum level of satisfaction.

4. SIMULATION RESULTS

This section presents the application of distributed satisficing agents to the control of the 8-junction urban traffic network shown in Figure 1. The experimental analysis aims to assess the performance of the distributed satisficing control approach by comparing it to a centralized approach. The satisficing agents solve the set $\{P_m^{\mathrm{S}}\}_{m\in\mathcal{M}}$ of analytic center problems and a centralized agent solves P^{C}, while respecting the constraints described in Section 2.

The centralized problem and the analytic center of the satisficing problems were simulated in `Matlab` and solved using `CVX` [9], a package for specifying and solving convex programs.

4.1 Experimental Setup

There are only four parameters that must be defined by the user, with the advantage that all of them have physical meaning. They are:

P.1) The satisfactory maximal average queues $\mathbf{x}_m^{\mathrm{s}}$: an average queue that still is acceptable at each link of junction m.

P.2) The capacity of the links: the maximum number of vehicles that the link can support.

P.3) The prediction horizon N_m^{p}.

P.4) The cycle time C_m.

For this simulation, the satisfactory maximal average queue was defined as two times the maximal discharge obtained by the nominal green time set as $\mathbf{u}_1^{\mathrm{nom}} = (40, 40, 40)$ and $\mathbf{u}_{2..8}^{\mathrm{nom}} = (60, 60)$, that is, $\mathbf{x}_m^{\mathrm{s}} = -2B_{m,m}\mathbf{u}_m^{\mathrm{nom}}$ for all $m \in \mathcal{M}$. The capacity of the links were set as three times the satisfactory queue, but in a real application the capacity is easily assessed based on the dimensions of the link. The prediction horizon was chosen equal to 5 minutes (300 seconds) and the cycle time equal to 2 minutes (120 seconds), for all agents.

The remaining parameters were set according to the following rules:

R.1) $T_m = C_m$: the sample time T_m was made equal to the cycle time for all agents.

R.2) $N_m^{\mathrm{u}} = N_m^{\mathrm{p}}$: the horizons were made equal in all junctions.

R.3) Matrix $R_m = 0$: in our simulation traffic signaling does not incur any cost. Green times should not be penalized.

R.4) Matrix $Q_m = \mathrm{diag}(1/\mathrm{cap}_{m,i} : i \in I(m))$: the matrix Q_m was set diagonal with its elements equal to the inverse of the capacity of each link that approaches junction m, as proposed in [4].

R.5) $\beta_m = N_m^{\mathrm{p}}(-\mathbf{x}_m^{\mathrm{s0}\prime} Q_m \mathbf{x}_m^{\mathrm{s0}})$, where

$$x_{m,i}^{\mathrm{s0}} = \max(x_{m,i}^{\mathrm{s}}, x_{m,i}(0)) \quad \forall i \in I_{\mathrm{E}}(m)$$
$$x_{m,i}^{\mathrm{s0}} = x_{m,i}^{\mathrm{s}} \quad \forall i \in I_{\mathrm{I}}(m)$$

With the original definition of β_m (see Equation 5), an excessive number of vehicles coming from outside may become impossible for the system to maintain all the queues less that the satisfactory maximal average. In rule R.5, the original definition of β_m is modified to incorporate the negotiation policy chosen for the network. For this experiment, the negotiation policy is to maintain the specification of the internal nodes and degrade only the nodes receiving vehicles from outside the system allowing them to accumulate more vehicles. Observe that because nodes $2, 3, 5$ and 6 have only internal input nodes, their minimum level of satisfaction is not modified by rule R.5. On the other hand, nodes $1, 4, 7$ and 8 tolerate external queue sizes greater than the satisfactory, tolerating at least their actual number of vehicles, whatever it is.

The network physical parameters, saturation flows $S_{i,j}$ in vehicles per minute and conversion rates $\rho_{m,i,j}$ in percentage, are in the appendix.

4.2 Experimental Analysis

The analysis of the satisficing agents was made against a centralized one in which weights were set $w_m = 1$ for all $m \in \mathcal{M}$. Notice that the tuning of the centralized agent is based on an ad hoc definition of weights. We are considering a constant arrival of vehicles in node 1 and 4 equal to $(5, 15, 10)$ and $(10, 10)$ respectively, and initial queues $\mathbf{x}_1(0)$ through $\mathbf{x}_8(0)$ as $(10, 50, 20)$, $(60, 20)$, $(5, 35)$, $(120, 30)$, $(10, 20)$, $(9, 20)$, $(15, 20)$ and $(17, 9)$.

Figures 5 to 10 show the evolution in 10 cycles (20 minutes) of the vehicle queues, the calculated green times, and the utility of the satisficing and centralized agents, respectively. The bars show the vector components stacked from below. For example, in Figure 5, the components of the vector $\mathbf{x}_1 = (x_{1,\emptyset 1}, x_{1,\emptyset 2}, x_{1,\emptyset 3})$ are in black, gray and white respectively.

We can see in Figures 5 and 6 that the satisficing agents $\mathcal{A}_1$ and $\mathcal{A}_4$ accumulate more queues than the centralized agent although agent $\mathcal{A}_4$ starts to decrease its queues after a moment of increase. Something impedes $\mathcal{A}_1$ to maximize its time of green and also forces $\mathcal{A}_4$ to postpone in 1 cycle any relevant green time.

The reason is that agents $\mathcal{A}_1$ and $\mathcal{A}_4$ have a compromise with the satisfaction of the internal agents due to the solution of the satisficing problems and due to the negotiation scheme induced by rule R.5. Due to the negotiation rule, we can see in Figures 5 and 6 that agents $\mathcal{A}_1$ and $\mathcal{A}_4$ adjust their minimum level of satisfaction (dash-dot line) to permit lower utilities (solid lines) and to accommodate more queues. Remember that the policy we chose was to maintain the specification of the internal nodes and degrade only

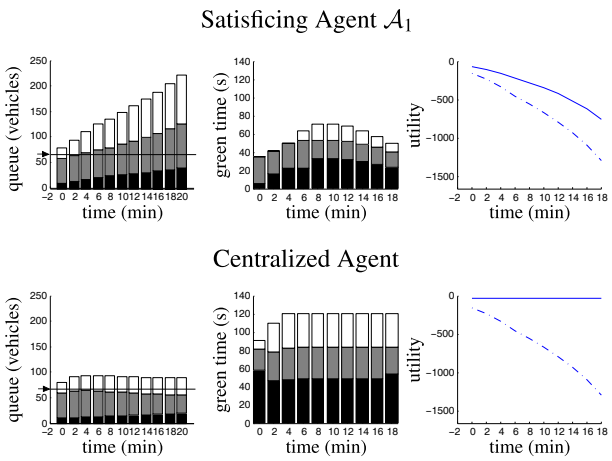


Figure 5: Control in junction 1.

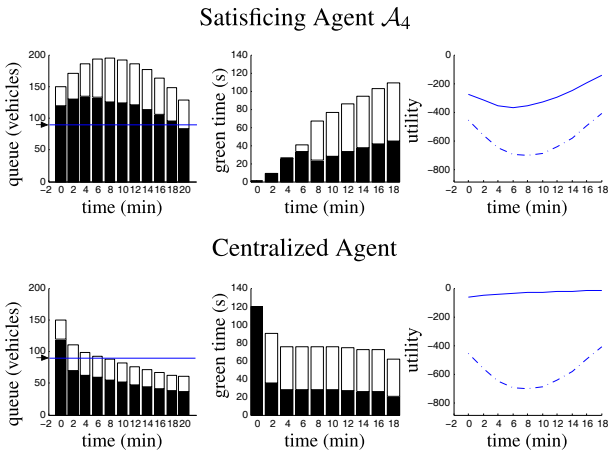


Figure 6: Control in junction 4.

the nodes receiving vehicles from outside the system. This compromise is difficult to obtain in the centralized control due to its ad hoc nature.

In Figures 7 and 8 we see that the discharge made by the centralized agent overcharges node 5 and node 6 making it unable to maintain the sum of the corresponding queues below the maximal satisfactory (horizontal line indicated by an arrow) even with the maximal of green (120 seconds). In the centralized case, the utility of nodes 5 and 6 are very below the minimum level specified for agents $\mathcal{A}_5$ and $\mathcal{A}_6$ (horizontal dash-dot line) indicating their dissatisfaction. We chose to show only the internal junctions 5 and 6 because they suffer the highest effect.

The satisficing agents also seem to present a better behavior in the presence of model error. Figures 9 and 10 show the behavior of junction 5 and 6 when the rates of flow coming from junction 4 are greater than what is expected by the nominal model. In this case, instead of 30%, the flow from junction 4 to junction 5 is 70% of the junction total flow. We see that the satisficing agents maintain the queues in a satisfactory level while the centralized agent builds up the queues even more.

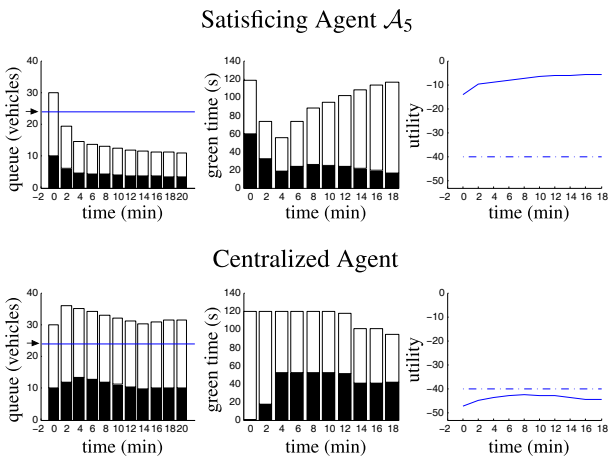


Figure 7: Control in junction 5.

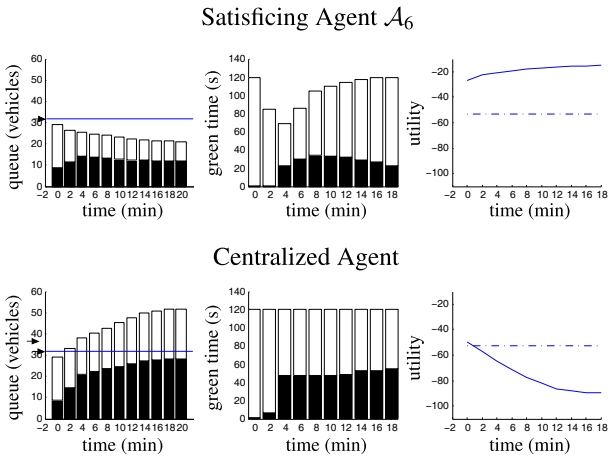


Figure 8: Control in junction 6.

5. CONCLUDING REMARKS

The satisficing approach offers a mechanism of coordination that, applied to an urban traffic network, has the following advantages if compared to a centralized classical approach:

- the adjustment of the agents are based on physical parameters instead of the ad hoc adjustment of weights;
- the definition of the minimum level of satisfaction gives meaning to the control objectives;
- the negotiation policy offers a mechanism to alleviate the control objectives in case of infeasibility and is flexible enough to accommodate other strategies. For example, instead of penalizing only the junction that receives vehicles from outside the system, one can define a negotiation policy where all the agents reduce their minimum level of satisfaction;
- and it seems to be more robust to model error.

It is also worth to mention that any satisficing solution, not only the analytic center, is good enough to make the agents satisfied. This fact can be used to simplify the distributed algorithm and reduce decision time.

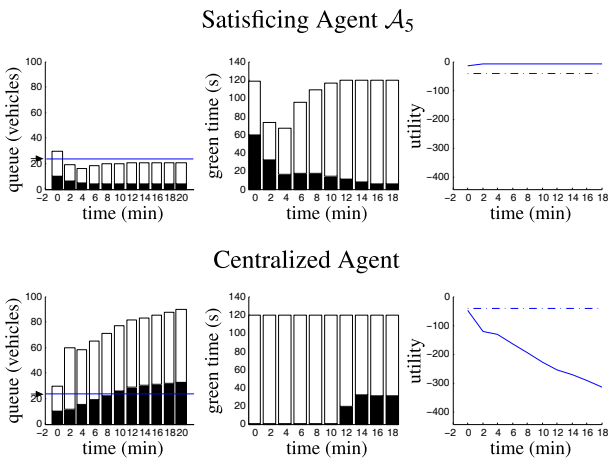


Figure 9: Junction 5 under model error.

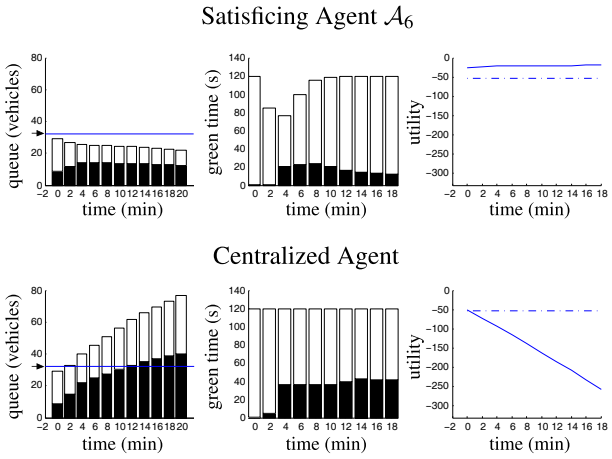


Figure 10: Junction 6 under model error.

6. REFERENCES

[1] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[2] E. Camponogara and H. F. Scherer. Distributed optimization for model predictive control of linear dynamic networks with control-input and output constraints. *IEEE Transactions on Automation Science and Engineering*, 8(1):233–242, 2011.

[3] L. B. de Oliveira and E. Camponogara. Predictive control for urban traffic networks: initial evaluation. In *Proceedings of the 3rd IFAC Symposium on System, Structure and Control*, Iguassu Falls, Brazil, Oct. 2007.

[4] C. Diakaki, M. Papageorgiou, and K. Aboudolas. A multivariable regulator approach to traffic-responsive network-wide signal control. *Control Engineering Practice*, 10:183–195, 2002.

[5] V. Dinopoulou, C. Diakaki, and M. Papageorgiou. Applications of the urban traffic control strategy TUC. *European Journal of Control*, 175(3):1652–1665, 2006.

[6] J. L. Farges, J. J. Henry, and J. Tufal. The PRODYN real-time traffic algorithm. In *Proceedings of the Fourth IFAC Symposium on Transportation Systems*, pages 307–312, Baden-Baden, Germany, 1983.

[7] N. H. Gartner. OPAC: a demand-responsive strategy for traffic signal control. *Transportation Research Record*, 906:75–84, 1983.

[8] M. A. Goodrich, W. C. Stirling, and R. L. Frost. A theory of satisficing decisions and control. *IEEE Transactions on Systems, Man and Cybernetics, Part A*, 28(6):763–779, 1998.

[9] M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, version 1.21, Apr. 2011.

[10] P. Mirchandani and L. Head. A real-time traffic signal control system: architecture, algorithms, and analysis. *Transportation Research Part C: Emerging Technologies*, 9(6):415–432, 2001.

[11] M. Papageorgiou. Overview of road traffic control strategies. In *Information and Communication Technologies: From Theory to Applications*, pages LIX–LLX, 2004.

[12] W. C. Stirling. *Satisficing Games and Decision Making: with Application to Engineering and Computer Science*. Cambridge University Press, 2003.

[13] W. C. Stirling and R. L. Frost. Reconciling group and individual preferences of multi-agent systems. *IEEE International Conference on Networking, Sensing and Control*, pages 477–482, 2007.

APPENDIX

A. PROOF OF THEOREM

THEOREM 1. *The analytic center* $\widetilde{\mathbf{v}}^\dagger = (\widetilde{\mathbf{u}}_1^\dagger, \cdots, \widetilde{\mathbf{u}}_M^\dagger)$ *of the satisficing set* $S = \{\widetilde{\mathbf{v}} : f_m(\widetilde{\mathbf{v}}_m) \geq \beta_m, \forall m \in \mathcal{M}\}$ *is Pareto optimal and equivalent to the centralized solution with* $w_m = 1/(f_m(\widetilde{\mathbf{v}}_m^\dagger) - \beta_m)$.

PROOF. (1) An optimal solution $\widetilde{\mathbf{v}}^\star = (\widetilde{\mathbf{u}}_1^\star, \cdots, \widetilde{\mathbf{u}}_M^\star)$ to the unconstrained centralized problem $P^{\mathrm{C}} : \max_{\widetilde{\mathbf{v}}} H(\widetilde{\mathbf{v}})$, where $H(\widetilde{\mathbf{v}}) \triangleq \sum_{m=1}^M w_m f_m(\widetilde{\mathbf{v}}_m)$ and $w_m > 0$, is Pareto optimal. An optimal solution is obtained when $\nabla_{\widetilde{\mathbf{v}}} H(\widetilde{\mathbf{v}}^\star) = \mathbf{0}$, that is, when $\sum_{m=1}^M w_m \nabla f_m(\mathbf{v}_m^\star) = \mathbf{0}$, because H is concave. (2) On the other hand, the analytic center of the satisficing set S is obtained by solving the problem $\min_{\widetilde{\mathbf{v}}}\{F(\widetilde{\mathbf{v}}) = \sum_{m=1}^M -\log(f_m(\widetilde{\mathbf{v}}_m) - \beta_m)\}$. A solution is given by $\nabla_{\widetilde{\mathbf{v}}} F(\widetilde{\mathbf{v}}^\dagger) = \sum_{m=1}^M \frac{1}{f_m(\widetilde{\mathbf{u}}_m^\dagger) - \beta_m} \nabla f_m(\widetilde{\mathbf{u}}_m^\dagger) = \mathbf{0}$ because F is convex. From (1) and (2), it follows that the analytic center coincides with the solution obtained by solving the centralized problem with $w_m = 1/(f_m(\widetilde{\mathbf{v}}_m^\dagger) - \beta_m)$. □

B. NETWORK PHYSICAL PARAMETERS

The network physical parameters, saturation S in vehicles per minute and direction rate ρ in percentage, are: $S_{1,\emptyset 1} = 5$, $S_{1,\emptyset 2} = 30$, $S_{1,\emptyset 3} = 15$, $S_{2,1} = 20$, $S_{2,3} = 25$, $S_{3,1} = 5$, $S_{3,4} = 25$, $S_{4,\emptyset 1} = 30$, $S_{4,\emptyset 2} = 15$, $S_{5,1} = 5$, $S_{5,4} = 7$, $S_{6,1} = 9$, $S_{6,5} = 7$, $S_{7,6} = 7$, $S_{7,\emptyset 1} = 10$, $S_{8,7} = 7$, $S_{8,\emptyset 1} = 5$, $\rho_{2,1,\emptyset 1} = 25$, $\rho_{2,1,\emptyset 2} = 70$, $\rho_{2,1,\emptyset 3} = 25$, $\rho_{2,3,1} = 40$, $\rho_{2,3,4} = 80$, $\rho_{3,1,\emptyset 1} = 7$, $\rho_{3,1,\emptyset 2} = 7$, $\rho_{3,1,\emptyset 3} = 20$, $\rho_{3,4,\emptyset 1} = 70$, $\rho_{3,4,\emptyset 2} = 70$, $\rho_{5,1,\emptyset 1} = 8$, $\rho_{5,1,\emptyset 2} = 8$, $\rho_{5,1,\emptyset 3} = 20$, $\rho_{5,4,\emptyset 1} = 30$, $\rho_{5,4,\emptyset 2} = 30$, $\rho_{6,1,\emptyset 1} = 60$, $\rho_{6,1,\emptyset 2} = 15$, $\rho_{6,1,\emptyset 3} = 35$, $\rho_{6,5,1} = 60$, $\rho_{6,5,4} = 90$, $\rho_{7,6,1} = 90$, $\rho_{7,6,5} = 90$, $\rho_{8,7,6} = 90$, $\rho_{8,7,\emptyset 1} = 50$.

Multi-agent Traffic Light Control Framework Based on Direct and Indirect Coordination

Takashi Shirai,
Yujiro Konaka
ISIR, Osaka Univ.
8-1, Mihogaoka, Ibaraki,
Osaka, 567-0047 JAPAN
shirai@ai.sanken.osaka-u.ac.jp
konaka@ai.sanken.osaka-u.ac.jp

Junji Yano,
Shigeki Nishimura,
Kouji Kagawa,
Tetsuo Morita
Information and Communication Labs
Sumitomo Electric Industries, Ltd.

Masayuki Numao,
and
Satoshi Kurihara
ISIR, Osaka Univ.
8-1, Mihogaoka, Ibaraki,
Osaka, 567-0047 JAPAN
numao@sanken.osaka-u.ac.jp
kurihara@sanken.osaka-u.ac.jp

ABSTRACT

Traffic congestion is a serious problem in urban life causing social problems such as time loss, economical loss, and environmental pollution. Therefore, we propose a multi-agent-based traffic light control framework for intelligent transport systems. For smooth traffic flow, real-time adaptive coordination of traffic lights is necessary, but many conventional approaches are of the centralized control type and do not have this feature. Our multi-agent-based control framework combines both indirect and direct coordination. Reaction to dynamic traffic flow is attained by indirect coordination, and green-wave formation, which is a systematic traffic flow control strategy involving several traffic lights, is attained by direct coordination. We show the detailed mechanism of our framework and verify its effectiveness through comparative evaluation through simulation.

Categories and Subject Descriptors

I.2 [**ARTIFICIAL INTELLIGENCE**]: Distributed Artificial Intelligence

General Terms

Algorithms, Performance

Keywords

ITS, intelligent traffic control, multi-agent coordination

1. INTRODUCTION

Traffic congestion in urban areas causes serious problems in terms of economic loss, time loss, and environmental pollution. Major solutions to eliminate traffic congestion are intelligent car navigation [5][8] and traffic light control. The former technology, such as the Vehicle Information and Communication System (VICS) in Japan, and the Probe-Car Information System, has progressed rapidly. VICS is an innovative information and communication system that enables one to receive real-time road traffic information, This information is edited and processed by the VICS Center and displayed on the navigation screen in text or graphical form. The Probe-Car Information System uses cars as mobile sensors for collecting data, which are sent to a central server to produce new information for avoiding congestion and providing efficient navigation.

Even though computer-based traffic light control systems are based on centralized control, they have disadvantages. Therefore, we focus on improving such traffic light control systems.

The basic steps in the traffic light control design process is deciding on phases and calculating the control parameters. The various traffic flows at an intersection are allowed to move in phases. Each phase of a signal cycle is devoted to only one traffic flow. The control parameters define the timing of switching phases. The major control parameters are "cycle length", "clearance", "split" and "offset" in the traffic light control system.

- Cycle length is the time required for one cycle of traffic light phases (e.g. green -> yellow -> red). Figure 1 shows an example of this.
- Clearance is the time it takes to clear an intersection area.
- Split is the percentage of cycle length allocated to each traffic light phase.
- Offset is the time lag between green indications of adjacent traffic lights. Figure 2 shows an example of offset control (green-wave formation).

Traffic congestion mainly begins at intersections. Traffic flow fluctuates dynamically during morning and evening rush

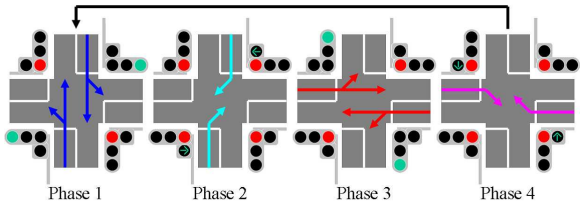


Figure 1: Example of phases and cycle

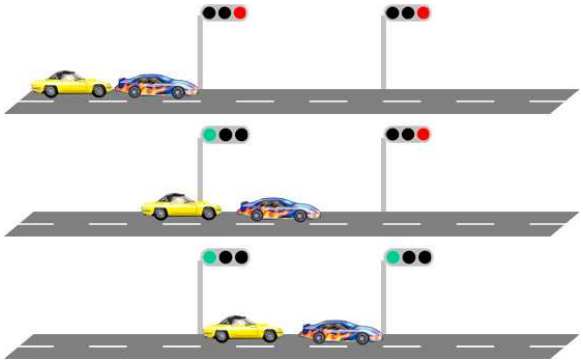


Figure 2: Example of offset control

hours. Moreover, unexpected events, such as road accidents, and unexpected popular events dynamically cause traffic congestion. Therefore, it is important to be able to appropriately align these parameters at any time.

The current traffic light control systems can be classified into two types; static, which use the above parameters calculated beforehand, and dynamic, in which traffic flow is monitored and the values of the parameters are adapted. These parameters are aligned dynamically (as in MODERATO in Japan and OPAC[4] in US). In the static type, several parameter sets are calculated beforehand according to each traffic flow situation during rush hour, daytime or nighttime. While this type is effective in envisioning changes in traffic flow, it cannot deal with unexpected situations. In the dynamic type, traffic flow is detected by sensors installed along roads, and traffic lights are controlled based on this sensor information. However, current systems are of a centralized control type, which is unsuitable for the management of dynamic complex traffic flow.current systems are of a centralized control type

In MODERATO, real-time information is not utilized appropriately. This is mainly due to its algorithm, which selects a favorable parameter set matched to each traffic flow situation from the several parameter sets calculated beforehand by off-line simulation.

For offset control in current traffic control systems, when one-way traffic flow becomes quite high during the morning rush hour from residential areas to urban areas, green-wave offset control (through-band offset control) is adopted. In green-wave offset control, offset timing of several traffic lights are aligned to allow each car to move without stopping at a traffic light. Ideally, the group of traffic lights that make up the green-wave control should be organized dynamically. However, in the current systems, these groups are pre-defined, and it is impossible to freely construct the green-wave control anywhere.

The essential factor for next generation traffic light control systems are real-time adaptability, to be able to quickly react to the dynamic traffic flows. To achieve this, we believe that a framework in which each traffic light is autonomous and coordinates with others to react to dynamic traffic flow is necessary.

We propose a traffic light control framework based on a multi-agent paradigm to react to dynamic traffic flow, decrease the number of cars stopping at a red light, and adaptively form a green-wave control group. An agent is implemented at each intersection for controlling the several traffic lights that belong to that intersection. Our framework combines indirect and direct coordination. That is, reaction to dynamic traffic flow is attained by indirect coordination using a spring model based on stigmergic dynamics, and green-wave organization is achieved by direct coordination.

In section 2 we discuss related studies and explain the major control parameters of traffic lights in Section 3. In Section 4, we discuss our proposed framework and show the evaluation results. We conclude our discussion in Section 5.

2. RELATED STUDIES

One approach for obtaining optimized parameters is using a genetic algorithm (GA). Takahashi et al. proposed an offset optimization model using a GA [13]. In this model, offset values of traffic lights were used to represent a chromosome. Sánchez et al. proposed another parameter optimization model using a GA [10]. In this study, optimized cycle length, clearance time, split, and offset could be calculated. Mikami et al. proposed a multi-agent-based model in which reinforcement learning is used to optimize the parameters [6]. In this model each agent performs reinforcement learning independently, and each parameter set, which is calculated by each agent, is aggregated to the central control module. Then the central control module uses a GA to find the optimized parameter set. Balaji et al. also proposed a multi-agent-based centralized optimization methodology using a GA [1]. Kosonen et al. proposed a multi-agent real-time traffic light control system using fuzzy inference, and Schmöcker et al. proposed a multi-objective traffic light control method using fuzzy logic [12]. The membership functions of fuzzy logic are optimized using a GA executed in a microscopic traffic simulator. These GA-based approaches are attractive when there are many parameters to be optimized, but require large calculation cost and time until convergence. A more optimized solution can be derived with centralized calculation approaches, but these approaches do not exhibit real-time adaptability.

Another approach is a stochastic control model. Yu et al. achieved traffic light parameter optimization as a decision making problem of a controlled Markov process [14]. They say that the stochastic approach is suitable for the traffic light control problem, especially under the conditions of high volume but not saturated traffic demand. However, when the size of the road network is increased, the dimension number of the proposed control framework increases, and more

memory space and computation time become necessary.

On the other hand, there are several related studies based on the distributed approach to achieve real-time adaptability. Nishikawa proposed an offset control algorithm based on the phase oscillator model [9]. In this study, the functions of each traffic light were modeled as oscillators and traffic lights were coordinated through synchronization of each oscillator. Satoh proposed a split control model based on the spring model [11]. In this study, traffic flow was assumed to be the same as the force of a spring. The split ratio was modeled as the force balance of a spring. Coordination between adjacent traffic lights was also modeled as a spring model. Traffic lights were connected with a spring, and the split ratio of traffic light was assumed to be the same as the force of a spring. Oliveira proposed a multi-agent-based split control approach [3]. Each agent calculates the congestion degree independently and controls its split value to decrease the total congestion degree.

These conventional approaches are all attractive, but their performances were evaluated using quite simple and small-scale road environments, and most of them concerned about only a few parameters. Therefore, it is difficult to apply them to more complex and large-scale environments.

3. MULTIAGENT CONTROL

In this chapter, we describe our multi-agent based traffic light control framework, our proposed split control model with spring model by indirect coordination, and our proposed offset control and green-wave formation model by direct coordination. As for agent based approach, useful survey was done in [2].

Generally, indirect coordination exhibits adaptability and low coordination cost but optimality cannot be ensured. On the other hand, direct coordination exhibits optimality but requires high coordination cost and longer convergence time than indirect control. A traffic light control consists of split and offset controls. To quickly reduce the waiting queue of cars at an intersection, control of the split value of each traffic light is necessary. Therefore, real-time adaptability is necessary for split control. On the other hand, to form a green-wave control group with several traffic lights, some deliberate coordination is necessary. Therefore, in our proposed framework, split control is attained using the indirect type of coordination, and offset control is attained using the direct type of coordination.

For split control, each agent calculates the split value autonomously by referring waiting queue of cars at a traffic light it directly controls. Each agent does not interfere with neighbor agents. That is, there is no direct coordination cost; therefore, real-time control can be achieved. On the other hand, functions of each agent indirectly affect its neighbor agents through the change of traffic flow. This indirect coordination is generally called "stigmergy[1]".

For offset control in current traffic control systems (e.g. MODERATO), several groups that may perform green-wave control are pre-defined, so dynamic formation is impossible. On the other hand, in our proposed framework, green-wave control formation can be dynamically established anywhere.

In normal daily traffic flow, each agent functions based on the indirect coordination mode. However, when the traffic flow balance collapses near certain agents, the agents change their coordination mode to direct coordination mode to form a green-wave control formation. Therefore, such direct coordination of an agent group can be seen as interfering with the indirect coordination of agents. However, indirect coordination has an advantage against such interference. The important point is affinity of both coordination types.

Cycle length and clearance were not considered in most related studies. However, both parameters also affect traffic flow; therefore, we focused on both parameters. We adopted the Webster cycle length approximation formula, which is also adopted in MODERATO (details are discussed in Section 5). Clearance length is a constant value.

3.1 Definition of agent

Agent A_i, which controls the traffic lights of $intersection_i$, collects the following information:

- Distance $l_{i,j}$ between $intersection_i$ and its directly connected $intersection_j$.
- Traffic flow (number of cars) per unit time $intersection_i$ to $intersection_j$, which is defined as $p_{(i,j)}$.
- Average velocity of cars heading from $intersection_i$ to $intersection_j$ is defined as $v_{i,j}$.
- C_i is the cycle length, S_i is the split value, and O_i is the offset value of $intersection_i$.
- T_i shows the start time of C_i, and current step count is t_i.
- Total traffic flow into $intersection_i$ is defined as $P_(i) = \sum_j p_{(j,i)}$

Traffic flow p is calculated based on the total traffic flow of the last five cycles that showed the best effect from the results from a pre-exploratory experiment. Each agent calculates and updates these values at the beginning of every cycle.

3.2 Cycle length control

Cycle length is controlled depending on whether each agent performs direct or indirect coordination. When the agent is in indirect coordination mode, cycle length is calculated using Webster's equation.

$$C_o = \frac{1.5L + 5}{1 - \lambda} \quad (1)$$

where C_o is the optimal cycle length, L is the clearance length, and λ is the ratio of p to the saturation traffic flow.

On the other hand, when the organization of green-wave formation, which is formed by several agents, becomes necessary, the coordination mode of these agents becomes direct, and the cycle length of these agents becomes the same.

[1]The term "stigmergy" was introduced by French biologist Pierre-Paul Grass in 1959 to refer to termite behavior.

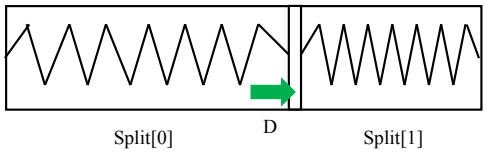


Figure 3: 2 phase spring model

Details of cycle length calculation are discussed in Section 3.4.

3.3 Split control by indirect coordination

Each agent at an intersection observes the traffic flow of each road connected to the intersection during the green phase of each road. The agent then calculates the split ratio based on the proposed spring model so as to equalize the traffic flow of each road. At this point, each agent calculates its split value by using only local information and does not directly interact with others, exhibiting real-time characteristics and low communication cost.

Now, we consider a crossroad and 2-phase traffic light (red and green) in this intersection. The split ratio of one of the two phases $phase_1$ is defined as $split[0]$, and the other phase $phase_2$ is defined as $split[1] = 1 - split[0]$.

Figure 3 shows a diagrammatic illustration of our spring model. Traffic flow is considered as force. The spring equation is defined as.

$$K(C - Csplit[0]) + D = K(C - Csplit[1]), \tag{2}$$

where C is the cycle length, D is the difference in traffic flow between $phase_1$ and $phase_2$, and K is the spring constant, which is defined as the number of cars waiting for the red light phase during one step.

$$split[0] = \frac{(KC + D)}{2KC} \tag{3}$$

Therefore, we can calculate $split[0]$ and then $split[1]$. However, Eq. 3 may give a split value of $split[0] \geq 1$ or $split[1] \geq 1$, where $split \geq 1$ means that the traffic light cannot change the phase. Therefore, we define the maximum value of $split$ as 0.9 and the minimum value as 0.1.

3.4 Offset control by direct coordination

The offset is calculated based on the traffic flow between two adjacent intersections. When the condition for constructing the green-wave formation is satisfied for a certain agent, the agent tries to start direct coordination with its adjacent agents by sending them a coordination request message. First, we define three agent modes. Then we explain the offset equations and show the sequence of green-wave formation.

3.4.1 Agent's mode

Each agent consists of three types of modes depending on the condition of its adjacent agents and amount of traffic flow it controls.

- Independent mode: An agent does not interact with the green-wave formation.
- Master mode: An agent in this mode becomes the center of coordination and is called the "master agent". When the construction of the green-wave formation is satisfied for a certain agent, that agent's mode changes from independent to master.
- Fellow mode: When a certain agent accepts the coordination request from the master agent, the mode of this agent changes from independent to fellow.

3.4.2 Offset calculation

The offset is calculated based on the difference between inbound and out-bound traffic flow on the road between two adjacent intersections. When the difference between in- and out-bound flows reaches a certain value, the offset is calculated to give priority to the more congested direction.

We define p_l as $p_{(i,j)}$ or $p_{(j,i)}$, whichever is the larger, and p_s as $p_{(i,j)}$ or $p_{(j,i)}$, whichever is the smaller. The notations γ and δ are thresholds of traffic flow (γ is bigger than δ). For $\frac{p_l}{p_s} \geq \gamma$, we consider only the more congested direction, and the relative offset value O_r is defined as

$$O_r = \frac{l_{(i,j)}}{v_l}, \tag{4}$$

where v_l is the velocity of the more congested traffic flow. On the other hand, in case of $\gamma > \frac{p_l}{p_s} > \delta \geq 1$, it is necessary to consider both flow directions. Therefore, the relative offset value O_r is defined as

$$O_r = \frac{l_{(i,j)}}{v_l} \frac{(\frac{p_l}{p_s} - \delta)}{(\gamma - \delta)} \tag{5}$$

Finally, when a master agent of intersection A_i sends a coordination request message to an independent agent of its adjacent intersection A_j, the offset value, which is assumed to be A_j, is $O_{(i,j)} = -Or$ $(p_{(i,j)} \geq p_{(j,i)})$ or $O_{(i,j)} = Or$ $(p_{(i,j)} < p_{(j,i)})$. As mentioned above, when A_j accepts this coordination request, its mode changes to fellow.

3.4.3 Direct coordination process

We describe the constructing sequence of green-wave formation through the coordination of agents.

All independent agents have the possibility of becoming a master or fellow agent. The condition for an agent A_i to become a master agent A_{xc} is that A_i is in independent mode and $P_i > \alpha$, or A_i is fellow mode and $P_i >= P_{xc}$. P_{xc} is defined as a master agent A_{xc}'s total traffic flow. The notation α is a threshold of traffic flow per unit of time to become master mode.

Step1 If $p_{(ic,j)} \geq \beta$ or $p_{(j,ic)} \geq \beta$, master agent A_{ic} starts direct coordination to control the offset value with its adjacent agent A_j. Then A_{ic} sends the calculated offset value $O_{(ic,j)}$, total traffic flow P_{ic}, start time of cycle T_{ic}, and distance $d_{(ic,j)}$ between A_{ic} and A_j to A_j. The notation β is another threshold of traffic flow per unit of time.

Figure 4: Agent mode
In this simulator, an agent's mode is denoted with three colors. Left is a master agent, Middle is a fellow agent, and Right is an independent agent. When the clock hand points to the colored area, the traffic light's phase is $phase_1$. When the clock hand points to the white area, the phase is $phase_2$.

Step2 If agent A_j is in independent mode, and if $t_j \leq \epsilon C_j$ or $t_j \geq (1-\epsilon)C_j$, it accepts the request from A_{ic}. The notation ϵ is a threshold of time path between the start and the time when A_j will accept the request.

On the other hand, if agent A_j is in the fellow mode with another master agent A_{yc}, the conditions for agent A_j to accept the request from A_{xc} are $t_j \leq \epsilon C_j$, or $t_j \geq (1-\epsilon)C_j$ and $P_{ic} > P_{yc}$, or $P_{ic} = P_{yc}$ and $l_{(ic,j)} > l_{(yc,j)}$.

Moreover, if agent A_j itself is a master A_{jc}, the condition for A_{jc} to accept the request from master agent A_{ic} are $P_{jc} < P_{ic}$ and $t_j \leq \epsilon C_j$ or $t_j \geq (1-\epsilon)C_j$. If A_j accepts the request from A_{ic}, A_j becomes a fellow agent of A_{ic}. Then, T_j is changed to T_{ic}, and O_j is changed to $O_{ic} + O_{(ic,j)}$.

Step3 Then fellow agent A_j checks the traffic flow $p_{(j,k)}$ and $p_{(k,j)}$, where k is the intersection adjacent to intersection j. Then if $p_{(j,k)} \geq \beta$ or $p_{(k,j)} \geq \beta$, A_j sends the coordination request to agent A_k, which is the adjacent agent to A_j, similar to a bucket brigade. Agent A_j sends the calculated offset value $O_{(j,k)}$, total traffic flow P_{ic}, start time of cycle $T_k = T_j = T_{ic}$, and distance $d_{(ic,k)} = d_{(ic,j)} + d_{(j,k)}$ to A_k.

Step4 If A_k accepts the request from A_j, the mode of A_k is changed from independent to fellow.

Step5 When the bucket brigade process terminates, the green-wave formation consisting of one master agent and several follow agents begins coordinated offset control.

4. EXPERIMENTS AND RESULTS

4.1 Traffic Simulator

We verified our traffic light control framework through simulation to confirm its effectiveness. The movement of cars is expressed with the Nagel-Schreckenberg (NS) model using cellular automaton rule 184[2] [7]. In the simulator, the unit of time is called "step(= 0.3 sec)" and the unit of distance is "cell". Each car flows into the simulator from the cell on the edge of the simulator according to an inflow probability. The simulator consisted of roads (edges) and intersections

[2]Rule 184 can be used as a simple cellular automaton model for traffic flow in a single lane of a road. In this model, cars can move in a single direction, stopping and starting depending on the cars in front of them.

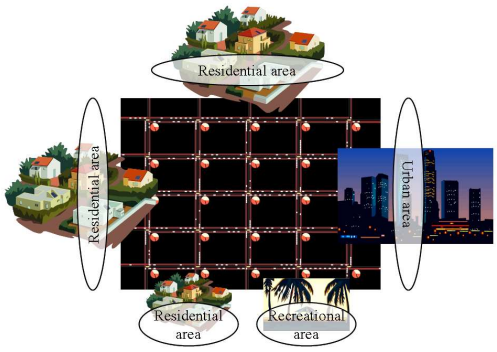


Figure 5: Experiment 3: Road Network

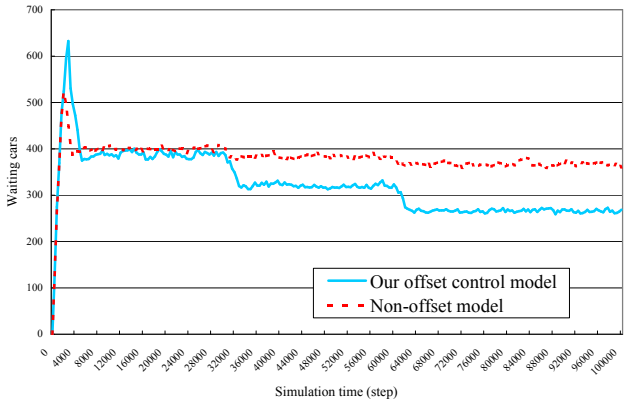


Figure 6: Experiment 1: Transition of Waiting Cars

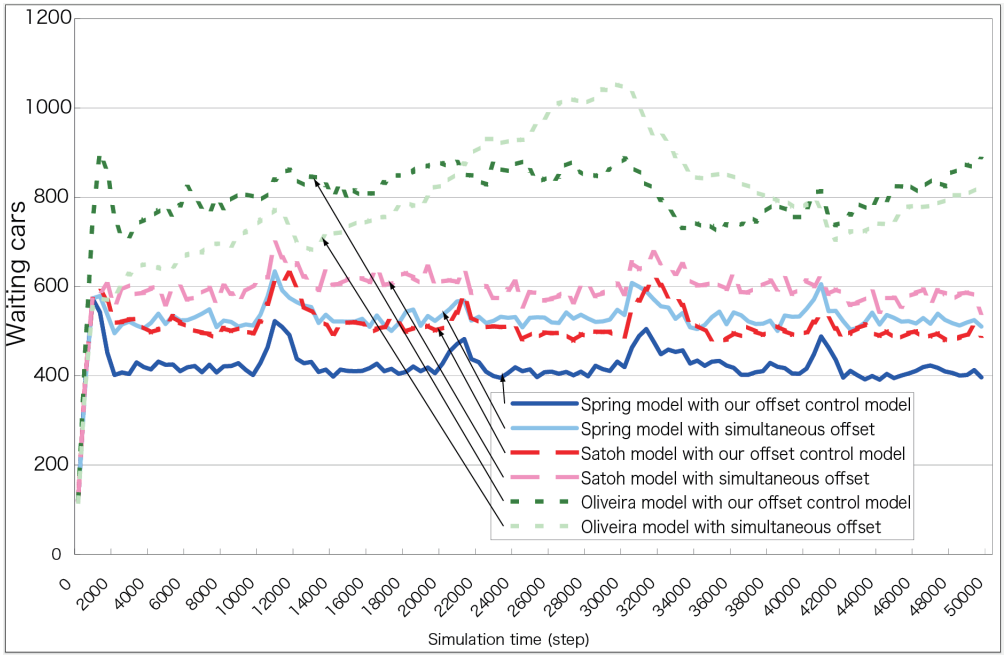


Figure 7: Experiment 2: Transition of Waiting Cars

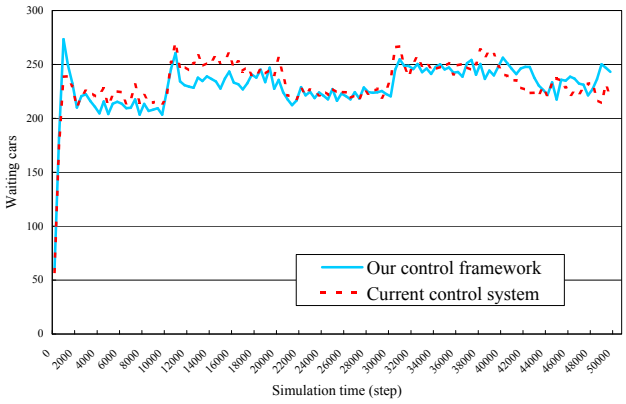


Figure 8: Experiment 3: Transition of Waiting Cars

(nodes). The road network was a grid-type network. Each intersection had the coordinate (x, y). We prepared 1×20 (20 intersections) and 10×10 (100 intersections) networks. The distance between adjacent intersections was 50 cells. The velocity of cars was 1 cell per step. We set each threshold value, which is the best value, based on pre-exploratory experiments as follows: α=0.25, β=0.125, γ=1.5, δ=1.1, ϵ=0.2. The simulation environment was Intel Core 2 Duo 2.40-GHz CPU and 2.00-GB RAM.

4.2 Experiments

Experiment 1

To confirm the effectiveness of green-wave control, we prepared a 1×20 straight road network. Cars enter the network from cells at both the east and west edges. Both initial in-

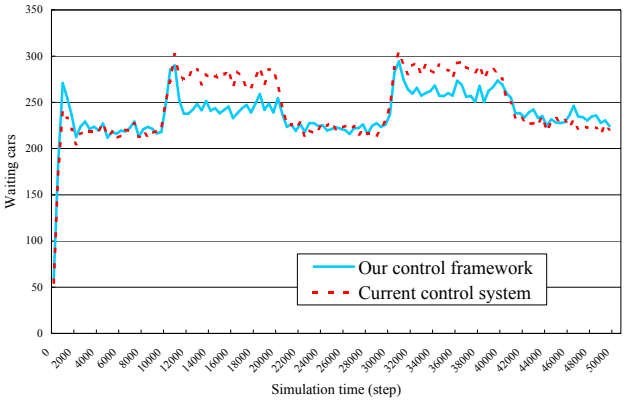


Figure 9: Experiment 4: Transition of Waiting Cars

flow probabilities were 22.5%. We then decreased the inflow probability from the east edge by 5% after 30000 steps, and decreased it an additional 5% after 60000 steps. We fixed the cycle length (400 time steps = 2 min) and split value ($split[0] = split[1] = 0.5$). We compared our offset control model with a non-offset model (all traffic lights control each phase at the same time).

Experiment 2

We verified the effect of our spring model-based split control algorithm. We selected two already proposed related split control models, the Satoh and Oliveira models, for comparison. Because our spring model is based on indirect coordination, each agent does not receive all the information and does not directly interact with other agents. The Satoh model is a direct coordination-type model. Each agent also does not receive all the information but it directly interacts with other agents. The Oliveira model is a direct coordination-type model. Each agent interacts with other agents, but it does not receive all the information.

In addition to these comparative experiments, we also verified the affinity of these three split control models and our offset control model. We prepared a 10×10 road network. We also prepared one input cell having 20% inflow probability, two input cells having 15% inflow probability, and three input cells having 10% inflow probability (total six cells). All other cells had 2.5% inflow probability. For verification of the real-time adaptability of our framework, we replaced these six cells and the other six cells having 2.5% inflow probability after every 10,000 steps. In this experiment, we used Webster's cycle length control.

Experiment 3

We compared our traffic light control framework with the current operating traffic control system model and confirmed its effectiveness.

We assumed this road network had four residential areas, one urban area, and one recreational area, as shown figure 5. We prepared the following traffic scenarios:

scenario 1 Traffic flow just before morning rush hour: Car flow is generally not so heavy, but the flow to the urban area is little heavier.

scenario 2 Traffic flow during morning rush hour: Many cars congregate on main roads heading into the urban area.

scenario 3 Traffic flow after morning rush hour: Car flow to recreational area becomes high.

scenario 4 Traffic flow during daytime: Car flow is not so heavy.

scenario 5 Traffic flow during evening rush hour: Many cars from urban area head to the residential areas.

We changed the traffic flow according to the above scenarios after every 10,000 steps. We evaluated a current operating traffic control system as a comparison. It prepared some traffic light control parameters beforehand and changed them according to a pre-defined time schedule. To prepare these traffic light control parameters beforehand, we conducted the simulation for each scenario with our framework and obtained five parameter sets for each.

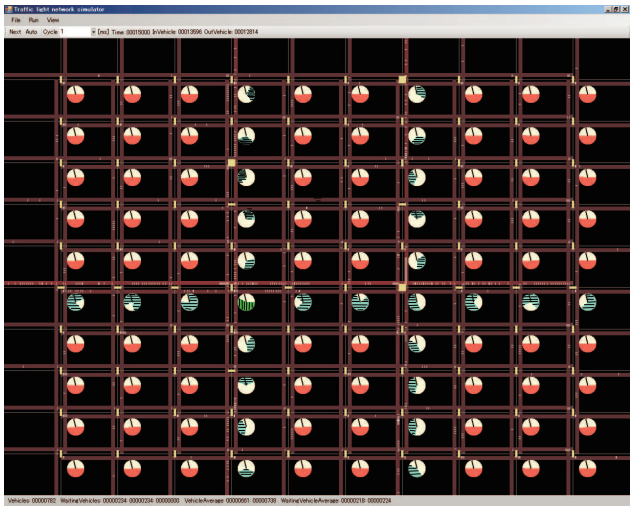


Figure 10: Experiment 3: Screenshot of Simulator during Simulation

Experiment 4

We verified the real-time adaptability of our framework against sudden changes in traffic flow. We modified scenarios 2 and 4 of experiment 3 as follows:

New scenario 2 The road between intersections (6,5) and (7,5) was suddenly closed. Therefore, cars had to go the urban area by bypassing this section.

New scenario 4 The road between intersections (6,5) and (6,7) was suddenly closed. Therefore, cars had to return to the residential areas from the recreational area by bypassing this section. Moreover, we assumed that the recreational area was crowded more than usual.

We changed the traffic flow according to the above scenario (scenario 1 -> new 2 -> 3 -> new 4 -> 5) after every 10,000 steps. We compared our traffic light control framework with a current operating control system, as in experiment 3.

4.3 Results

Figure 6 shows the evaluation results of experiment 1. The average total waiting queue of cars from simulation start to end was 382 with the non-offset model, and 318 cars with our offset control model. This result shows that the green-wave formation can make traffic flow more smoothly. This formation is more effective when the difference in both traffic flows increases.

Figure 7 shows the evaluation results of experiment 2. When we executed our offset control model, the average total waiting queue of cars from simulation start to end was 419 with our spring model, 507 with the Satoh model, and 797 with

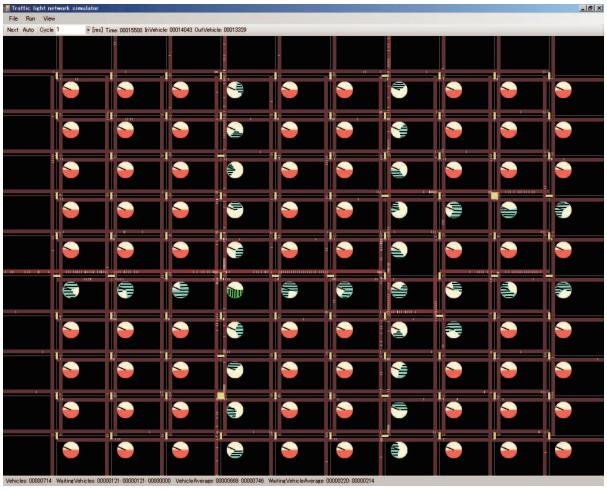


Figure 11: Screenshot of simulator of experiment 4 (New green-wave formation is constructed according to change in traffic flow)

the Oliveira model. When we did not use our offset model, the average total waiting queue of cars was 524 with our spring model, 590 with the Satoh model, and 786 with the Oliveira model.

In the Oliveira model, since the split value is fixed and does not flexibly change, the waiting queue of cars becomes longer. Moreover, This model could not attain smooth traffic flow, even if it was combined with our offset control model. On the other hand, with our spring model and the Satoh model, the waiting queue of cars did not become long. In addition, the length of the queue decreased when these models were combined with our offset control model, making traffic flow more smoothly.

Finally, when our offset control model was combined with our spring model the total waiting queue of cars decreased 20% compared with only our split control model. On the other hand, with the Satoh model, the queue length decreased by 16%. As mentioned above, our spring model is an indirect coordination-type model, and the Satoh model is a direct coordination-type model. Therefore, this result shows that when the split control model is combined with the offset control model using direct type coordination as the upper layer, indirect-type coordination is a more desirable approach for the split control model as the lower layer.

Figure 8 shows the evaluation results of experiment 3. The total average waiting queue of cars from simulation start to end was 228 with our control framework and 231 with the current operating control system. Figure 10 shows a screenshot of the simulator during the simulation of experiment 3. The green, aqua and orange sectors describe an agent's mode (see Figure 4). Agents coordinated with others and formed a subarea and a green-wave formation suitable for traffic flow: many cars congregated on main roads heading to the urban area.

Figure 9 shows the results of experiment 4. The total average waiting queue of cars from simulation start to end was 236 with our control framework and 244 cars with the current operating control system. Figure 11 shows a screenshot of the simulator during the simulation of experiment 4. Agents coordinated with others and formed a subarea and a green-wave formation for changed traffic flow. We could observe that agents on bypasses threw in the coordination, and several agents that had coordinated dissolved the coordination. From 10,000 to 20,000 steps, the queue length was 246 cars with our control framework and 276 cars with the current operating control system, and from 30,000 to 40,000 steps, the queue length was 263 cars with our control framework and 285 cars with the current operating control system.

When traffic flow was normal (situation in experiment 3), there was not much difference between our control framework and the current operating control system having several parameter sets beforehand because there was no unexpected event in this experiment. However, when an unexpected event occurs, such as a traffic accident, the difference in the waiting queue of cars between both our framework and the current system becomes quite large, as observed in new scenarios 2 and 4 in experiment 4. This is because in our framework, green-wave formation can be organized anywhere. This result shows that our control framework can deal with sudden change in traffic flow in contrast to the current operating control system.

With this simulation, we believe that we could show the fundamental effectiveness of our traffic control framework based on a multi-agent paradigm. Several parameters used in the simulation were optimized to the simulation environment. Therefore, to apply our framework to a real traffic system, a pre-experiment to find the appropriate parameter values is necessary. For example, in the current operating control system, MODERATO, many parameters are also necessary and they are set empirically and corrected through actual field observation. Therefore, we believe that a similar approach can also be applied to our framework.

5. CONCLUSIONS

We proposed a multi-agent-based traffic light control framework. To construct dynamic complex distributed systems, top down control and bottom up control are both necessary. In the proposed framework, we combined direct coordination as top-down control, and indirect coordination as bottom-up control. The important point is affinity of both coordination types. By comparative evaluation through simulation, we could verify the basic effectiveness of our framework. In our framework, direct coordination can be seen as interfering with indirect coordination. Mitigating this interference was our strategy. However, another strategy for making both coordination methodologies interact efficiently may be possible. Therefore, we will consider this in the future. Finally, the current simulated road network is a simple lattice structure, so we will perform more evaluations with large-scale and various road networks in the future.

6. REFERENCES

[1] P. Balaji, G. Sachdeva, and D. Srinivasan. Multi-agent system based urban traffic management. *2007 IEEE*

Congress on Evolutionary Computation, pages 1740–1747, September 2007.

[2] A. L. C. Bazzan. Opportunities for multiagent systems and multiagent reinforcement learning in traffic control. *Autonomous Agents and Multi-Agent Systems*, 18(3):313–341, June 2009.

[3] D. de Oliveira, A. L. Bazzan, and V. Lesser. Using cooperative mediation to coordinate traffic lights: a case study. *Proceeding of Fourth International Conference on Autonomous Agents and Multiagent Systems*, pages 463–469, July 2005.

[4] N. Gartner, F. Pooran, and C. Andrews. Implementation of the opac adaptive control strategy in a traffic signal network. *Proc. of Intelligent Transportation Systems*, pages 195–200, 2001.

[5] S. Kurihara, H. Tamaki, M. Numao, K. Kagawa, J. Yano, and T. Morita. Traffic congestion forecasting based on pheromone communication model for intelligent transport systems. *Proceeding of IEEE Congress on Evolutionary Computation*, pages 2879–2884, May 2009.

[6] S. Mikami and Y. Kakazu. Genetic reinforcement learning for cooperative traffic signal control. *1994 IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on Evolutionary Computation*, 1:223–228, June 1994.

[7] K. Nagel and M. Schreckenberg. A cellular automaton model for freeway traffic. *Journal of Physics I France*, 2:2221–2229, December 1992.

[8] T. Nakata and J. Takeuchi. Mining traffic data from probe-car system for travel time prediction. *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 817–822, August 2004.

[9] I. Nishikawa. Dynamics of oscillator network and its application to offset control of traffic signals. *2004 IEEE International Joint Conference on Neural Nwtworks*, 2:1273–1277, July 2004.

[10] J. J. Sánchez-Medina, M. J. Gálan-Moreno, and E. Rubio-Royo. Traffic signal optimization in la almozara district in saragossa under congestion conditions, using genetic algorithms, traffic microsimulation, and cluster computing. *IEEE Transaction on Intelligent Transportation Systems*, 11(1):132–141, March 2010.

[11] K. Satoh, R. Nagaoka, N. Yasuba, J. Yano, K. Kagawa, T. Morita, M. Numao, and S. Kurihara. Construction of autonomous traffic light control system using multi-agent model (japanese). *The 22nd Annual Conference of the Japanese Society for Artificial Intelligence CD-ROM*, June 2008.

[12] J.-D. Schmöcker, S. Ahuja, and M. G. Bell. Multi-objective signal control of urban junctions - framework and a london case study. *Transportation Research Part C*, 16(4):454–470, August 2008.

[13] S. Takahashi, H. Nakamura, H. Kazama, and T. Fujikura. *Adaptive traffic signal control for the fluctuations of the flow using a genetic algorithm*. WIT PRESS, 2002.

[14] X.-H. Yu and W. Recker. Stochastic adaptive control model for traffic signal systems. *Transportation Research Part C*, 14(4):263–282, August 2006.

Comparing context-aware routing and local intersection management

Adriaan W. ter Mors Delft University of Technology
Mekelweg 4, Delft, The Netherlands
a.w.termors@tudelft.nl

ABSTRACT

In multi-agent routing, there is a set of mobile agents each with a start location and destination location on a shared infrastructure. An agent wants to reach his destination as quickly as possible, but conflicts with other agents must be avoided. We have previously developed a single-agent route planning algorithm that can find a shortest-time route that does not conflict with any previously made route plans.

In this paper, we want to compare this route planning approach with non-planning approaches, in which intersection agents determine which agent may enter an intersection next, and where the agent will subsequently go (given its destination). When making these decisions, the intersection agents use only locally observed traffic information.

Our experiments show that context-aware routing produces more efficient results in case no incidents disrupt the execution. However, in the face of unexpected incidents, the performance of the intersection management policies proves very robust, while context-aware routing only produces good results when coupled with effective plan repair mechanisms.

Categories and Subject Descriptors

I.2.11 [**Computing methodologies**]: Multiagent systems

General Terms

Algorithms, Experimentation

Keywords

route planning, traffic agents, simulation

1. INTRODUCTION

In this paper we will discuss the problem of multi-agent route planning, in which there are multiple mobile agents each with a start and destination location on a roadmap. The roadmap consists of intersections and lanes connecting the intersections, and each agent wants to find a route that will bring it to its destination as quickly as possible.

In previous work [20], we developed a prioritized route planning approach in which agents are first assigned a priority (typically randomly, or based on their arrival time), and subsequently plan a route that is optimal for themselves and does not create any deadlock with any of the higher-priority agents. We named our algorithm *context-aware routing*, as each planning agent is aware of its context, which is the set of reservations from route plans of higher-priority agents. Deadlock prevention is especially relevant in roadmaps with bi-directional roads that can be traversed in only one direction at the same time (e.g., when the roads are not wide enough for two vehicles to travel side by side), for instance in application domains of automated guided vehicles [9] or airport taxi routing [6].

In this paper, however, we will focus on infrastructures in which all roads are directed, such as common in urban traffic control (cf. [1]), and investigate how different routing approaches influence congestion, and therefore the times the agents reach their destinations. We will compare our conflict-free routing approach with a number of local intersection management policies that we will define in section 4. These intersection management policies make routing decisions for the vehicles only on the basis of information that is local to the intersection, namely how many vehicles are waiting to enter the intersection, and how long they have been waiting.

1.1 Related work

The problem of finding an optimal set of conflict-free route plans is NP-hard [17], and all approaches that guarantee an optimal solution[1] that we know of only manage to find solutions for a handful of agents. In domains with larger numbers of agents, a common approach is to let the agents plan for themselves, usually one after the other, or by iteratively communicating about conflicting plans (cf. [14]). Silver [15], for example, presents an approach that is based on the straightforward idea of letting an agent perform an A* search, in which it checks whether the nodes it visits during search are not occupied by other agents at the time the agent would reach those nodes.

[1]Even for optimal approaches, there are often simplifying assumptions, for example dividing time up into 15-second intervals [3].

A problem with a straightforward A*-with-time approach is that it is unclear how many times a particular node must be visited during the search. The shortest-time path that finds the node unoccupied might not be extendible to the destination node in case all of the node's neighbours are occupied at the time (see also the example in section 3). We must therefore introduce the notion of a *free time window*, which is an interval during which the node is unoccupied. During search, we need only consider the shortest-time paths to each of the free time windows of a node, and then the single-agent routing problem can be solved optimally in polynomial time. Kim and Tanchoco [9] first developed and analysed such an algorithm, with a runtime complexity of $O(|R|^2|\mathcal{A}|^4)$, where R is the set of infrastructure *resources* (lanes and intersections), and $\mathcal{A}$ is the set of agents. Our context-aware route planning algorithm lowered that time complexity to $O(|R||\mathcal{A}|\log(|R||\mathcal{A}|) + |R|^2|\mathcal{A}|)$.

Further reductions in computation time can be achieved in case path and velocity planning are separated. In other words, if an agent first determines a path from start to destination, and then finds a conflict-free schedule along this path. Hatzack and Nebel [6] presented such an approach in an airport taxi routing scenario, whereas Lee et al. [10] consider an automated-guided-vehicle setting, in which agents first determine the k shortest paths between their respective start and destination locations, and then find conflict-free schedules along each of these paths, and choose the quickest. In previous work [19], we compared these *fixed-path scheduling* approaches with context-aware routing, and found that the performance of the former seriously degrades if too many agents plan to make use of the same roads; only if the k alternative routes constitute relevant alternatives can fixed path scheduling outperform context-aware routing.

So far, we have not compared our context-aware routing approach with non-planning approaches because these are either in some way restrictive of the infrastructure or how agents use it (e.g., in case agents are required to traverse the infrastructure in a loop [8]), or because the mechanisms to avoid or prevent deadlocks are time-consuming to run and implement (for instance the Petri-net-based approach from Fanti [5]). In this paper, we restrict ourselves to infrastructures that are less deadlock-prone, so we can compare the efficiency (in terms of global plan quality) of context-aware routing with other approaches.

In *urban traffic control*, most intersection management approaches make use of traffic lights, where the focus is on learning efficient behaviour for individual intersections [1]. Coordination is often limited to neighbouring intersections, although the implementation of higher-level agents to support the decision-making is also considered [2]. Another interesting line of work is that into Automated Intersection Management from the group of Peter Stone (see for instance [4]), in which intersections are not light-controlled, but vehicle agents place reservations for conflict-free trajectories in space and time over the intersection. Up until recently, work had focussed on the operation of a single intersection, but recent work by Hausknecht et al. [7] studies traffic phenomena when multiple intersections are linked together. Vasirani and Ossowski [22, 23] propose a market-based approach, in which intersection managers set prices according to current and future demand, and driver agents adapt their routes based on time and cost considerations. Although inspired by Dresner and Stone's Automated Intersection Management, Vasirani's research is moving from microscopic models, in which vehicle behaviour is affected by the movements of immediate neighbours, to mesoscopic models in which average traffic densities on roads determine traversal speeds. A difference between these urban traffic control approaches and our work (and other planning approaches like it — often originating from Automated Guided Vehicle (AGV) research) is the number of vehicles per intersection; in AGV applications, an intersection can be full with one vehicle (for instance in container terminal domains), while for automated intersection management, the intersections can hold many.

1.2 Contributions and organization

This paper makes two contributions to field of route planning and traffic control:

1. A comparison of the context-aware routing approach with local intersection management policies, both in terms of efficiency (measured in e.g. makespan and sum of agent plan costs) and in terms of robustness, i.e., how well the methods perform when unexpected incidents may disrupt the (planned) execution.

2. The definition of a set of simple local intersection management policies.

In section 2, we first present our model for context-aware routing, and then in section 3 we describe the context-aware route planning algorithm, as well as two plan repair mechanisms that are required when incidents can occur during plan execution. Section 4 presents our intersection management policies, and in section 5 we discuss our experimental results. Section 6 contains the conclusions and the ideas for future work.

2. MODEL

We assume a set $\mathcal{A}$ of agents that each have to find a quickest-time route from one location in the infrastructure to another. We model the infrastructure as a *resource graph* $G_R = (R, E_R)$, where resources in R can be roads, intersections, or interesting locations that the agents can visit. Formally, an agent can directly go from resource $r \in R$ to resource $r' \in R$ if the pair (r, r') is in the *successor relation* $E_R \subseteq R \times R$. A resource r has a capacity $c(r)$, denoting the maximum number of agents that can simultaneously make use of the resource, and a duration $d(r) > 0$ which represents the minimum time it takes for an agent to traverse the resource.

In this paper, we will restrict ourselves to (non-toroidal) *grids*, where two uni-directional lanes connect each pair of adjacent intersections. For these infrastructures, intersection resources have unit capacity and lane resources have capacity 8; minimum traversal times are 2 time units for the intersections and 7 for the lanes. In previous work (e.g. [18]), we have focused on bi-directional lanes, i.e., lanes on which travel in both directions is possible, though *not at the same*

time. In such a setting, however, the local intersection management policies we will evaluate in this paper would cause a deadlock almost instantly.

DEFINITION 1 (DEADLOCK). *Let $A^c = \{A_1, \ldots, A_m\} \subseteq \mathcal{A}$ be a set of agents, and let $R^c = \{r_1, \ldots, r_m\} \subseteq R$ be a set of resources such that, $\forall i \in \{1, \ldots, m\}$, agent A_i is on resource r_i, and $\forall i \in \{1, \ldots, m-1\} : (r_i, r_{i+1}) \in E_R$ and $(r_m, r_1) \in E_R$. The agents A^c are in a* deadlock *if and only if:*

1. *A_i's next resource is r_{i+1} ($\forall i \in \{1, \ldots, m-1\}$, or r_1 if $(i = m)$, and*
2. *$\forall i \in \{1, \ldots, m\}$ the number of agents on r_i equals $c(r_i)$.*

In our context-aware routing approach, deadlocks are prevented by ensuring that agents never make plans that exceed the resource capacities. An agent's plan consists of a sequence of resources, and a corresponding sequence of intervals in which to visit them.

DEFINITION 2 (ROUTE PLAN). *Given a start resource r, a destination resource r', and a release time t, a route plan is a sequence $\pi = (\langle r_1, \tau_1 \rangle, \ldots, \langle r_n, \tau_n \rangle)$, $\tau_i = [t_i, t_i')$, of n* plan steps *such that $r_1 = r$, $r_n = r'$, $t_1 \geq t$, and $\forall j \in \{1, \ldots, n\}$:*

$$meets(\tau_j, \tau_{j+1}) \, (j < n) \tag{1}$$

$$|\tau_j| \geq d(r_j) \tag{2}$$

$$(r_j, r_{j+1}) \in E_R \, (j < n) \tag{3}$$

The first constraint states that the exit time of the j^{th} resource in the plan must be equal to the entry time into resource $j+1$. The second constraint requires that the agent's occupation time of a resource is at least sufficient to traverse the resource in the minimum travel time. The third constraint states that if two resources follow each other in the agent's plan, then they must be adjacent in the resource graph. The cost of an agent's plan is defined as the difference between the end time and the release time.

In sequential route planning, an agent must respect the plans of all the agents that came before it. From the set of existing agent plans, we can infer how many agents will be in each of the resources for each point in time.

DEFINITION 3 (RESOURCE LOAD). *Given a set Π of agent plans and the set of all time points T, the* resource load *λ is a function $\lambda : R \times T \to \mathbb{N}$ that returns the number of agents occupying a resource r at time point $t \in T$:*

$$\lambda(r, t) = |\{\langle r, \tau \rangle \in \pi \,|\, \pi \in \Pi \wedge t \in \tau\}| \tag{4}$$

In our approach the resource load represents the information that agents need to know about the plans of the other, higher-priority agents. This is similar to other reservation-based approaches such as from Silver [15], but subtly different from the approach in Nishi et al. [14], in which agents inspect each others' plans, in order to detect conflicts.

An agent may only make use of a resource in time intervals when the resource load is less than the capacity of the resource. In such a *free time window*, an agent can enter a resource without creating a conflict with any of the existing agent plans.

DEFINITION 4 (FREE TIME WINDOW). *Given a resource-load function λ, a free time window on resource r is a maximal interval $w = [t_1, t_2)$ such that:*

$$\forall t \in w : \lambda(r, t) < c(r) \tag{5}$$

$$(t_2 - t_1) \geq d(r) \tag{6}$$

The above definition states that for an interval to be a free time window, there should not only be sufficient capacity at any moment during that interval (condition 5), but it should also be long enough for an agent to traverse the resource (condition 6). Within a free time window, an agent must enter a resource, traverse it, and exit the resource. Because of the (non-zero) minimum travel time of a resource, an agent cannot enter a resource right at the end of a free time window, and it cannot exit the window at the start of one. We therefore define for every free time window w an *entry window* $\tau_{\text{entry}}(w)$ and an *exit window* $\tau_{\text{exit}}(w)$. The sizes of the entry and exit windows of a free time window $w = [t_1, t_2)$ on resource r are constrained by the minimum travel time of the resource: $\tau_{\text{entry}}(w) = [t_1, t_2 - d(r))$, and $\tau_{\text{exit}}(w) = [t_1 + d(r), t_2)$.

An agent that wants to go from resource r to (adjacent) resource r' should find a free time window for both of these resources. By definition 2 of a route plan, the exit time out of r should be equal to the entry time into r'. Hence, for a free time window w' on r' to be *reachable* from free time window w on r, the *entry* window of w' should overlap with the *exit* window of w.

DEFINITION 5 (FREE TIME WINDOW GRAPH). *The* free time window graph *is a directed graph $G_W = (W, E_W)$, where the vertices $w \in W$ are the set of free time windows, and E_W is the set of edges specifying the* reachability *between free time windows. Given a free time window w on resource r, and a free time window w' on resource r', it holds that $(w, w') \in E_W$ if the following two conditions hold:*

$$(r, r') \in E_R \tag{7}$$

$$\tau_{exit}(w) \cap \tau_{entry}(w') \neq \varnothing \tag{8}$$

2.1 Plan execution assumptions

We make the simplifying assumption that vehicles do not require any acceleration or deceleration. That is, an agent can reach its desired speed instantaneously (whether this is its maximum speed or a standstill). In addition, we assume that an agent can see whether there is another vehicle directly in front of it on the road (or on the next road, in case the vehicle is on an intersection). This means that the problem of avoiding collisions is taken care of in the simulator, allowing us to focus on the problems of route planning and/or intersection management for the rest of this paper.

A final simplifying assumption concerns the start and finish locations of the vehicles. We assume that vehicles only

enter the infrastructure once they are granted permission to enter their start resource (an intersection, in our experiments), and leave the infrastructure as soon as the destination resource has been traversed (also an intersection). This assumption can be realistic in application domains such as airport taxi routing (where planes land and take off) and urban traffic, where vehicles enter and exit a city, but less realistic in, e.g., warehousing domains. The routing problem for vehicles that must occupy a location of the infrastructure at all times is sometimes referred to as *cooperative pathfinding* in the literature, and considerable effort must already be spent in finding feasible routes for all the vehicles, let alone efficient ones (cf. [16, 11]).

3. ROUTE PLANNING ALGORITHM

In classical shortest path planning, if a node v is on the shortest path from node s to node t, then a shortest path to v can always be expanded to a shortest path to t. Figure 1 shows that in prioritized multi-agent route planning, it is not the case that a shortest route to an intermediate resource can always be expanded to the destination: we see a blue agent that wants to go to the rightmost resource, and a black agent that has a plan to travel rightwards at least until the middle intersection. At time 1 (indicated by the numbers inside the vehicles), the blue agent might make a reservation for the leftmost intersection (i.e., slotting in just ahead of the black agent without hindering it), and expand this plan to the middle intersection. From the middle intersection, at time 2, it cannot plan to go right, because that road is momentarily full with vehicles. However, the blue agent must vacate the intersection, because the black agent has a reservation to use it. Hence, the earliest plan to the middle intersection can only be expanded in the upwards direction, which is a detour in space, and possibly time depending on how quickly the grey agents will start moving.

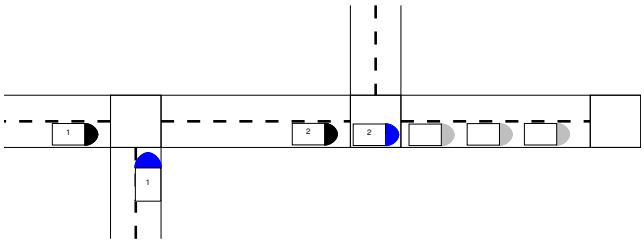


Figure 1: If the blue agent enters the intersection before the black agent, at time 1, then at time 2 it has to drive upwards in order to vacate the intersection for the black agent.

The idea behind our algorithm is that we only need to consider shortest partial plans to the free time windows on a resource: if we have a partial plan that arrives at resource r at time t that lies within free time window w, then all other partial plans to r that arrive at time t', $(t' \geq t) \wedge (t' \in w)$, can be simulated by waiting in resource r from time t to time t'. Waiting is possible because no conflict will ensue as long as the agent exits r before the end of w.

Our route planning algorithm performs a search through the free time window graph that is similar to A*: In each iteration, we remove a partial plan from an open list of partial plans with a lowest value of $f = g + h$, where g is the actual cost of the partial plan, and h is a heuristic estimate of reaching the destination resource. In algorithm 1 below, we will write $\rho(r, t)$ to denote the set of free time windows (directly) reachable from resource r at earliest exit time t.

Algorithm 1 Plan Route

Require: start resource r_1, destination resource r_2, start time t; free time window graph $G_W = (W, E_W)$
Ensure: shortest-time, conflict-free route plan from (r_1, t) to r_2.

```
 1: if ∃w [w ∈ W | t ∈ τ_entry(w) ∧ r_1 = resource(w)] then
 2:     mark(w, open)
 3:     entryTime(w) ← t
 4: while open ≠ ∅ do
 5:     w ← argmin_{w'∈open} f(w')
 6:     mark(w, closed)
 7:     r ← resource(w)
 8:     if r = r_2 then
 9:         return followBackPointers(w)
10:     t_exit ← g(w) = entryTime(w) + d(resource(w))
11:     for all w' ∈ {ρ(r, t_exit) \ closed} do
12:         t_entry ← max(t_exit, start(w'))
13:         if t_entry < entryTime(w') then
14:             backpointer(w') ← w
15:             entryTime(w') ← t_entry
16:             mark(w', open)
17: return null
```

In line 1 of algorithm 1, we check whether there exists a free time window on the start resource r_1 that contains the start time t. If there is such a free time window w, then in line 2 we mark this window as open, and we record the entry time into w as the start time t. In line 5, we select the free time window w on the open list with the lowest value of $f(w)$. As in the original A* algorithm, the function $f(w) = g(w) + h(w)$ is a combination of the actual cost $g(w)$ of the partial plan to w, plus a heuristic estimate $h(w)$ to reach the destination from w. If the resource r associated with w equals the destination resource r_2, then we have found the shortest route to r_2, for the following reason: all other partial plans on open have a higher (or equal) f-value, and if the heuristic is consistent[2], expansion of these partial plans will never lead to a plan with a lower f-value. We return the optimal plan in line 9 by following a series of backpointers.

If r is not the destination, we expand the plan. First, in line 10, we determine the earliest possible exit time out of r as the cost of the partial plan: $g(w) = \text{entryTime}(w) + d(r)$. Then, in line 11, we iterate over all reachable free time windows that are not closed. When expanding free time window w to free time window w', we determine the entry time into w' as the maximum of the earliest exit time out of resource r, and the earliest entry time into w'. We only expand the

[2]Because we make use of a closed list, it is not sufficient to require that the heuristic is merely admissible (i.e., that it would never overestimate the cost of reaching the destination). For a consistent heuristic, it should hold that $h(w) \leq g(w, w') + h(w')$, where $g(w, w')$ is the actual cost of getting from w to w'.

plan from w if there has been no previous expansion to free time window w' with an earlier entry time (initially, we assume that the entry times into free time windows are set to infinity). In line 14, we set the backpointer of the new window w' to the window w from which it was expanded. Then, we record the entry time into w' as t_{entry}, and we mark w' as open. Finally, in case no conflict-free plan exists, we return null in line 17. The worst-case complexity of algorithm 1 is $O(|W|\log(|W|)+|E_W|)$. In case no cyclic plans are allowed, then $|W| \leq (|A|+1)|R|$, and the complexity of algorithm 1 is $O(|\mathcal{A}||R|\log(|\mathcal{A}||R|)+|\mathcal{A}||R|^2)$ (proof in [17]). The worst-case complexity of maintaining the free time window graph G_W is $O(|\mathcal{A}||R|^2)$: for each of at most R reservations of the new plan, one or two new free time windows must be connected to $O(|W| = |\mathcal{A}||R|)$ existing free time windows.

3.1 Plan repair mechanisms

We will now briefly discuss two plan repair mechanisms that can be used to guarantee conflict-free execution for context-aware planners in dynamic environments. The first has been developed by Maza and Castagna [12] and can be considered a baseline approach in the sense that it guarantees conflict-free running without trying to find a repair solution that will result in efficient plan execution. The second is an extension of the first, in which agents can increase their priority over other, delayed agents.

Both plan repair mechanisms rely on the fact that, after all agents have made their plans, it is known for each resource (lane or intersection) in which order the agents are scheduled to visit it. The mechanism of Maza and Castagna is simply to adhere to this *resource priority* (not to be confused with the order in which the agents plan) during plan execution. So, for example, if agent A_1 in figure 2 is delayed, then agent A_2 (and all agents behind it) must wait in case A_1 was the first to enter intersection the middle intersection.

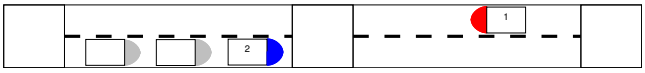


Figure 2: If the red agent A_1 is delayed, then the blue agent A_2 must wait its turn to enter the intersection.

In later work, Maza and Castagna developed a repair mechanism that allowed agents to increase their resource priority over delayed agents in such a way that no new deadlocks were introduced [13]. Note that in our current setting, it is not so obvious why such a change in priorities might lead to a deadlock, but for infrastructures with bi-directional resources, attempting a deadlock-free priority change often involves increasing priority over multiple agents for a whole corridor of resources. The second plan repair mechanism we will employ in this paper is an improvement over the algorithm from Maza and Castagna [13] in the sense that it identifies more deadlock-free priority changes, and also leads to a greater reduction in global delay; see [21].

3.2 Planning shortest paths

We will combine local intersection management policies with agents that follow a shortest path between start and destination locations (for those cases that the intersection does not determine the next road to be taken). In a grid infrastructure, there are many shortest paths (as we assume no cost for turning), so we let each agent construct a random shortest path.

4. INTERSECTION MANAGEMENT

In this section, we will first describe two types of intersection management policies, applied locally at each of the intersections in the infrastructure. The first, most basic type determines which agent is allowed to enter an intersection next, out of the agents ready to enter. The second type of policy then subsequently determines which lane an agent will drive into when it leaves the intersection. Recall from section 3 that a pre-determined path is followed in case only the first type of policy is applied. We now describe the three *intersection entry policies* that we have defined.

DEFINITION 6 (FCFS). *Under* First-Come First-Served *the agent with the earliest* entry request time *may enter first (ties broken arbitrarily); an agent may request entry once it has reached the intersection.*

One should note that an agent cannot request entry when it is waiting behind another agent; only when the agent is first in line can it request entry. The FCFS policy is simple and fair, but it does not take into account congestion formation on the infrastructure.

DEFINITION 7 (LQF). *Under* Longest Queue First*, the agent that forms the head of the longest queue of vehicles waiting to enter, is allowed to enter.*

Longest Queue First (LQF) aims to reduce congestion in the system by reducing the number of vehicles on the fullest of the roads leading into the intersection. In addition to the roads leading into an intersection, another source of vehicles wanting to enter the intersection is formed by those agents that have their starting point at this intersection. However, this set of vehicles is only counted as a queue of length 1; this means that the LQF policy gives precedence to vehicles already on the infrastructure.

DEFINITION 8 (WLQF). *Let t^* be the current time, t_i the time at which agent A_i requested entry to the intersection, and n_i the number of agents on the same road as A_i at time t^*. Under* Weighted Longest Queue First*, the agent that is next to enter is selected according to the formula:*

$$\text{argmax}_{i\in\{1,\ldots,|A|\}}\, n_i + f(t^* - t_i) \qquad (9)$$

for a given function f.

In this paper, we have chosen the function f to divide the argument $t^* - t_i$ by the minimum travel time of the intersection. Hence, when the function f returns a value of, say, 5, then it means that a particular agent has been waiting long enough for (at most) five agents to traverse the intersection since the time it requested entry.

We will now describe an intersection management policy that directs agents to their next lane resource, which we call the Routing Table Approach (RTA), inspired by the way internet routers send packets along their way over the internet. Under RTA, an intersection will select one of at most three outgoing lanes for the next part of the route of an agent, thus not including the direction the agent just came from. When an agent enters an intersection, it announces its destination to the intersection agent, which then computes a value for each of the eligible lanes. Note that the routing table approach only uses information from the lane resources adjacent to the intersection.

DEFINITION 9 (RTA). *Let t^* be the time at which agent A_k, with destination z is ready to leave the intersection, and let $L = \{l_1, \ldots, l_m\}$, $L \subset R$, be the eligible outgoing lane resources, and let $n(l_i)$ denote the number of agents on lane l_i at time t^*. Then the* Routing Table Approach *selects the next lane resource according to the formula:*

$$\text{argmin}_{i \in \{1,\ldots,|L|\}} g(l_i, z) + \alpha\left(\frac{n(l_i)}{\sum_{j=1}^{|L|} n(l_j)}\right) \qquad (10)$$

for some constant α and function g that returns the value of the shortest path between its arguments.

In our experiments, we settled on a value of 5.0 for α; by comparison, the maximum difference, in our setting, between the road with the shortest path and the road with the longest path was 4. This means that if only one outgoing lane has vehicles on it, then this lane will not be chosen.

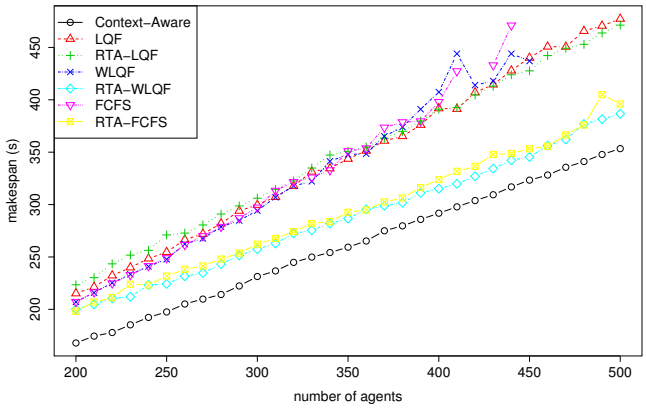
5. EXPERIMENTAL RESULTS

In this section, we describe a set of experiments conducted to compare the performance of context-aware routing to local intersection management strategies. Our principal performance measure is the makespan[3], but we also look at the sum of agent plan costs (where the cost of one agent plan is the time at which it reaches its destination), the distance travelled, and the number of times an intersection management policy will lead the agents into a deadlock.

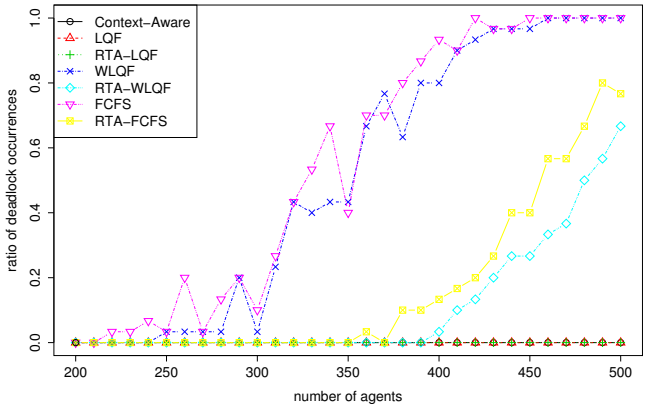
Figure 3 presents the first batch of experiments in which we compared performances for increasing number of agents on a grid infrastructure of five rows and five columns. Each data point in figure 3(a) is the average of 30 runs, or as many as were completed deadlock-free out of those 30 problem instances. The first conclusion we can draw from figure 3 is that context-aware route planning is invariably faster than any of the local intersection management policies. A second conclusion is that the attempt of the routing table approach to reduce congestion (by selecting a next road with congestion in mind) pays off for two out of three intersection *entry* policies; RTA combined with Weighted Longest Queue First seems to be the fastest of the local intersection management policies, although there is not much difference with the basic FCFS entry policy.

[3]All agents have a release time of 0, which means that all agents will either try to obtain a reservation for that time. The makespan is then simply the time at which all agents have reached their destination.

Figure 3(b) shows, however, that RTA-FCFS and RTA-WLQF are not the best from a completeness point of view; from about 350 to 400 agents, an increasing percentages of experiment runs result in a deadlock situation. If only intersection entry management is employed, then from about 300 agents the ability to route agents reduces drastically, in case either First-Come First-Served, or Weighted Longest Queue First is employed. Curiously, when the entry policy Longest Queue First is employed, intersection management has a zero-deadlock rate. This can be explained from figure 5, in which we see a screenshot from the execution of the same instance by RTA-LQF and RTA-WLQF. The main difference is that the latter method, by taking into account the waiting time of a vehicle wanting to enter the infrastructure, will now and then release a new vehicle into the infrastructure even when long queues of vehicles already on the infrastructure have formed at the intersection. The LQF approach, by contrast, will only release a new vehicle when the longest queue of vehicles waiting to enter is at most 1. Hence, using the LQF approach the number of vehicles simultaneously on the infrastructure will be lower, significantly reducing the probability of a deadlock.



(a) makespan



(b) deadlock ratio, with CA, LQF, and RTA-LQF at 0

Figure 3: Performance comparison on (5, 5) grid infrastructure, measured in makespan, and the percentage of deadlock occurrences.

As figure 3 is too cluttered to include confidence intervals, we have plotted the standard deviations for this batch of experiments in figure 4. The spike in the WLQF line (with

the 'x' symbol) is due to the fact that it only managed a handful of deadlock-free runs for 400 or more agents. Overall, we can conclude from figure 4 that context-aware routing is more predictable in its performance than the intersection management policies (RTA-FCFS briefly dips below the Context-Aware line at close to 500 agents, though by that point only around 25% of runs were deadlock free).

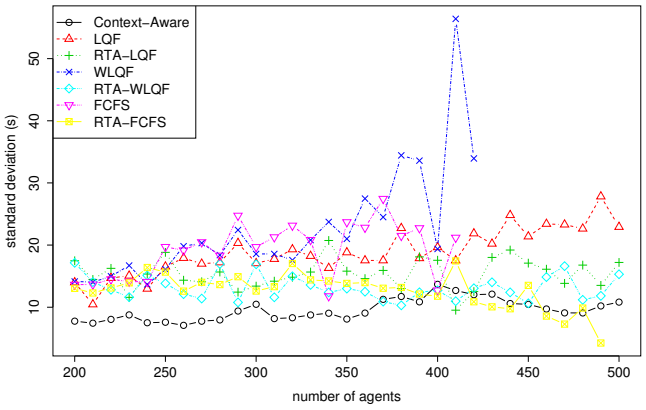


Figure 4: Standard deviations for the experiments of figure 3.

5.1 Cost and distance performance measures

We will now briefly look at the the results of the experiments from different cost perspectives, in figure 6. In figure 6(a) we see the cost per agent divided by the minimum attainable cost (i.e., the cost of traversing the shortest path when no other agents are around), averaged over all agents. This cost measure is a good indicator of the extent agents suffer from the presence of other vehicles, and we see it increases linearly with the number of agents in the system, pretty much regardless of which method is used. Figure 6 shows relative performances that are very similar to those in figure 3 for the makespan measure, although perhaps the best of the intersection-entry-only policies is closer to the best of the RTA policies.

Figure 6(b) shows the distances travelled by each agent (divided by the minimum distance, and averaged over all agents), for each of the methods. For intersection management without RTA, the agents always follow a fixed, and shortest path, so the distance ratio is always 1.0. Agents using context-aware routing have the option of taking a slightly longer route if the shortest one is congested, and this results in routes that are on average 5% longer than the shortest path. The routing table approach has agents travelling the greatest distances, directing agents away from congested areas. If, however, there is no way around the congested area, then it may happen that agents are kept circling in uncongested areas of the infrastructure until the congestion clears.

Another interesting aspect of figure 6(b) is that for RTA-FCFS and RTA-WLQF, the average distance travelled per agent decreases as the number of agents in the system increases. One explanation might be that, as the system becomes heavily congested, the difference between congestion levels on lanes decreases (i.e., if all are very congested). Certainly, if all outgoing lanes are equally congested, then the RTA approach will always select a lane with minimum distance.

5.2 Unexpected incidents

We will now investigate robustness, i.e., the ability of each of the methods to cope with unexpected delays. We will introduce *vehicle incidents* that render vehicles immobile for a fixed period of time. Incidents are generated according to a *rate* parameter, which specifies the average number of incidents per vehicle per time unit. Vehicles can only receive incidents when active, i.e., not before they have entered their start location, and not after they have vacated their destination location (recall the assumption regarding agents and their start and destination locations from section 2.1).

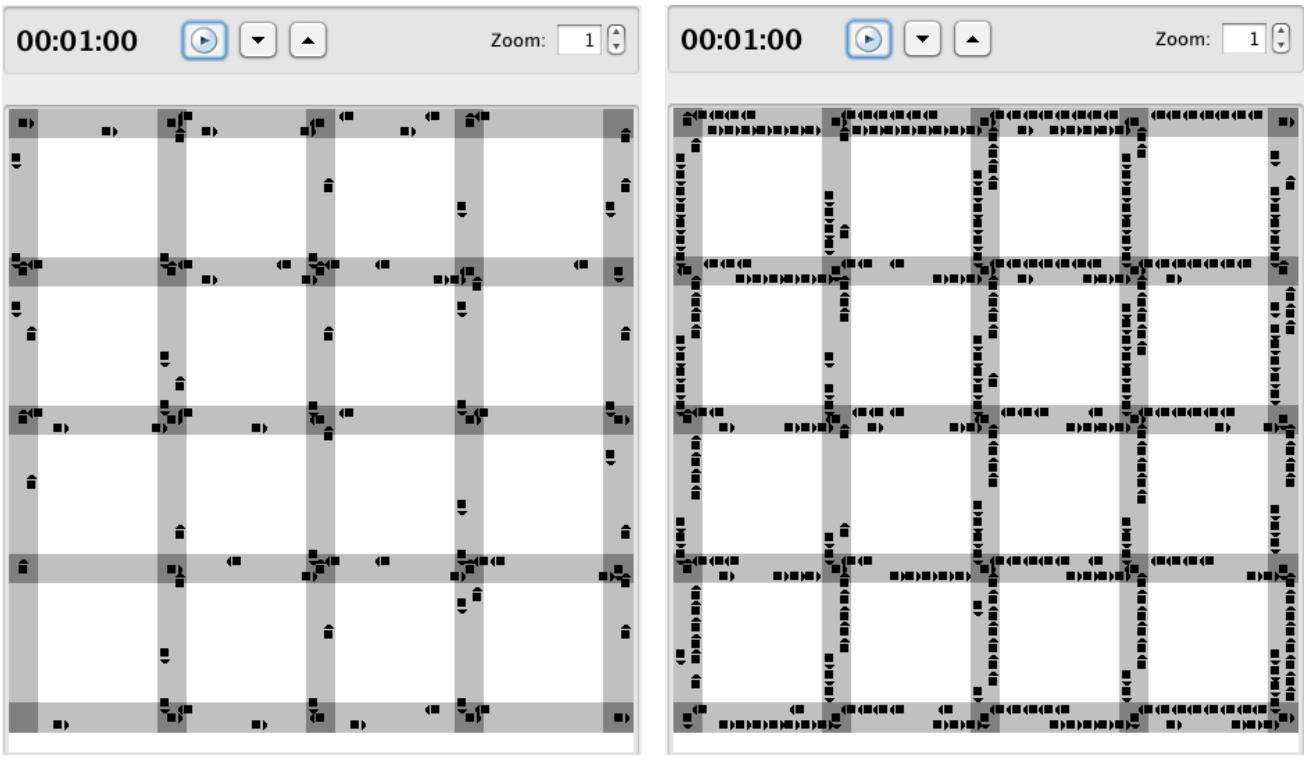
In figure 7, we vary the rate of incidents from 0 to 60 incidents per agent, per hour[4], and we try two different incident durations: 10 seconds per incident in figure 7(a), and 30 seconds for figure 7(b). All incident-experiments were conducted with 400 agents, about the number of agents for which RTA-WLQF is still able to produce a large percentage of deadlock-free runs.

In previous experiments [21, 17, 18], context-aware routing approaches were shown to be fairly robust under incidents of this magnitude, but for these types of infrastructures, standard context-aware quickly loses its advantage, especially for longer incidents. An explanation would be that on this type of grid infrastructure, there is a lot of interaction between the agents on a relatively small number of intersections. This means that if one agent is delayed, many other agents have to wait for it. Increasing the priority with the agent order swap mechanism (CA-AOS in figure 7) restores much of the performance of context-aware routing, although for incidents of longer duration it is now matched by the best intersection management policies. What is also interesting to note from figure 7 is that the local intersection management policies, and in particular LQF, are very robust in the face of vehicle incidents; although figures 7(a) and 7(b) represent different problem instances (i.e., different pairs of start-and-destination locations), it is interesting to see that the makespan barely increases for longer incidents of 30 seconds. Apparently, when one lane of cars is stuck behind a stricken vehicle, an intersection can use that to simply process more vehicles from the remaining lanes.

6. CONCLUSIONS AND FUTURE WORK

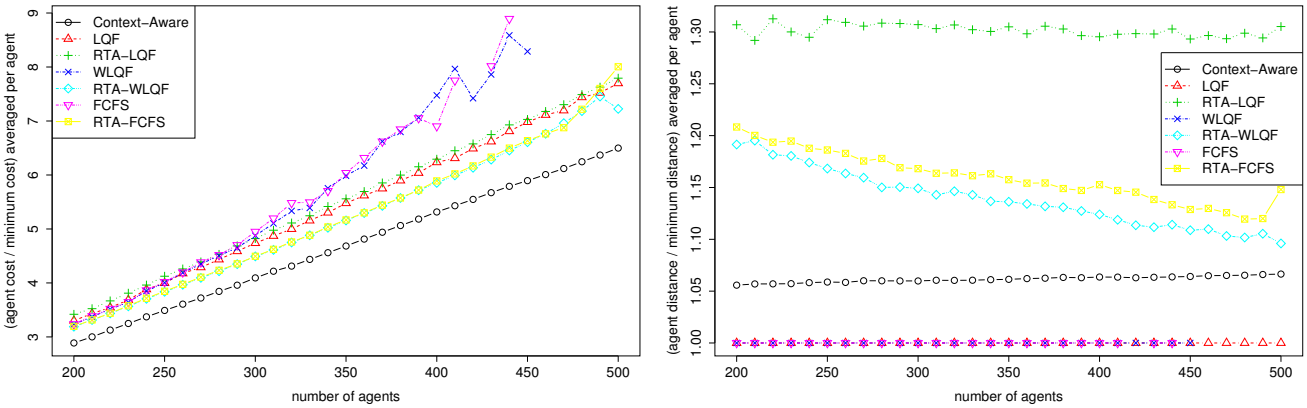
In this paper we compared context-aware routing, in which agents sequentially find locally optimal and conflict-free route plans, with local intersection management, in which an intersection agent decides which vehicle is the next to enter, and possibly directs it along the next lane. Our experiments show that, without any incidents disrupting plan execution, context-aware routing produces the most efficient route plans, when measured in makespan or agent traversal time, while only covering on average 5% more distance than always following the shortest path. Moreover, the most efficient intersection management policies are prone to produce deadlock situations.

[4]To put 60 incidents per agent per hour into perspective, note that the total simulation time equals the makespan, which from figure 7 can be seen to vary from around 200 to 700 seconds.



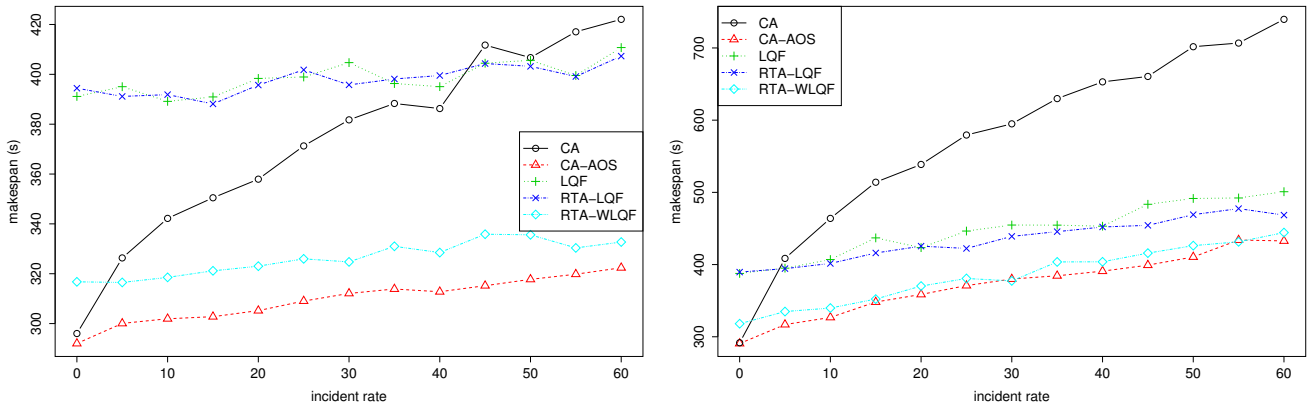
(a) RTA-LQF (b) RTA-WLQF

Figure 5: Execution screenshot of RTA-LQF(a) and RTA-WLQF(b) on the same instance, at one minute into the run.



(a) cost ratio (b) distance ratio

Figure 6: Performance comparison on (5, 5) grid infrastructure, measured in agent cost and distance travelled, divided by a lower bound on cost (and distance), which is the shortest path when other vehicles are not taken into account.



(a) Incident duration = 10s

(b) Incident duration = 30s

Figure 7: Performance comparison on (5, 5) grid infrastructure with 400 agents and makespan performance measure, with unexpected incidents during the execution.

If we do allow unexpected incidents to occur during plan execution, the performance of context-aware routing (with the standard deadlock-prevention mechanism of waiting for delayed vehicles) degrades fairly sharply as many agents end up waiting for one delayed agent — not only directly behind it, but also at an intersection where the delayed agent has failed to appear. The application of the agent order swap mechanism, which increases the priority of timely agents over delayed ones, can return the performance of context-aware routing to a good level. It does mean, however, that context-aware routing needs some kind of plan repair mechanisms in order to be applied in realistic settings. By contrast, the local intersection management policies can be used 'as is' (although there is still the possibility of deadlock, of course).

For future work, there are a number of lines of research we would like to pursue. First of all, we can look into different repair schemes for context-aware routing. The agent order swap mechanism employed in this paper changes the priorities of the agents, but keeps each agent to its originally planned path. Full plan repair, in which an agent computes a completely new route, has been tried in [18] with mixed results. On the one hand, each time an agent successfully makes a new plan it improves its own performance without hindering others (because the new reservations may not conflict with existing ones, adjusted for delays), so full plan repair should be able to improve performance considerably. On the other hand, continual re-planning by all agents has not led to significant global improvement, with agents going back and forth between plans, occasionally covering the same ground multiple times. Hence, a cleverer way of managing the re-planning process is required in order to gain real benefits.

The intersection management policies presented in this paper are of course only a first step in providing intelligent intersection control. One interesting area of possible extensions is to allow limited communication and cooperation between intersection management agents. We could see that the current routing table approach in particular suffered from the limitations of its myopic approach, with agents being directed around congested areas that they had no possibility of avoiding. In addition, when developing policies for intersection management, we must try to find a sensible solution to the possibility of deadlocks. In the Automated Guided Vehicle domain, for instance, a common approach is to model the infrastructure system as a Petri net (cf. [5]), although full deadlock prevention can be computationally expensive in such settings.

7. REFERENCES

[1] Ana L. C. Bazzan. A distributed approach for coordination of traffic signal agents. *Autonomous Agents and Multi-Agent Systems*, 10:131–164, 2005.

[2] Ana L.C. Bazzan, Denise de Oliveira, and Bruno C. da Silva. Learning in groups of traffic signals. *Engineering Applications of Artificial Intelligence*, 23(4):560 – 568, 2010.

[3] Guy Desaulniers, André Langevin, Diane Riopel, and Bryan Villeneuve. Dispatching and conflict-free routing of automated guided vehicles: An exact approach. *International Journal of Flexible Manufacturing Systems*, 15(4):309–331, November 2004.

[4] Kurt Dresner and Peter Stone. A multiagent approach to autonomous intersection management. *Journal of Artificial Intelligence Research*, pages 591–656, 2008.

[5] Maria P. Fanti. A deadlock avoidance strategy for AGV systems modelled by coloured Petri nets. In *Proceedings of the Sixth International Workshop on Discrete Event Systems (WODES'02)*, 2002.

[6] Wolfgang Hatzack and Bernhard Nebel. The operational traffic problem: Computational complexity and solutions. In A. Cesta, editor, *Proceedings of the 6th European Conference on Planning (ECP'01)*, pages 49–60, 2001.

[7] Matthew Hausknecht, Tsz-Chiu Au, and Peter Stone. Autonomous intersection management: Multi-intersection optimization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2011)*, pages 4581–4586, San Francisco, CA, USA, September 25–30 2011. IEEE.

[8] Ying-Chin Ho. A dynamic-zone strategy for

vehicle-collision prevention and load balancing in an AGV system with a single-loop guide path. *Comput. Ind.*, 42(2-3):159–176, 2000.

[9] Chang W. Kim and Jose M.A. Tanchoco. Conflict-free shortest-time bidirectional AGV routeing. *International Journal of Production Research*, 29(1):2377–2391, 1991.

[10] Jung H. Lee, Beom H. Lee, and Myoung Hwan Choi. A real-time traffic control scheme of multiple AGV systems for collision-free minimum time motion: a routing table approach. *IEEE Transactions on Man and Cybernetics, Part A*, 28(3):347–358, May 1998.

[11] Ryan Luna and Kostas E. Bekris. Push and swap: Fast cooperative path-finding with completeness guarantees. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2011.

[12] Samia Maza and Pierre Castagna. Conflict-free AGV routing in bi-directional network. In *Proceedings of the 8^{th} IEEE International Conference on Emerging Technologies and Factory Automation*, volume 2, pages 761–764, Antibes-Juan les Pins, France, October 2001.

[13] Samia Maza and Pierre Castagna. A performance-based structural policy for conflict-free routing of bi-directional automated guided vehicles. *Computers in Industry*, 56(7):719–733, 2005.

[14] Tatsushi Nishi, Masakazu Ando, and Masami Konishi. Experimental studies on a local rescheduling procedure for dynamic routing of autonomous decentralized AGV systems. *Robotics and Computer-Integrated Manufacturing*, 22(2):154–165, April 2006.

[15] David Silver. Cooperative pathfinding. In *Proceedings of the 1st Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2005.

[16] Trevor Standley. Finding optimal solutions to cooperative pathfinding problems. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-10)*, pages 173–178, 2010.

[17] A. W. ter Mors. *The world according to MARP*. PhD thesis, Delft University of Technology, March 2010.

[18] Adriaan W. ter Mors. Conflict-free route planning in dynamic environments. In *Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2166–2171, San Francisco, USA, September 2011.

[19] Adriaan W. ter Mors, Cees Witteveen, Jonne Zutt, and Fernando A. Kuipers. Context-aware route planning. In *Proceedings of the Eighth German Conference on Multi-Agent System Technologies (MATES)*, Lecture Notes in Artificial Intelligence, Leipzig, Germany, September 2010. Springer.

[20] Adriaan W. ter Mors, Jonne Zutt, and Cees Witteveen. Context-aware logistic routing and scheduling. In *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling*, pages 328–335, 2007.

[21] A.W. ter Mors and C. Witteveen. Plan repair in conflict-free routing. In Been-Chian Chien, Tzung-Pei Hong, Shyi-Ming Chen, and Moonis Ali, editors, *Proceedings of the The Twenty Second International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems IEA-AIE 2009*, Lecture Notes in Artificial Intelligence, pages 46–55, Berlin, Heidelberg, June 2009. Springer Verlag LNAI. June 24-27, 2009.

[22] Matteo Vasirani and Sascha Ossowski. A market-inspired approach to reservation-based urban road traffic management. In K. Decker, J.S. Sichman, C. Sierra, and C. Castelfranchi, editors, *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems*, volume I, pages 49–56, Richland, SC, May 2009. IFAAMAS. May 10-15, 2009.

[23] Matteo Vasirani and Sascha Ossowski. A computational market for distributed control of urban road traffic systems. *IEEE Transactions on Intelligent Transportation Systems*, 12(2):313–321, June 2011.

Applying Strategic Multiagent Planning to Real-World Travel Sharing Problems

Jan Hrnčíř
Agent Technology Center
Faculty of Electrical Engineering
Czech Technical University in Prague
121 35 Prague, Czech Republic
jan.hrncir@agents.fel.cvut.cz

Michael Rovatsos
School of Informatics
The University of Edinburgh
Edinburgh EH8 9AB, United Kingdom
mrovatso@inf.ed.ac.uk

ABSTRACT

Travel sharing, i.e., the problem of finding parts of routes which can be shared by several travellers with different points of departure and destinations, is a complex multiagent problem that requires taking into account individual agents' preferences to come up with mutually acceptable joint plans. In this paper, we apply state-of-the-art planning techniques to real-world public transportation data to evaluate the feasibility of multiagent planning techniques in this domain. The potential of improving travel sharing technology has great application value due to its ability to reduce the environmental impact of travelling while providing benefits to travellers at the same time.

We propose a three-phase algorithm that utilises performant single-agent planners to find individual plans in a simplified domain first and then merges them using a best-response planner which ensures resulting solutions are individually rational. Finally, it maps the resulting plan onto the full temporal planning domain to schedule actual journeys.

The evaluation of our algorithm on real-world, multi-modal public transportation data for the United Kingdom shows linear scalability both in the scenario size and in the number of agents, where trade-offs have to be made between total cost improvement, the percentage of feasible timetables identified for journeys, and the prolongation of these journeys. Our system constitutes the first implementation of strategic multiagent planning algorithms in large-scale domains and provides insights into the engineering process of translating general domain-independent multiagent planning algorithms to real-world applications.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence – *Multiagent systems*

General Terms

Algorithms, Design, Experimentation

Keywords

multiagent planning, real-world application, travel sharing

1. INTRODUCTION

Travelling is an important and frequent activity, yet people willing to travel have to face problems with rising fuel prices, carbon footprint and traffic jams. These problems can be ameliorated by *travel sharing*, i.e., groups of people travel together in one vehicle for parts of the journey. Participants in such schemes can benefit from travel sharing in several ways: sharing parts of a journey may reduce cost (e.g., through group tickets), carbon footprint (e.g., when sharing a private car, or through better capacity utilisation of public means of transport), and travellers can enjoy the company of others on a long journey. In more advanced scenarios one could even imagine this being combined with working together while travelling, holding meetings on the road, etc.

Today, there exist various commercial online services for car[1], bike, and walk sharing as well as services which assist users in negotiating shared journeys[2], and, of course, plenty of travel planning services[3] that automate *individual* travel planning for one or several means of transport. On the research side, there is previous work that deals with the ridesharing and car-pooling problem [1, 8, 14]. However, no work has been done that attempts to compute *joint* travel plans based on *public transportation* timetable data and geographical stop locations, let alone in a way that takes into account the *strategic* nature of the problem, which comes about through the different (and potentially conflicting) preferences of individuals who might be able to benefit from travelling together. From the point of view of (multiagent) planning, this presents itself as a very complex application scenario: Firstly, even if one restricted oneself to centralised (non-strategic) planning, the domain is huge – public transportation data for the UK alone currently involves 240,590 timetable connections for trains and coaches (even excluding local city buses), which would have to be translated to a quarter of a million planning actions, at least in a naive formalisation of the domain. Secondly, planning for multiple self-interested agents that are willing to cooperate only if it is beneficial for them is known to be exponentially harder than planning for each agent individually [2]. Yet any automated service that proposes joint journeys would have to guarantee such *strategic* properties in order to be acceptable for human users (who could then even leave it to the service to negotiate trips on their behalf).

[1]E.g., www.liftshare.com or www.citycarclub.co.uk.
[2]E.g., www.companions2travel.co.uk, www.travbuddy.com.
[3]E.g., in the United Kingdom: www.nationalrail.co.uk for trains, www.traveline.info and www.google.com/transit for multi-modal transportation.

In this paper, we present an implementation of best-response planning (BRP) [13] within a three-phase algorithm that is capable of solving strategic travel sharing problems for several agents based on real-world transportation data. Based on a simplified version of the domain that ignores timetabling information, the algorithm first builds individual travel plans using state-of-the-art single-agent planners that are available off the shelf. It then merges these individual plans and computes a multiagent plan that is a Nash equilibrium and guarantees individual rationality of solutions, as well as stability in the sense that no single agent has an incentive to deviate from the joint travel route. This is done using BRP as the underlying planner, as it is the only available planner that can solve strategic multiagent planning problems of such scale, and is proven to converge in domains that comply with certain assumptions, as is the case for our travel sharing domain. In a third and final step, the resulting multiagent plan is mapped onto the full temporal planning domain to schedule actual journeys. This scheduling task is not guaranteed to always find a feasible solution, as the previous simplification ignores a potential lack of suitable connections. However, we show through an extensive empirical evaluation that our method finds useful solutions in a large number of cases despite its theoretical incompleteness.

The contribution of our work is threefold: Firstly, we show that current multiagent planning technology can be used in important planning domains such as travel sharing by presenting its application to a practical problem that cannot be solved with other existing techniques. In the process, we describe the engineering steps that are necessary to deal with the challenges of real-world large-scale data and propose suitable solutions. Secondly, we present an algorithm that combines different techniques in a practically-oriented way, and which is largely based on domain-independent off-the-shelf heuristic problem solvers. In fact, only data preprocessing and timetable mapping use domain-specific knowledge, and much of the process of incorporating this knowledge could be replicated for similar other domains (such as logistics, manufacturing, and network communications). Finally, we provide a potential solution to the hard computational problem of travel sharing that could be exploited for automating important tasks in a future real-world application to the benefits of users, who normally have to plan such routes manually and would be overwhelmed by the choices in a domain full of different transportation options which is inhabited by many potential co-travellers.

We start off by describing the problem domain in section 2 and specifying the planning problem formally in section 3, following the model used in [13]. Section 4 introduces our three-phase algorithm for strategic planning in travel sharing domains and we present an extensive experimental evaluation of the algorithm in section 5. Section 6 presents a discussion of our results and section 7 concludes.

2. THE TRAVEL SHARING DOMAIN

The real-world travel domain used in this paper is based on the public transport network in the United Kingdom, a very large and complex domain which contains 4,055 railway and coach stops supplemented by timetable information. An agent representing a passenger is able to use different means of transport during its journey: walking, trains, and coaches. The aim of each agent is to get from its starting location to its final destination at the lowest possible cost, where the cost of the journey is based on the duration and the price of the journey. Since we assume that all agents are travelling on the same day and that all journeys must be completed within 24 hours, in what follows below we consider only travel data for Tuesdays (this is an arbitrary choice that could be changed without any problem). For the purposes of this paper, we will make the assumption that sharing a part of a journey with other agents is cheaper than travelling alone. While this may not currently hold in many public transportation systems, defining hypothetical cost functions that reflect this would help assess the potential benefit of introducing such pricing schemes.

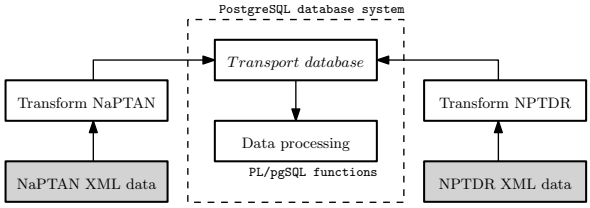


Figure 1: Overview of the data transformation and processing

2.1 Source data

The travel sharing domain uses the National Public Transport Data Repository (NPTDR)[4] which is publicly available from the Department for Transport of the British Government. It contains a snapshot of route and timetable data that has been gathered in the first or second complete week of October since 2004. For the evaluation of the algorithm in section 5, we used data from 2010[5], which is provided in TransXChange XML[6].

National Public Transport Access Nodes (NaPTAN)[7] is a UK national system for uniquely identifying all the points of access to public transport. Every point of access (bus stop, rail station, etc.) is identified by an ATCO code[8], e.g., *9100HAYMRKT* for Haymarket Rail Station in Edinburgh. Each stop in NaPTAN XML data is also supplemented by common name, latitude, longitude, address and other pieces of information. This data also contains information about how the stops are grouped together (e.g., several bus bays that are located at the same bus station).

To be able to use this domain data with modern AI planning systems, it has to be converted to the Planning Domain Definition Language (PDDL). We transformed the data in three subsequent stages, cf. Figure 1. First, we transformed the NPTDR and NaPTAN XML data to a spatially-enabled PostgreSQL database. Second, we automatically processed and optimised the data in the database. The data processing

[4] data.gov.uk/dataset/nptdr
[5] www.nptdr.org.uk/snapshot/2010/nptdr2010txc.zip
[6] An XML-based UK standard for interchange of route and timetable data.
[7] data.gov.uk/dataset/naptan
[8] A unique identifier for all points of access to public transport in the United Kingdom.

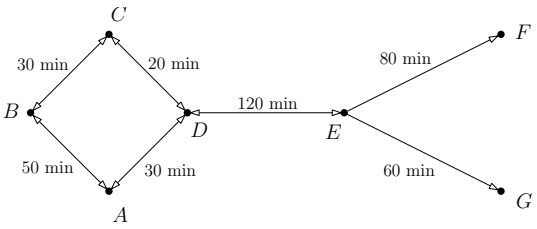


Figure 2: An example of the relaxed domain (e.g., it takes 50 minutes to travel from the stop A to B)

by SQL functions in the procedural PL/pgSQL language included the following steps: merging bus bays at bus stations and parts of train stations, introducing walking connections to enable multi-modal journeys, and eliminating duplicates from the timetable. Finally, we created a script for generating PDDL specifications based on the data in the database. More details about the data processing and PDDL specifications can be found in [11].

2.2 Planning domain definitions

Since the full travel planning domain is too large for any current state-of-the-art planner to deal with, we distinguish the *full domain* from a *relaxed domain*, which we will use to come up with an initial plan before mapping it to the full timetable information in our algorithm below.

The *relaxed domain* is a single-agent planning domain represented as a directed graph where the nodes are the stops and the edges are the connections provided by a service. The graph must be directed because there exist stops that are used in one direction only. There is an edge from A to B if there is at least one connection from A to B in the timetable. The cost of this edge is the minimal time needed for travelling from A to B. A plan P_i found in the relaxed domain for the agent i is a sequence of connections to travel from its origin to its destination. The relaxed domain does not contain any information about the traveller's departure time. This could be problematic in a scenario where people are travelling at different times of day. This issue could be solved by clustering of user requests, cf. chapter 7.

A small example of the relaxed domain is shown in Figure 2. An example plan for an agent travelling from C to F is $P_1 = \langle C \to D, D \to E, E \to F \rangle$. To illustrate the difference between the relaxed domain and the full timetable, there are $8,688$ connections in the relaxed domain for trains and coaches in the UK compared to $240,590$ timetable connections.

Direct trains that do not stop at every stop are filtered out from the relaxed domain for the following reason: Assume that in Figure 2, there is only one agent travelling from C to F and that its plan in the relaxed domain is to use a direct train from C to F. In this case, it is only possible to match its plan to direct train connections from C to F, and not to trains that stop at C, D, E, and F. Therefore, the agent's plan cannot be matched against all possible trains between C and F which is problematic especially in the case where the majority of trains stop at every stop and only a few trains are direct. On the other hand, it is possible to match a plan with a train stopping in every stop to a direct train, as it is explained later in section 4.3.

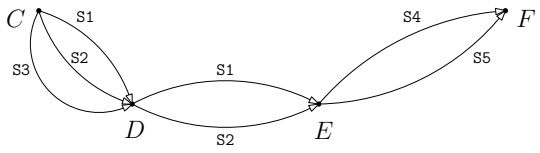


Figure 3: An example of the full domain with stops C, D, E and F for the joint plan $P = \{C \xrightarrow{(1)} D \xrightarrow{(1,2)} E \xrightarrow{(1)} F\}$

The *full domain* is a multiagent planning domain based on the joint plan P. Assume that there are N agents in the full domain (each agent i has the plan P_i from the relaxed domain). Then, the joint plan P is a merge of single-agent plans defined by formula

$$P = \bigcup_{i=1}^{N} P_i$$

where we interpret $\bigcup$ as the union of graphs that would result from interpreting each plan as a set of edges connecting stops. More specifically, given a set of single-agent plans, the plan merging operator $\bigcup$ computes its result in three steps: First, it transforms every single-agent plan P_i to a directed graph G_i where the nodes are the stops from the single-agent plan P_i and the edges represent the actions of P_i (for instance, a plan $P_1 = \langle C \to D, D \to E, E \to F \rangle$ is transformed to a directed graph $G_1 = \{C \to D \to E \to F\}$). Second, it performs a graph union operation over the directed graphs G_i and labels every edge in the joint plan with the numbers of agents that are using the edge (we don't introduce any formal notation for these labels here, and simply slightly abuse the standard notation of sets of edges to describe the resulting graph).

As an example, the joint plan for agent 1 travelling from C to F and sharing a journey from D to E with agent 2 would be computed as

$$\langle C \to D, D \to E, E \to F \rangle \cup \langle D \to E \rangle = \{C \xrightarrow{(1)} D \xrightarrow{(1,2)} E \xrightarrow{(1)} F\}$$

With this, the *full domain* is represented as a directed multigraph where the nodes are the stops that are present in the joint plan of the relaxed domain. Edges of the multigraph are the service journeys from the timetable. Every service is identified by a unique service name and is assigned a departure time from each stop and the duration of its journey between two stops. In the example of the full domain in Figure 3, the agents can travel using some subset of five different services `S1` to `S5`. In order to travel from C to D using service `S1`, an agent must be present at stop C before its departure.

3. THE PLANNING PROBLEM

Automated planning technology [9] has developed a variety of scalable heuristic algorithms for tackling hard planning problems, where plans, i.e., sequences of actions that achieve

a given goal from a given initial state, are calculated by domain-independent problem solvers. To model the travel sharing problem, we use a multiagent planning formalism which is based on MA-STRIPS [2] and coalition-planning games [3]. States are represented by sets of ground fluents, actions are tuples $a = \langle pre(a), eff(a)\rangle$. After the execution of action a, positive fluents p from $eff(a)$ are added to the state and negative fluents $\neg p$ are deleted from the state. Each agent has individual goals and actions with associated costs. There is no extra reward for achieving the goal, the total utility received by an agent is simply the inverse of the cost incurred by the plan executed to achieve the goal.

More formally, following the notation of [13], a multiagent planning problem (MAP) is a tuple

$$\Pi = \langle N, F, I, \{G_i\}_{i=1}^n, \{A_i\}_{i=1}^n, \Psi, \{c_i\}_{i=1}^n\rangle$$

where

- $N = \{1, \ldots, n\}$ is the set of agents,
- F is the set of fluents,
- $I \subseteq F$ is the initial state,
- $G_i \subseteq F$ is agent i's goal,
- A_i is agent i's action set,
- $\Psi : A \rightarrow \{0, 1\}$ is an admissibility function,
- $c_i : \times_{i=1}^n A_i \rightarrow \mathbb{R}$ is the cost function of agent i.

$A = A_1 \times \ldots \times A_n$ is the joint action set assuming a concurrent, synchronous execution model, and $G = \wedge_i G_i$ is the conjunction of all agents' individual goals. A MAP typically imposes concurrency constraints regarding actions that cannot or have to be performed concurrently by different agents to succeed which the authors of [13] encode using an admissibility function Ψ, with $\Psi(a) = 1$ if the joint action a is executable, and $\Psi(a) = 0$ otherwise.

A *plan* $\pi = \langle a^1, \ldots, a^k\rangle$ is a sequence of joint actions $a^j \in A$ such that a^1 is applicable in the initial state I (i.e., $pre(a^1) \subseteq I$), and a^j is applicable following the application of $a^1, \ldots, a^{j-1}$. We say that π *solves* the MAP Π if the goal state G is satisfied following the application of all actions in π in sequence. The cost of a plan π to agent i is given by $C_i(\pi) = \sum_{j=1}^k c_i(a^j)$. Each agent's contribution to a plan π is denoted by π_i (a sequence of $a_i \in A_i$).

3.1 Best-response planning

The *best-response planning* (BRP) algorithm proposed in [13] is an algorithm which, given a solution π^k to a MAP Π, finds a solution π^{k+1} to a *transformed planning problem* Π_i with minimum cost $C_i(\pi^{k+1})$ among all possible solutions:

$$\pi^{k+1} = \arg\min\{C_i(\pi) | \pi \text{ identical to } \pi^k \text{ for all } j \neq i\}$$

The transformed planning problem Π_i is obtained by rewriting the original problem Π so that all other agents' actions are fixed, and agent i can only choose its own actions in such a way that all other agents still can perform their original actions. Since Π_i is a single-agent planning problem, any cost-optimal planner can be used as a best-response planner.

In [13], the authors show how for a class of congestion planning problems, where all fluents are *private*, the transformation they propose allows the algorithm to converge to a Nash equilibrium if agents iteratively perform best-response steps using an optimal planner. This requires that every agent can perform its actions without requiring another agent, and hence can achieve its goal in principle on its own, and conversely, that no agent can invalidate other agents' plans. Assuming infinite capacity of vehicles, the relaxed domain is an instance of a congestion planning problem[9].

The BRP planner works in two phases: In the first phase, an initial plan for each agent is computed (e.g., each agent plans independently or a centralised multi-agent planner is used). In the second phase, the planner solves simpler best-response planning problems from the point of view of each individual agent. The goal of the planner in a BRP problem is to minimise the cost of an agent's plan without changing the plans of others. Consequently, it optimises a plan of each agent with respect to the current joint plan.

This approach has several advantages. It supports full concurrency of actions and the BRP phase avoids the exponential blowup in the action space resulting in much improved scalability. For the class of potential games [16], it guarantees to converge to a Nash equilibrium. On the other hand, it does not guarantee the optimality of a solution, i.e., the quality of the equilibrium in terms of overall efficiency is not guaranteed (it depends on which initial plan the agents start off with). However, experiments have proven that it can be successfully used for improving general multiagent plans [13]. Such non-strategic plans can be computed using a *centralised multiagent planner*, i.e., a single-agent planner (for instance Metric-FF [10]) which tries to optimise the value of the joint cost function (in our case the sum of the values of the cost functions of agents in the environment) while trying to achieve all agents' goals. Centralised multiagent planners have no notion of self-interested agents, i.e., they ignore the individual preferences of agents.

4. A THREE-PHASE STRATEGIC TRAVEL SHARING ALGORITHM

The main problem when planning for multiple agents with a centralised multiagent planner is the exponential blowup in the action space which is caused by using concurrent, independent actions [13]. Using a naive PDDL translation has proven that a direct application of a centralised multiagent planner to this problem does not scale well. For example, a simple scenario with two agents, ferries to Orkney Islands and trains in the area between Edinburgh and Aberdeen resulted in a one-day computation time.

As mentioned above, we tackle the complexity of the domain by breaking down the planning process into different phases

[9] Following the definition of a congestion planning problem in [13], all actions are private, as every agent can use transportation means on their own and the other agents' concurrently taken actions only affect action cost. A part of the cost function defined in section 4.4 depends only on the action choice of individual agent.

Input

- a relaxed domain
- a set of N agents $A = \{a_1, \ldots, a_N\}$
- an origin and a destination for each agent

1. The initial phase

For $i = 1, \ldots, N$ **do**

Find an initial journey for agent a_i using a single-agent planner.

2. The BR phase

Do until no change in the cost of the joint plan

For $i = 1, \ldots, N$ **do**

1. Create a simpler best-response planning (BRP) problem from the point of view of agent a_i.
2. Minimise the cost of a_i's plan without changing the plans of others.

End

3. The timetabling phase

Identify independent groups of agents $G = \{g_1, \ldots, g_M\}$.

For $i = 1, \ldots, M$ **do**

1. Find the relevant timetable for group g_i.
2. Match the joint plan of g_i to timetable using a temporal single-agent planner in the full domain with the relevant timetable.

End

Figure 4: Three-phase algorithm for finding shared journeys for agents

that avoid dealing with the full fine-grained timetable data from the outset. Our algorithm, which is shown in Figure 4, is designed to work in three phases.

4.1 The initial phase

First, in the *initial phase*, an initial journey is found for each agent using the relaxed domain. A journey for each agent is calculated independently of other agents in the scenario using a single-agent planner. As a result, each agent is assigned a single-agent plan which will be further optimised in the next phase. This approach makes sense in our domain because the agents do not need each other to achieve their goals and they cannot invalidate each other's plans.

4.2 The BR phase

Second, in the *BR phase* (best-response phase), which is also based on the relaxed domain, the algorithm uses the BRP algorithm as described above. It iteratively creates and solves simpler best-response planning problems from the point of view of each individual agent. In the case of the relaxed domain, the BRP problem looks almost the same as a problem of finding a single-agent initial journey. The difference is that the cost of travelling is smaller when an agent uses

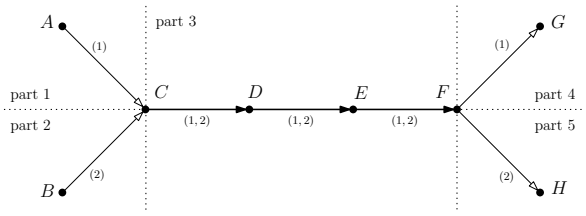


Figure 5: Parts of the group journey of two agents

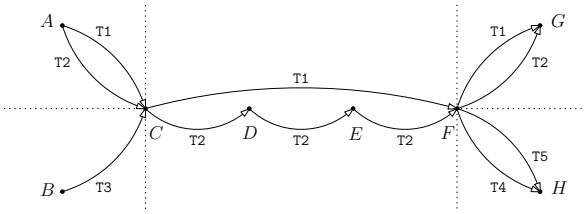


Figure 6: The full domain with services from the relevant timetable. There are five different trains T1 to T5, and train T1 is a direct train.

a connection which is used by one or more other agents, as will be explained below, cf. equation (1).

Iterations over agents continue until there is no change in the cost of the joint plan between two successive iterations. This means that the joint plan cannot be further improved using the best-response approach. The output of the BR phase is the joint plan P in the relaxed domain (defined in section 2.2) that specifies which connections the agents use for their journeys and which segments of their journeys are shared. The joint plan P will be matched to the timetable in the final phase of the algorithm.

4.3 The timetabling phase

In the final *timetabling phase*, the optimised shared journeys are matched against timetables using a temporal single-agent planner which assumes the full domain. For this, as a first step, independent groups of agents with respect to journey sharing are identified. An independent group of agents is defined as an edge disjoint subgraph of the joint plan P. This means that actions of independent groups do not affect each other so it is possible to find a timetable for each independent group separately.

Then, for every independent group, *parts* of the group journey are identified. A *part* of the group journey is defined as a maximal continuous segment of the group journey which is performed by the same set of agents. As an example, there is a group of two agents that share a segment of their journeys in Figure 5: Agent 1 travels from A to G while agent 2 travels from B to H. Their group journey has five parts, with the shared part (part 3) of their journey occurring between stops C and F.

In order to use both direct and stopping trains when the group journey is matched to the timetable, the *relevant timetable* for a group journey is composed in the following way: for every part of the group journey, return all timetable services in the direction of agents' journeys which connect

Table 1: Parameters of the testing scenarios

Scenario code	S1	S2	S3	S4	S5
Number of stops	344	721	1 044	1 670	2 176
Relaxed domain connections	744	1 520	2 275	4 001	4 794
Timetabled connections	23 994	26 702	68 597	72 937	203 590
Means of transport	trains	trains, coaches	trains	trains, coaches	trains

the stops in that part. An example of the relevant timetable for a group of agents from the previous example is shown in Figure 6. Now, the agents can travel using the direct train `T1` or using train `T2` with intermediate stops.

The relevant timetable for the group journey is used with the aim to cut down the amount of data that will be given to a temporal single-agent planner. For instance, there are 23 994 timetabled connections in Scotland. For an example journey of two agents, there are only 885 services in the relevant timetable which is approximately 4 % of the data. As a result, the temporal single-agent planner gets only the necessary amount of data as input, to prevent the time-consuming exploration of irrelevant regions of the state space.

4.4 Cost functions

The timetable data used in this paper (cf. section 2.1) contains neither information about ticket prices nor distances between adjacent stops, only durations of journeys from one stop to another. This significantly restricts the design of cost functions used for the planning problems. Therefore, the cost functions used in the three phases of the algorithm are based solely on the duration of journeys.

In the initial phase, every agent tries to get to its destination in the shortest possible time. The cost of travelling between adjacent stops A and B is simply the duration of the journey between stops A and B. In the BR phase, we design the cost function in such a way that it favours shared journeys. The cost $c_{i,n}$ for agent i travelling from A to B in a group of n agents is then defined by equation (1):

$$c_{i,n} = \left(\frac{1}{n}\,0.8 + 0.2\right) c_i \tag{1}$$

where c_i is the individual cost of the single action to i when travelling alone. In our experiments below, we take this to be equal to the duration of the trip from A to B.

This is designed to approximately model the discount for the passengers if they buy a group ticket: The more agents travel together, the cheaper the shared (leg of a) journey becomes for each agent. Also, an agent cannot travel any cheaper than 20 % of the single-agent cost. In reality, pricing for group tickets could vary, and while our experimental results assume this specific setup, the actual price calculation could be easily replaced by any alternative model.

In the timetabling phase, every agent in a group of agents tries to spend the shortest possible time on its journey. When matching the plan to the timetable, the temporal planner tries to minimise the sum of durations of agents' journeys including waiting times between services.

5. EVALUATION

We have evaluated our algorithm on public transportation data for the United Kingdom, using various off-the-shelf planners for the three phases described above, and a number of different scenarios. These are described together with the results obtained from extensive experiments below.

5.1 Planners

All three single-agent planners used for the evaluation were taken from recent International Planning Competitions (IPC) from 2008 and 2011. We use LAMA [18] in the initial and the BR phase, a sequential *satisficing* (as opposed to cost-optimal) planner which searches for any plan that solves a given problem and does not guarantee optimality of the plans computed. LAMA is a propositional planning system based on heuristic state-space search. Its core feature is the usage of landmarks [17], i.e., propositions that must be true in every solution of a planning problem.

SGPlan$_6$ [12] and POPF2 [7] are temporal satisficing planners used in the timetabling phase. Such temporal planners take the duration of actions into account and try to minimise makespan (i.e., total duration) of a plan but do not guarantee optimality. The two planners use different search strategies and usually produce different results. This allows us to run them in sequence on every problem and to pick the plan with the shortest duration. It is not strictly necessary to run both planners, one could save computation effort by trusting one of them.

SGPlan$_6$ consists of three inter-related steps: parallel decomposition, constraint resolution and subproblem solution [4, 10, 15, 19]. POPF2 is a temporal forward-chaining partial-order planner with a specific extended grounded search strategy described in [5, 6]. It is not known beforehand which of the two planners will return a better plan on a particular problem instance.

5.2 Scenarios

To test the performance of our algorithm, we generated five different scenarios of increasing complexity, whose parameters are shown in Table 1. They are based on different regions of the United Kingdom (Scotland for S1 and S2, central UK for S3 and S4, central and southern UK for S5). Each scenario assumes trains or trains and coaches as available means of transportation.

In order to observe the behaviour of the algorithm with different numbers of agents, we ran our algorithm on every scenario with $2, 4, 6, \ldots, 14$ agents in it. To ensure a reasonable likelihood of travel sharing to occur, all agents in the scenarios travel in the same direction. This imitates a preprocessing step where the agents' origins and destinations are clustered according to their direction of travel. For sim-

plicity reasons, we have chosen directions based on cardinal points (N–S, S–N, W–E, E–W). For every scenario and number of agents, we generated 40 different experiments (10 experiments for each direction of travel), resulting in a total of 1,400 experiments. All experiments are generated partially randomly as defined below.

To explain how each experiment is set up, assume we have selected a scenario from S1 to S5, a specific number of agents, and a direction of travel, say north–south. To compute the origin–destination pairs to be used by the agents, we place two axes x and y over the region, dividing the stops in the scenario into four quadrants **I**, **II**, **III** and **IV**. Then, the set O of possible origin–destination pairs is computed as

$$O := \{(A,B) | ((A \in \mathbf{I} \wedge B \in \mathbf{IV}) \vee (A \in \mathbf{II} \wedge B \in \mathbf{III})) \wedge |AB| \in [20, 160]\}$$

This means that each agent travels from A to B either from quadrant **I** to **IV** or from quadrant **II** to **III**. The straight-line distance $|AB|$ between the origin and the destination is taken from the interval 20–160 km (when using roads or rail tracks, this interval stretches approximately to a real distance of 30–250 km). This interval is chosen to prevent journeys that could be hard to complete within 24 hours. We sample the actual origin-destination pairs from the elements of O, assuming a uniform distribution, and repeat the process for all other directions of travel.

5.3 Experimental results

We evaluate the performance of the algorithm in terms of three different metrics: the amount of time the algorithm needs to compute shared journeys for all agents in a given scenario, the success rate of finding a plan for any given agent and the quality of the plans computed. Unless stated otherwise, the values in graphs are averaged over 40 experiments that were performed for each scenario and each number of agents. The results obtained are based on running the algorithm on a Linux desktop computer with 2.66 GHz Intel Core 2 Duo processor and 4 GB of memory. The data, source codes and scenarios in PDDL are archived online[10].

5.3.1 Scalability

To assess the scalability of the algorithm, we measure the amount of time needed to plan shared journeys for all agents in a scenario.

In many of the experiments, the SGPlan$_6$ and POPF2 used in the timetabling phase returned some plans in the first few minutes but then they continued exploration of the search space without returning any better plan. To account for this, we imposed a time limit for each planner in the temporal planning stage to 5 minutes for a group of up to 5 agents, 10 minutes for a group of up to 10 agents, and 15 minutes otherwise.

Figure 7 shows the computation times of the algorithm. The graph indicates that overall computation time grows roughly linearly with increasing number of agents, which confirms that the algorithm avoids the exponential blowup in the action space characteristic for centralised multiagent planning.

[10] agents.fel.cvut.cz/download/hrncir/journey_sharing.tgz

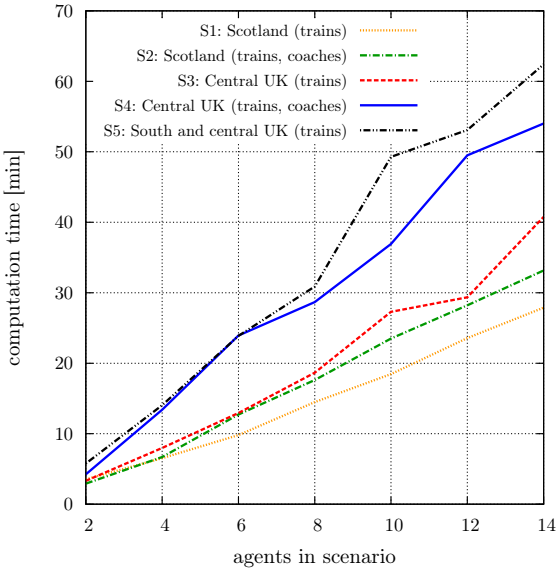


Figure 7: Computation time against number of agents

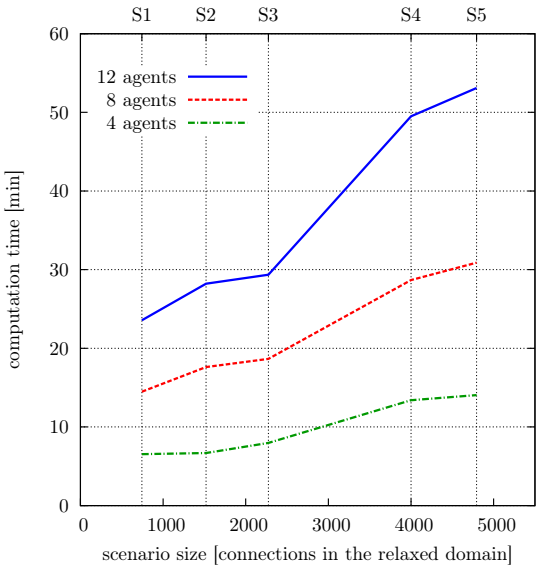


Figure 8: Computation time against scenario size

Computation time also increases linearly with growing scenario size. Figure 8 shows computation times for 4, 8 and 12 agents against the different scenarios.

While the overall computation times are considerable (up to one hour for 14 agents in the largest scenario), we should emphasise that the algorithm is effectively computing equilibrium solutions in multi-player games with hundreds of thousands of states. Considering this, the linear growth hints at having achieved a level of scalability based on the structure of the domain that is far above naive approaches to plan jointly in such state spaces. Moreover, it implies that the runtimes could be easily reduced by using more processing power.

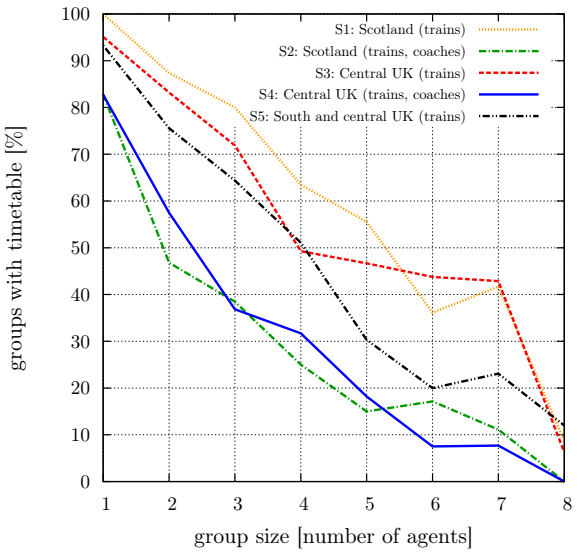


Figure 9: Percentage of groups for which a timetable was found as a function of group size.

5.3.2 *Success rate*

To assess the value of the algorithm, we also need to look at how many agents end up having a valid travel plan. Planning in the relaxed domain in the initial and the BR phase of the algorithm is very successful. After the BR phase, 99.4 % of agents have a journey plan. The remaining 0.6 % of all agents does not have a single-agent plan because of the irregularities in the relaxed domain caused by splitting the public transportation network into regions. The agents without a single-agent plan are not matched to timetable connections in the timetabling phase.

The timetabling phase is of course much more problematic. Figure 9 shows the percentage of groups for which a timetable was found, as a function of group size. In order to create this graph, number of groups with assigned timetable and total number of groups identified was counted for every size of the group. There are several things to point out here.

Naturally, the bigger a group is, the harder it is to find a feasible timetable, as the problem quickly becomes overconstrained in terms of travel times and actually available transportation services. When a group of agents sharing parts of their journeys is big (5 or more agents), the percentage of groups for which we can find a timetable drops below 50 %. With a group of 8 agents, almost no timetable can be found. Basically what happens here is that the initial and BR phases find suitable ways of travelling together in principle, but that it becomes impossible to find appropriate connections that satisfy every traveller's requirements, or do not add up to a total duration of less than 24 hours.

We can also observe that the success rate is higher in scenarios that use only trains than in those that combine trains and coaches. On closer inspection, we can observe that this is mainly caused by different *service densities* in the rail and coach networks, i.e., the ratios of timetabled connections

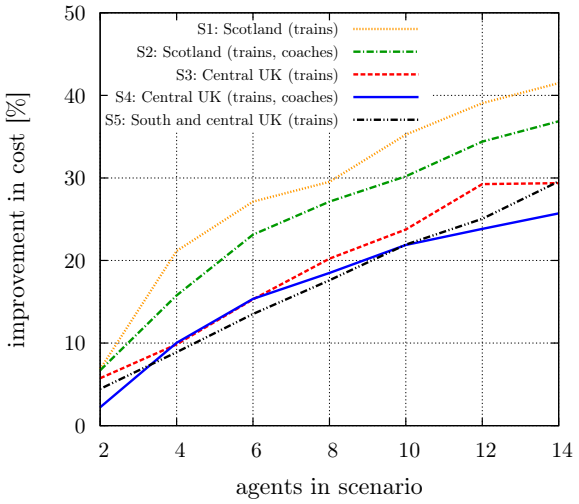


Figure 10: Average cost improvement

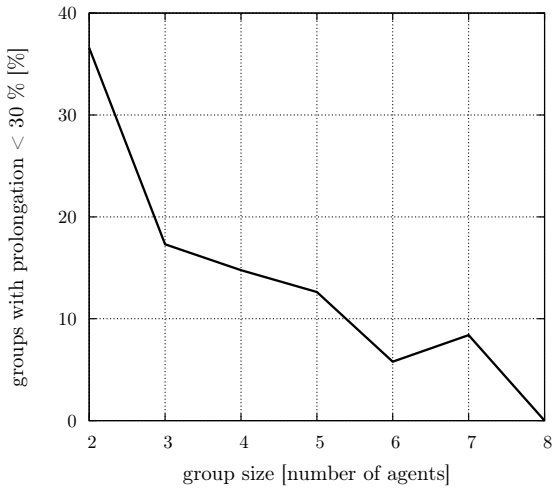


Figure 11: Percentage of groups with less than 30 % journey prolongation

over connections in the relaxed domain. For example, the service density is 33 train services a day compared to only 4 coach services in Scotland. As a consequence, it is much harder to find a timetable in a scenario with both trains and coaches because the timetable of coaches is much less regular than the timetable of trains. However, this does not mean that there is less sharing if coaches are included. Instead, it just reflects the fact that due to low service density, many of the envisioned shared journeys do not turn out to be feasible using coaches. The fact that this cannot be anticipated in the initial and BR phases is a weakness of our method, and is discussed further in section 7.

5.3.3 *Plan quality*

Finally, we want to assess the quality of the plans obtained with respect to improvement in cost of agents' journeys and their prolongation, to evaluate the net benefit of using our method in the travel sharing domain. We should mention that the algorithm does not explicitly optimises the solutions with respect to these metrics. To calculate cost im-

provement, recalling that $C_i(\pi) = \sum_j c_i(a^j)$ for a plan is the cost of a plan $\pi = \langle a^1, \ldots, a^k \rangle$ to agent i, assume $n(a^j)$ returns the number of agents with whom the jth step of the plan is shared. We can define a cost of a shared travel plan $C_i'(\pi) = \sum_j c_{i,n(a^j)}(a^j)$ using equation (1). With this we can calculate the improvement ΔC as follows:

$$\Delta C = \frac{\sum_{i \in N} C_i(\pi_i) - \sum_{i \in N} C_i'(\pi_N)}{\sum_{i \in N} C_i(\pi_i)} \quad (2)$$

where N is the set of all agents, π_i is the single-agent plan initially computed for agent i, and π_N is the final joint plan of all agents after completion of the algorithm (which is interpreted as the plan of the "grand coalition" N and reflects how subgroups within N share parts of the individual journeys).

The average cost improvement obtained in our experiments is shown in Figure 10, and it shows that the more agents there are in the scenario, the higher the improvement. However, there is a trade-off between the improvement in cost and the percentage of groups that we manage to find a suitable timetable for, cf. Figure 9.

On the one hand, travel sharing is beneficial in terms of cost. On the other hand, a shared journey has a longer duration than a single-agent journey in most cases. In order to evaluate this trade-off, we also measure the journey prolongation. Assume that $T_\mathrm{i}(\pi)$ is the total duration of a plan to agent i in plan π, and, as above, π_i/π_N denote the initial single-agent plans and the shared joint plan at the end of the timetabling phase, respectively. Then, the prolongation ΔT of a journey is defined as follows:

$$\Delta T = \frac{\sum_{i \in N} T_i(\pi_N) - \sum_{i \in N} T_i(\pi_i)}{\sum_{i \in N} T_i(\pi_i)} \quad (3)$$

Journey prolongation can be calculated only when a group is assigned a timetable and each member of the group is assigned a single-agent timetable. For this purpose, in every experiment, we also calculate single-agent timetables in the timetabling phase of the algorithm.

A graph of the percentage of groups that have a timetable with prolongation less than 30 % as a function of group size is shown in Figure 11. The graph shows which groups benefit from travel sharing, i.e., groups whose journeys are not prolonged excessively by travelling together. Approximately 15 % of groups with 3–4 agents are assigned a timetable that leads to a prolongation of less than 30 %. Such a low percentage of groups can be explained by the algorithm trying to optimise the price of the journey by sharing in the BR phase. However, there is a trade-off between the price and the duration of the journey. The more agents are sharing a journey, the longer the journey duration is likely to be.

These results were obtained based on the specific cost function (1) we have introduced to favour travel sharing, and which would have to be adapted to the specific cost structure that is present in a given transportation system. Also, the extent to which longer journey times are acceptable for the traveller depends on their preferences, but these could be easily adapted by using different cost functions.

6. DISCUSSION

The computation of single-agent plans in the initial phase involves solving a set of completely independent planning problems. This means that the planning process could be speeded up significantly by using parallel computation on multiple CPUs. The same is true for matching different independent groups of agents to timetabled connections in the timetabling phase. As an example, assume that there are N agents in the scenario and $t_1, \ldots, t_N$ are the computation times for respective single-agent initial plans. If computed concurrently, this would reduce the computation time from $t = \sum_{i=1}^{N} t_i$ to $t' = \max_{i=1}^{N}(t_i)$. Similar optimisations could be performed for the timetabling phase of the algorithm. In the experiments with 10 agents, for example, this would lead to a runtime reduced by 48 % in scenario S1 and by 44 % in scenario S5.

A major problem of our method is the inability to find appropriate connections in the timetabling phase for larger groups. There are several reasons for this. Firstly, the relaxed domain is overly simplified, and many journeys found in it do not correspond to journeys that would be found if we were planning in the full domain. Secondly, there are too many temporal constraints in bigger groups (5 or more agents), so the timetable matching problem becomes unsolvable given the 24-hour timetable. However, it should also be pointed out that such larger groups would be very hard to identify and schedule even using human planning. Thirdly, some parts of public transportation network have very irregular timetables.

Our method clearly improves the cost of agents' journeys by sharing parts of the journeys, even though there is a trade-off between the amount of improvement, the percentage of found timetables and the prolongation of journeys. On the one hand, the bigger the group, the better the improvement. On the other hand, the more agents share a journey, the harder it is to match their joint plan to timetable. Also, the prolongation is likely to be higher with more agents travelling together, and will most likely lead to results that are not acceptable for users in larger groups.

Regarding the domain-independence of the algorithm, we should point out that its initial and BR phases are completely domain-independent so they could easily be used in other problem domains such as logistics, network routing or service allocation. In the traffic domain, the algorithm can be used to plan routes that avoid traffic jams or to control traffic lights. What is more, additional constraints such as staying at one city for some time or travelling together with a specific person can be easily added. On the other hand, the timetabling phase of the algorithm is domain-specific, providing an example of the specific design choices that have to be made from an engineering point of view.

To assess the practical value of our contribution, it is worth discussing how it could be used in practice as a part of a travel planning system for real passengers. In such a system, every user would submit origin, destination and travel times. Different users could submit their preferences at different times, with the system continuously computing shared journeys for them based on information about all users' preferences. Users would need to agree on a shared journey in

time to arrange meeting points and to purchase tickets, subject to any restrictions on advance tickets etc. Because of this lead time, it would be entirely sufficient if the users got an e-mail with a planned journey one hour after the last member of the travel group submits his or her journey details, which implies that even with our current implementation of the algorithm, the runtimes would be acceptable.

From our experimental evaluation, we conclude that reasonable group sizes range from two to four persons. Apart from the fact that such groups can be relatively easily coordinated, with the price model used in this paper, cf. formula (1), every member of a three-person group could save up to 53 % of the single-agent price. The success rate of the timetabling phase of the algorithm for three-person groups in the scenario S3 (trains in the central UK) is 70 %.

7. CONCLUSION

We have presented a multiagent planning algorithm which is able to plan meaningful shared routes in a real-world travel domain. The algorithm has been implemented and evaluated on five scenarios based on real-world UK public transport data. The algorithm exhibits very good scalability, since it scales linearly both with the scenario size and the number of agents. The average computation time for 12 agents in the scenario with 90 % of trains in the UK is less than one hour. Experiments indicate that the algorithm avoids the exponential blowup in the action space characteristic for a centralised multiagent planner.

To deal with thousands of users that could be in a real-world travel planning system, a preprocessing step would be needed: The agents would have to be divided into smaller groups by clustering them according to departure time, direction of travel, origin, destination, length of journey and preferences (e.g., travel by train only, find cheapest journey). Then, the algorithm could be used to find a shared travel plan with a timetable. To prevent too large groups of agents which are unlikely to be matched to the timetable, a limit can be imposed on the size of the group. If a group plan cannot be mapped to a timetable, the group can be split into smaller sub-groups which are more likely to identify a suitable timetable.

Finally, the price of travel and flexibility of travel sharing can be significantly improved by sharing a private car. In the future, we would like to explore the problem of planning shared journeys when public transport is combined with ride sharing. Then, in order to have a feasible number of nodes in the travel domain, train and bus stops can be used as meeting points where it is possible to change from a car to public transport or vice versa.

8. ACKNOWLEDGMENTS

Partly supported by the Ministry of Education, Youth and Sports of Czech Republic (grant No. LD12044) and European Commission FP7 (grant agreement No. 289067).

9. REFERENCES

[1] S. Abdel-Naby, S. Fante, and P. Giorgini. Auctions Negotiation for Mobile Rideshare Service. In *Procs. ICPCA 2007*, pages 225–230, 2007.

[2] R. I. Brafman and C. Domshlak. From One to Many: Planning for Loosely Coupled Multi-Agent Systems. In *Procs. ICAPS 2008*, pages 28–35. AAAI Press, 2008.

[3] R. I. Brafman, C. Domshlak, Y. Engel, and M. Tennenholtz. Planning Games. In *Procs. IJCAI 2009*, pages 73–78, July 2009.

[4] Y. Chen, B. W. Wah, and C.-W. Hsu. Temporal planning using subgoal partitioning and resolution in SGPlan. *Journal of Artificial Intelligence Research*, 26:323–369, Aug. 2006.

[5] A. J. Coles, A. I. Coles, A. Clark, and S. T. Gilmore. Cost-Sensitive Concurrent Planning under Duration Uncertainty for Service Level Agreements. In *Procs. ICAPS 2011*, pages 34–41. AAAI Press, June 2011.

[6] A. J. Coles, A. I. Coles, M. Fox, and D. Long. Forward-Chaining Partial-Order Planning. In *Procs. ICAPS 2010*, pages 42–49. AAAI Press, May 2010.

[7] A. J. Coles, A. I. Coles, M. Fox, and D. Long. POPF2: a Forward-Chaining Partial Order Planner. In *Procs. IPC-7*, 2011.

[8] E. Ferrari, R. Manzini, A. Pareschi, A. Persona, and A. Regattieri. The car pooling problem: Heuristic algorithms based on savings functions. *Journal of Advanced Transportation*, 37(3):243–272, 2003.

[9] M. Ghallab, D. Nau, and P. Traverso. *Automated Planning: Theory and Practice*. Morgan Kaufmann Publishers Inc., 2004.

[10] J. Hoffmann and B. Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.

[11] J. Hrnčíř. Improving a Collaborative Travel Planning Application. Master's thesis, The University of Edinburgh, Aug. 2011.

[12] C.-W. Hsu and B. W. Wah. The SGPlan Planning System in IPC-6. In *Procs. IPC-6*, 2008.

[13] A. Jonsson and M. Rovatsos. Scaling Up Multiagent Planning: A Best-Response Approach. In *Procs. ICAPS 2011*, pages 114–121. AAAI Press, June 2011.

[14] P. Lalos, A. Korres, C. K. Datsikas, G. S. Tombras, and K. Peppas. A Framework for Dynamic Car and Taxi Pools with the Use of Positioning Systems. In *Computation World: Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns*, pages 385–391, Nov. 2009.

[15] N. Meuleau, M. Hauskrecht, K.-E. Kim, L. Peshkin, L. P. Kaelbling, T. Dean, and C. Boutilier. Solving very large weakly coupled Markov decision processes. In *Procs. AAAI 1998*, pages 165–172. AAAI Press, 1998.

[16] D. Monderer and L. S. Shapley. Potential Games. *Games and Economic Behavior*, 14(1):124–143, 1996.

[17] S. Richter, M. Helmert, and M. Westphal. Landmarks Revisited. In *Procs. AAAI 2008*, pages 975–982. AAAI Press, July 2008.

[18] S. Richter and M. Westphal. The LAMA planner. Using landmark counting in heuristic search. In *Procs. IPC-6*, 2008.

[19] B. W. Wah and Y. Chen. Constraint partitioning in penalty formulations for solving temporal planning problems. *Artificial Intelligence*, 170:187–231, Mar. 2006.

Algorithm for Vehicle Routing Problem with Time Windows Based on Agent Negotiation

Petr Kalina
Department of Cybernetics
Faculty of Electrical Engineering
Czech Technical University in Prague
peta.kalina@gmail.com

Jiří Vokřínek
Agent Technology Center
Faculty of Electrical Engineering
Czech Technical University in Prague
jiri.vokrinek@fel.cvut.cz

ABSTRACT

We suggest an efficient algorithm for the vehicle routing problem with time windows (VRPTW) based on agent negotiation. The algorithm is based on a set of generic negotiation methods and state-of-the-art insertion heuristics. Experimental results on well known Solomon's and Homberger-Gehring benchmarks demonstrate that the algorithm outperforms previous agent based algorithms. The relevance of the algorithm with respect to the state-of-the-art centralized solvers is discussed within a comprehensive performance and algorithmic analysis, that has not been provided by previous works. The main contribution of this work is the assessment of general applicability of agent based approaches to routing problems in general providing for a solid base for future research in this area.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Intelligent agents, Multiagent systems*

General Terms

Algorithms, Measurement, Performance, Experimentation

Keywords

Vehicle routing problem with time windows, Multi-agent problem solving, Agent negotiation

1. INTRODUCTION

The vehicle routing problem with time windows (VRPTW) is one of the most important and widely studied problems in the the transportation domain. For a comprehensive literature review refer e.g. to surveys presented by [1, 2]. The VRPTW is a problem of finding a set of routes from a single depot to serve customers at geographically scattered locations. Each customer is visited by exactly one route with each route starting and ending at the depot. For each route the sum of demands of the customers served by the route must not exceed the capacity of the vehicle serving the route (capacity constraint). Also, the service at each customer must begin within a given time interval (time window constraints). The primary objective of the VRPTW is to find the minimum number of routes servicing all customers. Usually a secondary optimization objective is to minimize the total distance traveled. The primary objective corresponds to solving the underlying multiple bin-packing problem while the secondary objective corresponds to a variant of the multiple traveling salesman problem — both solved in a state space constrained by the time windows. Traditionally the VRPTW (together with the closely related pickup and delivery problem with time windows - PDPTW) is a problem area dominated by centralized solvers e.g.[9, 11].

Real world applications of routing algorithms are often very complex with highly dynamic, heterogenous and potentially non-cooperative or privacy conscious environments having to be captured and processed, being part of the higher level transactions e.g. general supply chain management processes etc. The multi-agent systems are an emerging choice for modeling systems with attributes similar to those mentioned above. An interesting survey on real-world applicability of agent based approaches in the transportation domain is presented in [13].

The aim of this paper is not, however, to stress the real-world applicability of presented algorithm. On the other hand, we present a thorough assessment of the agent based algorithm in terms of overall performance in an effort to establish its position among the state-of-the-art algorithms.

2. RELATED WORK

As already mentioned, a thorough survey of VRPTW algorithms is presented by [1, 2]. Thus we only refer to the two currently leading state-of-the-art algorithms.

In [9] the authors present an algorithm based on the *ejection pools* principle. The algorithm is based on performing very good unfeasible insertions of customers to individual routes, followed by an ejection procedure in which the feasibility is recovered by ejecting some other customers from the unfeasible routes. The algorithm equals the best known cumulative number of vehicles (CVN) of 405 on the Solomon's instances with new best known cumulative travel time (CRT) of 57233.

An improved algorithm presented in [11] further employs a specific local search strategy guiding the ejections. Also, a feasible insertion mechanism denoted as *squeeze* as well as a search diversification *perturb* procedure are employed throughout the solving process boosting the algorithm's convergence. The algorithm provides for the contemporary best known CVN of 10290 over the whole extended Homberger-Gehring benchmark set.

A number of approaches have been suggested for solving the VRPTW and routing problems in general by means of multi-agent negotiation profiting from well known multi-agent based approaches to general task allocation problems [16]. On simple VRP good results have been reported by an

agent based solver presented by [15]. In general, however, there has been very few works trying to rigorously establish the position of agent negotiation in a field dominated by centralized solvers, with most contributions focusing on the real-world applicability rather than outright performance.

An agent based algorithm for VRPTW is presented in [5], built around the concepts of a Shipping Company and underlying Shipping Company Truck. The planning is done dynamically and is based on the well known contract net protocol (CNP) accompanied by a "simulated trading" improvement strategy based on finding the optimal customer exchanges by solving a maximal pairing problem on a graph representing the proposed exchanges. No relevant performance assessment is provided and the algorithm is found to be sensitive to the ordering of routed tasks.

The algorithm for PDPTW presented by [7] is essentially a parallel insertion procedure based on CNP with subsequent improvement phase consisting of reallocating some randomly chosen tasks from each route. Used cost structure is based on the well known Solomon's I1 insertion heuristic [14]. The performance is assessed on an ad-hoc dataset.

The algorithm for VRPTW presented by [8] is based on agents representing individual customers, individual routes and a central planner agent. A sequential insertion procedure based on Solomon's I1 heuristic is followed by an improvement phase in which the agents propose moves gathered in a "move pool" with the most advantageous move being selected and performed. Additionally, a route elimination routine is periodically invoked — which is not well described in the text. Experimental assessment is based on Solomon's instances [14] with a CVN of 436 and CRT of 59281. No runtime information is provided.

In [4] the authors propose a VRPTW algorithm based on Order agent — Scheduling agent — Vehicle agent hierarchy. The algorithm is based on a modified CNP insertion procedure limiting the negotiation to agents whose routes are in proximity of the task being allocated in an effort to minimize the number of negotiations. Again no relevant performance information is provided.

3. NOTATION

Let $\{1..N\}$ represent the set of customers with the depot denoted as 0. Let a sequence of customers $\langle c_0, c_1, ..c_m, c_{m+1}\rangle$ denote a route served by a single vehicle with c_0 and c_{m+1} corresponding to the depot. For each customer c_i on the route let (e_i, l_i, s_i, d_i) denote the earliest/latest service start times (the *time window*), service time and demand at the customer respectively. For simplicity, we will use the term *task* to denote a customer and all accompanying service information. Let D denote the vehicle capacity and let $t_{i,j}$ correspond to the travel time between customers c_i and c_j (in an Euclidian space). We use the term *partial solution* to denote a solution with some unserved customers.

Given a route $\langle c_0, c_1, ..c_m, c_{m+1}\rangle$ let (E_i, L_i) correspond to the *earliest* and *latest possible service start* at customer c_i computed recursively according to:

$$\begin{aligned} E_1 &= max\,(e_1, t_{0,1}) \\ E_i &= max\,(e_i, E_{i-1} + s_{i-1} + t_{i-1,i}) \end{aligned} \tag{1}$$

and

$$\begin{aligned} L_m &= l_m \\ L_i &= min\,(l_i, L_{i+1} - t_{i,i+1} - s_i) \end{aligned} \tag{2}$$

As shown in [3], the time window constraints are satisfied when $E_i \leq L_i$ for all $i \in 1..m$. The capacity constraint is satisfied when $\sum_1^m d_i \leq D$.

4. ALGORITHM BASED ON AGENT NEGOTIATION

As mentioned above, a relevant rigorous assessment of the key properties of the respective agent-based algorithms e.g. runtime, convergence, etc. has not been provided by neither of the previous studies. Thus the main contribution of this work is: (i) the establishment of a general framework for agent based approaches based on the state-of-the-art knowledge, (ii) assessment of its algorithmic properties on known widely used benchmark sets using a performance conscious prototype implementation and (iii) the discussion of important areas for future research in an effort to provide a sound alternative to traditional solvers.

The presented algorithm is similar in its approach to the generalizad algorithmic framework for task allocation based problem solving described in [16, 15]. Thus the presented framework is generic and can adopt a number of approaches as far as the actual negotiation/allocation process is concerned. It provides a base for formalizing agent negotiation based solving approaches to routing problems in general.

A three layer basic architecture features a top layer represented by a Task Agent, middle layer represented by an Allocation Agent and a fleet of Vehicle Agents present at the bottom level of the architecture.

Task Agent acts as an interface between the algorithm's computational core and the surrounding infrastructure. It is responsible for registering the tasks and submitting them to the underlying Allocation Agent.

Allocation Agent instruments the actual solving process by negotiating with the Vehicle Agents. The negotiation is conducted based upon task commitment and decommitment cost estimates provided by the Vehicle Agents.

Vehicle Agent represents an individual vehicle serving a route. It provides the Allocation Agent with the above mentioned inputs. These are computed based on local (private) Vehicle Agent's plan processing.

Fiugure 1 illustrates the allocation algorithm process implemented by the Allocation Agent. The process is started given a partial solution σ and a set of unallocated tasks T. For the dynamic problem variant the set T corresponds to a one-element set with the actually processed task and σ represents the partial solution σ at the time of allocation. In static case the set T corresponds to an ordered set of all instance tasks, while σ is a partial solution representing a set of empty routes.

In essence, the allocation process consists of a series of *negotiation* interactions between the Allocation Agent and the Vehicle Agents serving the routes within the partial solution σ. In various places of the algorithm the Allocation Agent may require the Vehicle Agents to: (i) estimate the cost of

`Input:` Ordered set of tasks T, Partial solution σ
`Output:` Solution σ after task allocation

Procedure $allocate(T, \sigma)$
begin
1: Init reallocate counters $r[t] := 0$ for all $t \in T$;
2: **while** (exists$(t \in T)$, $r[t] \leq reallocationLimit$)
3: $dynamicImprove(\sigma)$;
4: Select first $t \in \{t \in T, r[t]$ minimal$\}$;
5: $I := \{v \in Ins^{feas}(\sigma, t), costCommit(t, v)$ is minimal$\}$;
6: **if** $(I \neq \emptyset)$ **then**
7: Randomly select $v \in I$;
8: $commit(t, v)$;
9: remove t from T;
10: **else**
11: $r[t] := r[t] + 1$;
12: **endif**
13: **endwhile**
14: $finalImprove1(\sigma, T)$;
15: $finalImprove2(\sigma, T)$;
16: ..
17: **return** σ;
end

Figure 1: The Allocation Agent main algorithm.

committing to a given task, (ii) estimate the gain resulting from dropping some commitment, (iii) identify the most costly task within their respective routes or (iv) commit to or decommit from a given task.

The interactions with the Vehicle Agents are represented by the $costCommit(t, v)$ and $commit(t, v)$ functions (lines 5 and 8) corresponding to the cost estimate of agent v committing to task t and the actual commitment. From the Allocation Agent's point of view these are Vehicle Agent's private operations. Thus they may reflect various aspects and constraints the vehicles need to consider (e.g. loading constraints, vehicle actual position, vehicle shift times, etc.), potentially reflecting the heterogeneity of the real-world problem being solved. Similarly, various semantics of the actual commitments may be introduced (e.g. revocable/irrevocable etc.). The other interactions mentioned above are carried out within the improvement methods (lines 3, 14 and 15) using the corresponding $gainDecommit(t, v)$, $worstCommitment(v)$ and $decommit(t, v)$ methods and will be described later in the text.

The process begins with resetting the reallocate counters (line 1) and runs in a loop that is terminated when the limit on unsuccessful allocation retries has been reached for all unallocated tasks or until no such tasks exist (line 2). In the former case the allocation has not been successful. Depending on the real world problem semantics the Task Agent may instantiate (dispatch/require) another vehicle and restart the process using either an empty solution or reusing the partial solution returned by the previous run.

The $dynamicImprove(\sigma)$ function (line 2) corresponds to a particular *dynamic improvement method* being executed iteratively throughout the allocation process. Later in the text we describe a set of applicable methods e.g. *ε-ReallocateWorst* or *ReallocateAll*. At this stage the application of a specific improvement method may enhance the partial solution σ and therefore: (i) potentially increase the chance of success for the latter stages of the allocation process and (ii) possibly modify the solution in a way that enables allocation of tasks that were not successfully allocated in previous attempts. As the individual routes get denser, the space for changing the solution decreases. Thus the ability to improve the solution in the early stages of the allocation process is an important feature of the algorithm.

The tasks with the lowest number of retries are processed first in the order in which they are encountered in the set T (line 4). An auction in which each of the Vehicle Agents provides a commitment cost estimate for the currently processed task is carried out on behalf of the Allocation Agent. Thus the agents that can feasibly undertake the task (the set Ins^{feas}) with the best commitment costs are identified (line 5). In a distributed environment the auction process is carried out using the CNP protocol. A randomly chosen agent from this set then commits to the task (lines 7 and 8) and the task is marked as allocated (line 9). In case no agent can feasibly undertake the task (line 6), the reallocate counter for the task is incremented (line 11).

The $finalImprove1, 2(\sigma, T)$ (lines 14, 15) correspond to the final improvement strategies being applied. Just like the $dynamicImprove(\sigma)$ function these correspond to a certain improvement method being applied here. However, the method used for final improvement may differ from the one used for the dynamic improvement. For example it may be advantageous to employ a route length conscious improvement method at the end of the allocation process to address the secondary optimization criteria of the problem. The difference in signature between the two functions illustrates the fact that throughout the final improvement we may still try to allocate the unallocated tasks.

Thus a particular algorithm is instantiated by supplying the actual fleet of Vehicle Agents with respective cost estimation functions and specifying the individual improvement methods for the $dynamicImprove$ and $finalImprove1, 2$ functions.

For the static variant of the VRPTW discussed within this work, the primary optimization criteria is addressed by running the allocation process with an initial solution σ corresponding to an appropriately small fleet of homogenous empty vehicles represented by Vehicle Agents. In case the process fails, a new Vehicle Agent is instantiated and added to the fleet and the process is restarted.

Within the next sections we present two different variants of VRPTW Vehicle Agent implementations based on the state-of-the-art insertion heuristics and three improvement methods for the static VRPTW problem variant, as well a theoretically sound setting for the initial size of the fleet. Several ways in which the set of tasks T can be ordered are discussed as well.

4.1 Insertion Heuristics

The two Vehicle Agent implementations presented within this study are based on the well known *cheapest insertion* principle. Let c_j be the customer associated with the task t, let $\langle c_0, c_1, ..c_m, c_{m+1} \rangle$ be the corresponding route of the agent v. Let $costIns(t, v, i)$ represent the cost estimate of inserting t between the customers c_{i-1} and c_i. The cost estimate for agent v committing to t is thus given by

$$costCommit(t, v) = \underset{i \in fi(1..m)}{\text{argmin}} \ (costIns(t, v, i)) \qquad (3)$$

where $fi(1..m)$ represents the set of all feasible insertion points on the route.

Given an insertion index i, let (E_j, L_j) represent the *earliest possible* and *latest possible service start* at c_j when inserted at index i. The E_j and L_j values can be computed according to Equations 1 and 2 as

$$E_j = max\left(e_j, E_{i-1} + s_{i-1} + t_{i-1,j}\right) \tag{4}$$

and

$$L_j = min\left(l_i, L_i - t_{j,i} - s_j\right). \tag{5}$$

The insertion is feasible when both the time window constraint $E_j \leq L_j$ and the capacity constraint $\left(\sum_1^m d_i\right) + d_j \leq D$ are satisfied. By storing agent's cumulative demand $D^c = \sum_1^m d_i$ alongside the agent's plan the capacity constraint can be checked trivially by verifying that $D^c + d_j \leq D$.

Given the identified best insertion index i, the actual commitment of the agent v to the task t requires the $E_k, k = i..m$ and $L_l, l = 1..i-1$ values to be updated according to Equations 1 and 2 as well as the agent's cumulative demand D^c.

As mentioned above, we evaluated two respective implementations of the Vehicle Agent's interface functions based on two well known insertion heuristics.

4.1.1 Travel Time Savings Heuristic

The *travel time savings* heuristic is notoriously known to the routing community. Using the same example as in the previous section, the insertion cost corresponds to

$$costIns^{TT}(t, v, i) = t_{i-1,j} + t_{j,i} - t_{i-1,i}. \tag{6}$$

Let c_k denote a customer corresponding to a task t' already within v's plan. The decommitment gain is computed accordingly as

$$gainDecommit^{TT}(t', v) = t_{k-1,k} + t_{k,k+1} - t_{k-1,k+1}. \tag{7}$$

The travel time savings heuristic leverages the spatial aspects of the problem, with a cost structure corresponding to the impacts of agent commitments or decommitments on the travel time of the agents. It has been shown [14, 10], however, that an insertion heuristic exploiting the temporal relations of the tasks given by their respective time windows can yield significantly better results.

4.1.2 Slackness Savings Heuristic

The *slackness savings heuristic* thus introduces elements to the cost structure based on the interactions between individual time windows constraints caused by their respective widths and placements within the agent's route. It is a simplified adaptation of PDPTW heuristic presented by [10] for the VRPTW problem.

Given c_k corresponding to a customer on agent v's route from previous examples, let $sl_k = L_k - E_k$ represent the *slack time* at customer c_k. An insertion of c_j requires the L_k and E_k values to be updated along the route possibly reducing the corresponding sl_k values. Reductions to slack times correspond to the constraining effects an insertion of a customer imposes on the rest of the agent's route. Let $sl'_j = L_j - E_j$ represent the slack time at the inserted customer c_j after the insertion. We denote $sl_j = l_j - e_j$ the slack time at c_j *prior* to the insertion. Given $sl'_k = L'_k - E'_k, k = 1..m$ being the updated slack times after the insertion, the overall reduction in route slackness is given by

$$SLR(t, v, i) = \left(\sum_1^m \left(sl_k - sl'_k\right)\right) + \left(sl'_j - sl_j\right). \tag{8}$$

The i variable in function's signature corresponds to the fact the $sl'_k, k \in 1..m$ and sl'_j are particular for the insertion index i.

The $costIns^{SL}(t, v, i)$ for the slackness savings heuristic is based on both the spatial and the temporal aspects of the insertion with

$$costIns^{SL}(t, v, i) = \alpha.SLR(t, v, i) + \beta.costIns^{TT}(t, v, i). \tag{9}$$

where α and β, $\alpha + \beta = 1$ correspond to the respective weights of the two criteria being considered.

The $removalGain(t', v)$ is computed using an analogous approach as

$$\begin{aligned} removalGain^{SL}(t, v) = \alpha.SLI(t, v) \\ +\beta.removalGain^{TT}(t, v). \end{aligned} \tag{10}$$

where $SLI(v, t)$ corresponds to a slack time increase resulting from updated sl'_i values as a result of removing customer c_j with the updated slack time at customer c_j given by $sl'_j = l_j - e_j$.

4.2 Improvement Methods

Within the *dynamicImprove* and $finalImprove1, 2$ functions each Vehicle Agent decommits from some of its tasks, based on the particular negotiation method being applied. Each decommitment is followed by an auction process in which an agent with the lowest $costCommit(t, v)$ commitment cost estimate commits to the task. Thus the task can be reinserted to exactly the same position within the same agent's plan, to a different position within the agent's plan or a different agent can commit to the task. We refer to a decommitment and subsequent task reinsertion as the *reallocation* of a single task t. In the first above mentioned case we consider the reallocation unsuccessful as the solution was not changed. Following three negotiation methods were considered:

- ***ReallocateAll***: For each Vehicle Agent all of its tasks are reallocated. The tasks are processed in the order in which they appear in respective agents' routes. The agents are processed in the order in which they are added to the partial solution σ being processed.
- ***ε-ReallocateRandom***: For each Vehicle Agent a portion of its tasks corresponding to $\varepsilon \in (0, 1)$ is reallocated. For $\varepsilon = 1$ this corresponds to the previous method but for an individual agent the tasks are processed in a random order. The order in which the agents are processed is the same as above.
- ***ε-ReallocateWorst***: Each agent drops commitments to a portion of its tasks corresponding to $\varepsilon \in (0, 1)$. The tasks processed in decreasing order based on their respective decommitment gain estimates. The ordering is achieved via the *worstCommitment* agent interface function. For presented implementation this consists of the agent privately invoking the *gainDecommit* function for each task within its route and returning the most costly one.

The interactions between the Allocation Agent and the Vehicle Agents are carried out using the Vehicle Agent interface functions: *costCommit*, *commit*, *gainDecommit*, *decommit* and *worstCommitment*. As already mentioned, these are Vehicle Agent's private operations potentially reflecting the real world problem semantics.

4.3 Algorithm Initial Settings

The remaining settings necessary for instantiating a particular algorithm for the static VRPTW case are: (i) the count of empty Vehicle Agents in the initial partial solution σ and (ii) the ordering of the set of tasks T being passed to the allocation process.

4.3.1 Initial Vehicles Count

As already mentioned, the static VRPTW case primary optimization criteria is addressed by instantiating a fleet of empty vehicles and restarting the allocation process with increased number of vehicles in case of failure until a feasible solution is found.

A sound setting for the initial vehicles count should correspond to the lower bound number of vehicles for the problem instance being solved. Such a number can be computed based on the mutual incompatibilities of the tasks given their respective time windows and travel times. Two tasks are incompatible if they cannot be served by a single vehicle, that is when it is impossible to start the service at either of the customers at the earliest possible moment and reach the other customer within the corresponding time window. The minimal number of vehicles necessary for solving the instance is bound by the size of the maximal set of mutually incompatible tasks, providing for the time windows based lower bound on the number of vehicles. Also, the cumulative demand of all customers has to be lower than the cumulative capacity of all the vehicles combined providing for a capacity based lower bound on the number of vehicles.

Within this work thus the vehicles count in the initial partial solution σ is set to the bigger of the two above mentioned lower bounds. The size of the maximal set of mutually incompatible tasks is estimated using a graph based algorithm approximately solving a maximal clique problem on a graph with edges corresponding to the mutually incompatible tasks. The algorithm is described in [10].

4.3.2 Initial Task Ordering

Within the $allocate(T, \sigma)$ function the task to be processed next is chosen based upon the ordering of the set T. We considered the following settings:

- **Most Demand First** (MDF): Tasks are ordered decreasingly by the volume of their demands, according to the well known *most-constrained-first* greedy allocation principle applied to the underlying multiple bin packing problem.
- **Tightest Time Window First** (TTF): Tasks are ordered increasingly by the duration of their time windows following the *most-constrained-first* approach, this time based on the time windows of the individual tasks.
- **Earliest First** (EF): Tasks are ordered increasingly by the beginning time of their time window. This setting causes naturally competing tasks to be allocated in close succession.

During our experimental assessment of the presented algorithm we found that none of the orderings is dominant. To the contrary each of the orderings performed well on different subset of benchmarked instances. Thus finding a fitting task ordering for a particular problem instance is an interesting problem in its own right, however, it is outside the scope of this study. Instead, in an effort to appropriately illustrate the limits of the presented algorithm, we treated the *set of orderings* to be used as an additional parameter of the algorithm, running the $allocate(T, \sigma)$ function for each of the orderings and choosing the best result from these runs.

4.4 Complexity Analysis

Given N is number of tasks for a given problem instance and assuming the number of reallocation retries is constant, the asymptotic complexity of the allocation process corresponds to

$$O^{clique} + O^{ordering} + N \times \left(O^{dyn} + O^{alloc}\right) + O^{fin1,2} \quad (11)$$

where O^{clique} is the complexity of estimating the initial Vehicle Agents count, $O^{ordering}$ is the complexity of the initial ordering of the set of tasks T prior to the allocation process, O^{dyn} is the complexity of the *dynamicImprove* function, O^{alloc} corresponds to the complexity of the auction process of finding the agent with the minimal $costCommit(v, t)$. The $O^{fin1,2}$ corresponds to the complexity of the corresponding $finalImprove1, 2$ functions.

The complexity of O^{alloc} is inherent to the heuristic being used. Given m is the number of tasks within agent v's route, the complexity of the $costCommit^{TT}(t, v)$ function is $O(m)$. The $costCommit^{SL}(t, v)$ requires the updated slack times to be computed for each insertion point thus resulting in a complexity of $O(m^2)$. As the total number of tasks in agents' routes is bound by N, the worst case complexity of the O^{alloc} is $O(N)$ for the travel time savings heuristic and $O(N^2)$ for the slackness savings heuristic. Similarly the $decommitGain(t, v)$ function complexity depends on the insertion heuristic being used with $decommitGain^{TT}(t, v)$ having an O(1) complexity while $decommitGain^{SL}(t, v)$ being $O(N)$ in the worst case. The subsequent $commit(t, v)$ results in $O(N)$ worst case complexity due to the need to update the E_i and L_i values stored alongside the winning agent's route having N tasks in the worst case. For the same reasons the complexity of the $decommit(t, v)$ function is $O(N)$ as well. The improvement functions correspond to the application of a particular improvement method. With $\varepsilon = 1$ all of the methods consist of reallocating all tasks within the partial solution σ under construction. A single task reallocation consists of a $decommit(t, v)$ function being invoked, followed by an auction process and the subsequent invocation of $commit(t, v)$. Thus the complexity of an improvement strategy being applied is given by $O^{imp} = N \times \left(O^{decommit} + O^{alloc}\right)$ resulting in $O(N^2)$ and $O(N^3)$ worst case complexities for the two presented heuristic. Within the *ε-ReallocateWorst* method, the task with the maximal $decommitGain(t, v)$ value has to be identified prior to each reallocation. The complexity of such an operation is identical to the corresponding O^{alloc} however, so it does not affect the above mentioned conclusion. Thus $O^{fin1,2}$ and O^{dyn} correspond to the complexity of O^{imp} for respective heuristics.

As the $O^{clique} = O(N^3)$ [10] and $O^{ordering} = O(N\ log(N))$ the overall worst case complexity of presented algorithm is

Table 1: Performance of presented algorithm compared to best known results

Type	Best	Agents	*Algorithm-B*	*Algorithm-FI*	*Algorithm-DI*	*Algorithm-DIA*
All	10695, 5049252	–	+1110, +3375926 10.4%, 66.9%	+703, +2254681 6.6%, 44.7%	+343, +1962274 3.2%, 38.9%	+343, +944053 3.2%, 18.7%
100	405, 57233	+31, +2048 7.7%, 3.6%	+67, +39144 16.5%, 50.6%	+49, +23847 12.1%, 36.0%	+24, 20595+ 5.9%, 33.6%	+24, +4040 5.9%, 7.1%
200	694, 168307	–	+53, +128545 7.6%, 76.4%	+38, +89906 5.5%, 53.4%	+21, +83809 3.0%, 49.8%	+21, +29828 3.0%, 17.7%
400	1380, 389688	–	+120, +312576 8.7%, 80.2%	+73, +220114 5.3% /56.5%	+38, +201421 2.8%, 51.7%	+38, +84058 2.8%, 21.6%
600	2065, 823937	–	+194, +596673 9.4%, 72.4%	+127, +411362 6.2%, 49.9%	+56, +365305 2.7%, 44.2%	+56, +173849 2.7%, 21.1%
800	2734, 1478704	–	+278, +881897 10.2%, 59.6%	+180, +564243 6.6%, 38.1%	+89, +482700 3.3%, 32.6%	+89, +221219 3.3%, 14.7%
1000	3417, 2131385	–	+398, +1417088 11.6%, 66.5%	+236, +945209 6.9%, 44.3%	+115, +809443 3.4%, 38.0%	+115, +431058 3.4%, 20.2%
C1	2914, 952995	–	+470, +490242 16.1%, 51.4%	+333, +326360 11.4%, 34.2%	+151, +246114 5.2%, 25.8%	+151, +131224 5.2%, 13.8%
C2	895, 443144	–	+158, +456219 17.7%, 103.0%	+113, +283553 12.6%, 64.0%	+48, +246208 5.4%, 55.6%	+48, +90111 5.4%, 20.3%
R1	2881, 1121497	–	+136, +712998 4.7%, 63.6%	+67, +482043 2.3%, 43.0%	+48, +386052 1.7%, 34.4%	+48, +263126 1.7%, 23.5%
R2	600, 720454	–	+18, +799141 3.0%, 110.9%	+7, +594375 1.2%, 82.5%	+3, +564348 0.5%, 78.3%	+3, +190399 0.5%, 26.4%
RC1	2801, 1064510	–	+218, +483182 7.8%, 45.4%	+120, +304586 4.3%, 28.6%	+65, +267166 2.3%, 25.1%	+65, +166324 2.3%, 15.6%
RC2	603, 746602	–	+110, +434145 18.2%, 58.1%	+63, +263764 10.4%, 35.3%	+28, +252385 4.6%, 33.8%	+28, +12792 4.6%, 1.7%

$O(N^3)$ for the travel time savings heuristic and $O(N^4)$ for the slackness savings heuristic.

5. EXPERIMENTAL EVALUATION

The experiments were carried out using the set of well-known Homberger-Gehring benchmark instances [6]. To provide reference to previous agent-based approaches we also included the original Solomon's instance set [14], sharing the same basic attributes as the formerly mentioned set. Thus the complete benchmark set consists of 6 sets of instances with 100, 200, 400, 600, 800 and 1000 customers respectively, with 60 instances in each set (except Solomon's with 56 instances). For each set there are 6 instance types provided — the R1, R2, RC1, RC2, C1, and C2 type, each with a slightly different topology and time windows properties. For C1 and C2 types the customer locations are grouped in clusters, unlike the R1 and R2 classes where the customers are randomly placed. The RC1 and RC2 instance types combine the previous two types with a mix of both random and clustered locations. The C1, R1 and RC1 also differ from C2, R2 and RC2 in terms of the scheduling horizon, the former having a shorter horizon resulting in routes of about 10 customers on the average, the latter having a longer horizon providing for routes of around 50 customers.

The reason for using these particular widely used benchmark sets was to provide a relevant comparison with the state-of-the-art centralized solvers that has been missing from previous agent-based studies. Therefore, the inclusion of the extended Homberger-Gehring benchmarks is one of the unique assets setting this work apart.

5.1 Algorithm Configurations

We examined four different settings for the suggested algorithm. The simplest *Algorithm-B* setting refers to a baseline algorithm not employing neither the *dynamicImprove* nor the $finalImprove1, 2$ functions. Such a setting corresponds to the simple parallel cheapest insertion procedure. *Algorithm-FI* extends the previous setting by employing the $finalImprove1$ function using one of the presented negotiation methods. The *Algorithm-DI* refers to a setting with both the *dynamicImprove* and $finalImprove$ functions being used. For all three mentioned settings the cost structure used throughout the whole algorithm corresponds to the slackness savings heuristic. We present these configuration to provide an insight into the role of the *dynamicImprove* and $finalImprove$ functions within the solving process.

The full fledged algorithm as presented within this study is denoted as *Algorithm-DIA*. It further extends the *Algorithm-DI* setting with a $finalImprove2$ function using the route length savings heuristic in an effort to address the secondary optimization criteria.

We used the *ReallocateAll* improvement method for the $finalImprove1, 2$ functions in all applicable settings. With respect to the *dynamicImprove* function, we tested all three presented negotiation methods in the applicable *Algorithm-DI* and *Algorithm-DIA* settings. The presented results correspond to the best of these three runs.

We used $\varepsilon = 0.3$ setting for the ε parameter affecting two of the three presented improvement methods. With respect to the α and β parameters affecting the slackness savings heuristic cost structure we used $\alpha = \beta = 0.5$ setting. The

Table 2: Equalled best known solutions per instance types

Instance Type	VN Equal	VN and RT Equal	RT Error on VN Equal
All	48.6%	8.1%	24.6%
C1	33.9%	18.6%	9.9%
C2	51.7%	27.6%	9.9%
R1	30.6%	1.6%	28.4%
R2	95.1%	1.6%	26.7%
RC1	6.9%	0.0%	10.9%
RC2	72.4%	0.0%	29.3%

choice of these particular values is discussed later in the text.

5.2 Evaluation of Results

The performance of our algorithm is illustrated by Table 1. Two commonly used metrics of quality are presented: (i) the cumulative number of vehicles (CVN) and (ii) the cumulative travel time (CRT). The "Best" column presents the best known CVN and CRT for given set of instances taken from [11, 12]. The rest of the columns corresponds to the absolute and relative errors in both criteria listed for previous agent based studies [8] (the *Agents* column) and for the four settings of the presented algorithm. The results are presented for individual instance sizes ("100" corresponds to the Solomon's instances, while the "200 – 1000" sets correspond to the Homberger-Gehring instances) as well as for individual instance types that are common among both benchmark sets. Thus, for example, the second row of the last column shows that on Solomon's instance set the *Algorithm-DIA* setting achieved a CVN of 24 more than the CVN of the best known solutions, with a CRT of 4040 higher, resulting in a 5.9% and 7.1% respective relative errors.

Table 2 further illustrates the success of the full *Algorithm-DIA* settings in terms of being able to match the best known solutions in terms of: (i) the number of vehicles (VN), (ii) both criteria (VN and total travel time - RT). Complementing this information is the relative error in RT for the solutions matching the best known VN.

5.2.1 Overall Quality Analysis

In overall, the presented algorithm in the full *Algorithm-DIA* setting achieved a 3.2% CVN and 18.7% CRT average relative error when compared to the best known solutions. The algorithm was able to match the best known solutions in 48.6% in terms of the primary VN optimization criteria. In 8.1% of the cases both VN and RT criteria of the best known solutions were achieved. The average relative RT error for the VN best known matching instances was 26.4%. The algorithm outperforms all previous agent-based approaches, achieving a CVN of 429 compared to 436 presented by [8] on the Solomon's instances. The algorithm sets new best known solutions for agent-based approaches on the extended Homberger-Gehring datasets.

The performance is consistent across all instance sizes. The difference in performance between the Solomon's and the extended Homberger-Gehring datasets corresponds to the fact that slower solvers are typically not tested on the extended datasets. Such is the case, for example, with the previously presented agent-based algorithm featuring within the comparison. The results achieved on both datasets thus suggest, that an agent based approach to VRPTW is a sound alternative to the traditional centralized solvers.

Table 3: Insertion heuristic relative errors over 200 customer instances

Algorithm setting	Slack savings	Travel time savings
Algorithm-B	7.6%	25.1%
Algorithm-FI	5.5%	11.1%
Algorithm-DI	3.0%	5.2%

5.2.2 Dynamic and Final Improvements Analysis

The results for the individual algorithm settings illustrate the significance of both the $dynamicImprove$ and the $finalImprove1, 2$ functions. With the *Algorithm-B* setting there is no possibility to recover from a potentially bad allocation taking place in the earliy stages of the allocation process. For example, an early allocation may render some of the subsequent allocations infeasible due to the time window or capacity constraints, effectively preventing some parts of the search space to be traversed. In overall the *Algorithm-B* setting achieved an error of 10.4% in the VN criteria.

The *Algorithm-FI* setting extends the *Algorithm-B* setting by allowing some exchanges of the tasks within and between the routes during the final stage of the allocation process. At this stage, however, as a result of previous allocations, the partial solution σ is already tightly constrained. Thus the chance of reallocating a task already in σ is correspondingly small, resulting in a relative VN error of 6.6% across all instances.

With an average relative VN error of 3.2% the *Algorithm-DI* setting significantly outperforms the *Algorithm-FI* setting. Arguably this is due to the fact that the improvements are performed dynamically throughout the allocation process on smaller and therefore less constrained partial solutions. The slackness savings heuristic specifically tries to minimize the constraining effects of the insertions. Therefore, the *Algorithm-DI* setting dynamically improves the partial solution σ in an effort to increase the chance of future advantageous allocations or reallocations being performed.

Finally, by employing the $finalImprove2$ function, the *Algorithm-DIA* traverses the feasible neighborhood of the resulting solution σ using a travel-time driven cost structure in an effort to find a local travel-time minima. A success of such an adaptive strategy is illustrated by reducing the 38.9% relative RT error from the previous *Algorithm-DI* setting to only 18.7%.

There is a notable difference in performance of *Algorithm-B* and *Algorithm-FI* settings on clustered (C1, C2, RC1, RC2) and non-clustered (R1, R2) instances. The customers in clusters are temporally and spatially very close while the distances between clusters are much higher. A good solution is thus characterized by minimizing the number of travels between the clusters. In an early partial solution σ where not all customers are yet known the *Algorithm-B* may easily make very bad decisions like having a vehicle visit more clusters — a situation from which the final improvement of the *Algorithm-FI* cannot recover. The dynamic improvements performed within the *Algorithm-DI* setting help to counter this to some extent.

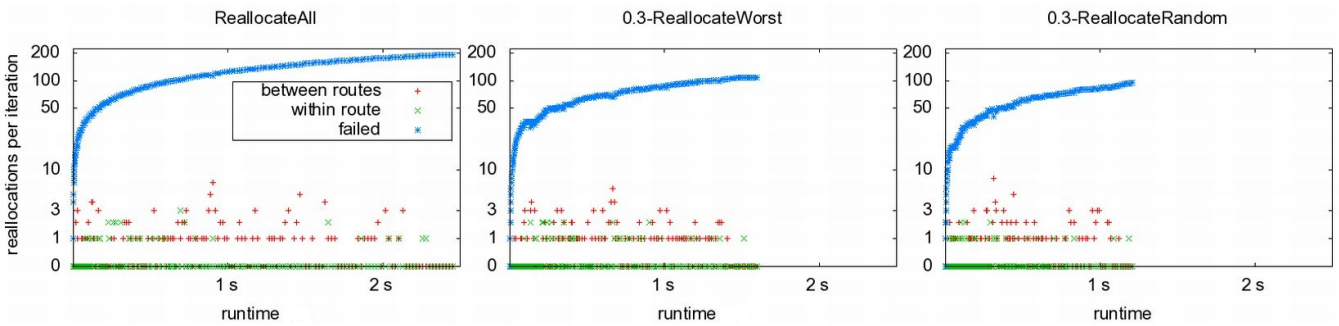


Figure 2: Improvement methods reallocation success

Table 4: Relative errors for insertion heuristics and individual orderings

	Algorithm-B		*Algorithm-DI*	
Ordering	Travel time	Slack	Travel time	Slack
MDF	46.0%	27.9%	14.6%	9.5%
TTF	28.2%	18.5%	11.4%	8.7%
EF	25.7%	12.3%	8.5%	6.4%
BEST	23.1%	7.6%	5.5%	3.0%
RAND	36.3%	23.0%	14.8%	9.7%

5.2.3 Insertion Heuristics Analysis

The commitment/decommitment cost structure provided by the insertion heuristics is the sole input for the otherwise abstract allocation process. Table 3 lists relative errors of the two presented heuristics measured for the 200 customer benchmark set. The results show that the slackness savings heuristic outperforms the traditional travel time savings heuristic in all three relevant algorithm settings (the $Algorithm-DIA$ setting actually uses both heuristics). The difference is most pronounced with the *Algorithm-B* setting while being less pronounced in *Algorithm-FI* and *Algorithm-DI* settings. Thus, interestingly, the improvement methods are able to exploit both of the heuristics with similar success. The results correspond to $\alpha = \beta = 0.5$ slackness savings heuristic parameters that has proved to be the most efficient in the computational tests that are outside the scope of this study.

Not surprisingly the slackness savings heuristic proved to be significantly slower of the two, with runtime in the *Algorithm-DI* setting being approximately 3 times longer.

5.2.4 Ordering Sensitivity Analysis

The relative errors for various initial orderings of the set of tasks T corresponding to the 200 customer benchmark set are listed by Table 4. The results for both the baseline *Algorithm-B* and the full-fledged *Algorithm-DI* settings and for each of the used insertion heuristics are presented. The BEST ordering row corresponds to the best results of MDF, TTF and EF orderings, while the RAND ordering corresponds to a baseline random ordering of the tasks.

The results suggest that neither of the proposed orderings is dominant in terms of outperforming the remaining two across the whole range of instances. To the contrary, the fact that the BEST results are significantly better than the results of either of the orderings proves that each of the orderings performs well on a different subset of instances. The results thus suggest that the individual instances differ in their nature, potentially favoring some particular ordering. For example, the MDF ordering attributed for 8 wins across the 60 measured instances suggesting that these may be the instances where leveraging the capacity aspect is beneficial, while in the rest of the cases the best results were achieved using orderings leveraging the temporal aspects of the problem. Finding an ordering that is fitting for a particular problem instance is an interesting problem in its own right that has not yet been addressed. To overcome this, we treated the set of orderings to be used as an additional parameter of the algorithm, running the *allocate* function for each ordering and choosing the best result from these runs. The presented results and runtimes correspond to the MDF, TTF and EF orderings being used.

The results further show that out of the two heuristics the slackness savings heuristic is clearly the less sensitive to the ordering of the two in the baseline *Algorithm-B* setting. In the *Algorithm-DI* setting the difference is less marked. This suggest that the dynamic improvement methods are successful in offsetting the sensitivity of used heuristic to ordering, a result that supports previous findings of [7].

5.2.5 Improvement Methods Analysis

We analyzed the respective performance of individual improvement methods used within the *dynamicImprove* function of the *Algorithm-DI* setting varying the ε parameter where applicable. Surprisingly, beginning with $\varepsilon = 0.3$, the quality of the resulting solution did not improve with bigger ε while the runtime did increase linearly. For the final improvement, we found the *ReallocateAll* method to achieve marginally better results. In overall, we found that the number of reallocations does not have a strong positive influence on the quality of resulting solution.

Figure 2 illustrates the number of reallocations performed within the individual calls of the *dynamicImprove* function for a 200 customer instance and the runtime in which they occurred within an invocation of the *allocate* function. Note that these results correspond to the slackness savings heuristic based implementation of the *costCommit* and *costDecommit* functions. Note also that the y axis is presented in logarithmic scale. The three types of points correspond to: (i) reallocations of tasks between routes, (ii) reallocations of tasks within a single route and (iii) reallocations that failed to find a feasible improving allocation for the task being reallocated. The three graphs correspond to

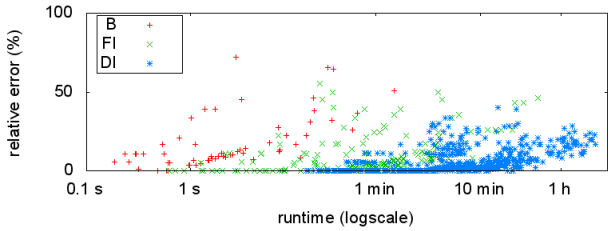


Figure 3: Results for individual algorithm settings for 1000 customer instances

Table 5: Cumulative and worst runtimes for individual instance sizes

Size	Nagata [11] Avg. RT	Lim [9] Avg. RT	*Algorithm-DIA* Avg. RT	*Algorithm-DIA* Worst RT
200	1 min	10 min	3 s	13 s
400	1 min	20 min	22 s	3 min
600	1 min	30 min	2 min	19 min
800	1 min	40 min	6 min	47 min
1000	1 min	50 min	8 min	74 min

the three presented improvement methods with $\varepsilon = 0.3$ for the two ε methods. The rising curve shape for the number of failed reallocations corresponds to the fact that throughout the process more tasks are being allocated and processed by the respective improvement methods.

The results suggest that neither of the methods is dominant terms of reallocation success. Contrary to our expectations, the ε-ReallocateWorst method did not succeed in selecting the most likely to be reallocated tasks. Furthermore, the number of successful reallocations drops towards the end of the solving process, suggesting that the cost structure provided by the slackness savings heuristic together with the proposed improvement methods get stuck in local optima. The inability to further transform the solution towards higher quality is arguably due to the fact that the number of feasible task reallocations drops rapidly as the solution gets denser.

The presented negotiation methodology only allows for traversing the solution space of partial solutions that are feasible in terms of time window and capacity constraints. We argue that for the negotiation based methodology to achieve stronger results than the arguably very promising results presented within this study, a method allowing for traversing also the infeasible space or performing more complex moves would have to be developed. Such a methodology could be embedded to the presented general solving architecture in form of some backtracking strategy and accompanying ejection based heuristic, allowing for temporal infeasibility of individual routes.

5.2.6 Runtime and Convergence Analysis

The convergence of *Algorithm-B, Algorithm-FI* and the full *Algorithm-DIA* settings is illustrated by Figure 3. The results correspond to the 1000 customers benchmark set. Note that the x-axis uses a logarithmical scale. The results confirm that the quality and robustness of the algorithm increase with more complex setting being used with obvious penalty in terms of runtime. Also the difference in terms of runtime between individual algorithm settings is dramatic. Interestingly, in many cases the very short-running settings produce a very good quality results matching even the best known solutions. This feature of the algorithm can be exploited by running various strategies in parallel competition returning an improving sequence of results over time. Based on previous evidence, using a wide set of task orderings in combination with the shorter running solvers might, for example, produce a very efficient and robust strategy with ideal parallelization features.

The comparison in terms of runtime of the full *Algorithm-DIA* setting with the currently leading algorithms is presented by Table 5. The average runtime-per-instance as well as the worst runtime recorded by presented algorithm is listed for individual instance sizes of the extended benchmark sets. The results correspond to a C++ implementation run on AMD Opteron 2.4G system for [11], a Java implementation run on a Intel Pentium 2.8G system for [9] and a normalized (single threaded) runtime on a 4G RAM AMD Athlon 2G Gentoo system running the 64-bit Sun JRE 1.6.0_22. The approach to parallelization is not mentioned in neither [9] nor [11]. Also, the secondary travel time minimization criteria is not addressed by [11]. We must note, however, that: (i) compared algorithms outperform presented algorithm in terms of CVN and (ii) are not computationally bound. Therefore to be able to draw a more relevant conclusions, settings with similar solution quality would have to be compared.

With respect to previously presented agent-based algorithms, no comparable data were provided by any of the previous works. The likely cause for that is that agent-based approaches typically rely on agent execution platforms corresponding to a loosely coupled distributed environment (JADE etc.) making them extremely inefficient.

The results show that there is a striking difference between the average and the worst runtime for the presented algorithm. The worst results correspond to the instances where the initial vehicles count estimation was much lower than the VN of the particular solution being found. In such a case the *allocate* function is restarted with sequentially increasing empty vehicles count until a feasible solution is found. The effect is most pronounced given an unfitting ordering is used for the particular run. Also, the effect increases the size of the instance. Looking back at Figure 3, the above mentioned observation is clearly illustrated. Apparently, for each respective algorithm setting, the runtimes for the individual results increase with decreasing quality of the corresponding solutions. We suggest that an improved restarts strategy could be developed, addressing this shortcoming. Such a strategy could benefit from performing variable size steps based upon analysis of the partial solution at the end of last step — the number of remaining unallocated tasks in particular. Also the set of orderings could be pruned based on their respective performance in shorter running settings e.g. *Algorithm-B, Algorithm-FI*. Another option is a restart strategy reusing the partial solution σ from the previous step (keeping the agents' commitments), but such an approach didn't prove successful in our testing.

The last interesting conclusion concerns the overall convergence attributes of the presented algorithm. Considering the previously discussed high ratio of unsuccessful reallocations, the above mentioned high number of restarts and the

fact that the implementation is a prototypal one, rather than a fully optimized one, the listed runtimes actually correspond to a very limited portion of the search space being traversed in comparison with the competing algorithms. This suggests that the algorithm navigates through the search space very efficiently providing for a very good convergence. This finding provides further evidence of the potential of the method and suggests that further research is needed to unlock its full potential.

6. CONCLUSION

This paper describes an algorithm for the VRPTW based on agent negotiation. The performance of the algorithm is evaluated using the well known benchmark sets in an effort to assess the relevance of agent based approaches to routing problems in general. The algorithm outperforms all previously presented agent based algorithms, being also the first agent based algorithm to be tested using the extended benchmark sets typically used by centralized performance optimized solvers. Experimental results show that the algorithm is able to match the best known solutions achieved by the centralized solvers in 48.6% of the cases with an average relative error of 3.2% across all tested instances with respect to the VRPTW primary optimization criteria.

The algorithm uses a generic negotiation based task allocation process embedded in a multi-agent hierarchy that promises to be flexible in terms of capturing the semantics of typically heterogenous, dynamic and privacy conscious systems modeled within the transportation domain. For purposes of this paper, the adaptation to the VRPTW is achieved by supplying specific cost estimation functions for agent commitments and decommitments that can be easily extended or modified.

A comprehensive analysis of the algorithm is presented suggesting promising future research opportunities in: (i) modifying presented allocation process to employ more complex moves or allow for traversing non-feasible search space, (ii) developing a method to identify the best fitting ordering for a particular VRPTW instance, (iii) improving the restarts strategy for the VRPTW case and (iv) adapting the system for more challenging problem variants exploiting its inherent flexibility.

7. ACKNOWLEDGEMENTS

This work was supported by the Ministry of Education, Youth and Sports of Czech Republic within the Research program MSM6840770038: Decision Making and Control for Manufacturing III and also within the grant no. LD12044.

8. REFERENCES

[1] O. Bräysy and M. Gendreau. Vehicle routing problem with time windows, part I route construction and local search algorithms. *Transportation Science*, 39(1):104–118, 2005.

[2] O. Bräysy and M. Gendreau. Vehicle routing problem with time windows, part II metaheuristics. *Transportation Science*, 39(1):119–139, 2005.

[3] A. M. Campbell and M. Savelsbergh. Efficient insertion heuristics for vehicle routing and scheduling problems. *Transportation Science*, 38:369–378, August 2004.

[4] Z. Dan, L. Cai, and L. Zheng. Improved multi-agent system for the vehicle routing problem with time windows. *Tsinghua Science Technology*, 14(3):407–412, 2009.

[5] K. Fischer, J. P. Müller, and M. Pischel. Cooperative transportation scheduling: an application domain for dai. *Journal of Applied Artificial Intelligence*, 10:1–33, 1995.

[6] J. Homberger and H. Gehring. A two-phase hybrid metaheuristic for the vehicle routing problem with time windows. *European Journal of Operational Research*, 162(1):220–238, 2005.

[7] R. Kohout and K. Erol. In-time agent-based vehicle routing with a stochastic improvement heuristic. In *11th Conference on Innovative Applications of Artificial Intelligence*. AAAI/MIT Press, 1999.

[8] H. W. Leong and M. Liu. *A multi-agent algorithm for vehicle routing problem with time window*, page 106Ű111. ACM, 2006.

[9] A. Lim and X. Zhang. A two-stage heuristic with ejection pools and generalized ejection chains for the vehicle routing problem with time windows. *INFORMS Journal on Computing*, 19(3):443–457, 2007.

[10] Q. Lu and M. M. Dessouky. A new insertion-based construction heuristic for solving the pickup and delivery problem with hard time windows. *European Journal of Operational Research*, 175:672–687, 2005.

[11] Y. Nagata and O. Bräysy. A powerful route minimization heuristic for the vehicle routing problem with time windows. *Operations Research Letters*, 37(5):333–338, 2009.

[12] Sintef. Sintef web pages - transportation optimization portal, problems section.

[13] P. Skobelev. Multi-agent systems for real time resource allocation, scheduling, optimization and controlling: Industrial applications. In V. Maríık, P. Vrba, and P. Leitao, editors, *Holonic and Multi-Agent Systems for Manufacturing*, volume 6867 of *Lecture Notes in Computer Science*, pages 1–14. Springer Berlin / Heidelberg, 2011.

[14] M. M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35:254–265, 1987.

[15] J. Vokřínek, A. Komenda, and M. Pěchouček. Agents towards vehicle routing problems. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1 - Volume 1*, AAMAS '10, pages 773–780, Richland, SC, 2010. International Foundation for Autonomous Agents and Multiagent Systems.

[16] J. Vokřínek, A. Komenda, and M. Pěchouček. Abstract architecture for task-oriented multi-agent problem solving. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 41(1):31 –40, January 2011.

Coevolution and Transfer Learning in a Heterogeneous, Point-to-Point Fleet Coordination Problem

Logan Yliniemi
Oregon State University
Corvallis, OR, USA
logan.yliniemi@engr.orst.edu

Kagan Tumer
Oregon State University
Corvallis, OR, USA
kagan.tumer@oregonstate.edu

ABSTRACT

In this work we present a multiagent Fleet Coordination Problem (FCP). In this formulation, agents seek to minimize the fuel consumed to complete all deliveries while maintaining acceptable on-time delivery performance. Individual vehicles must both (i) bid on the rights to deliver a load of goods from origin to destination in a distributed, cooperative auction and (ii) choose the rate of travel between customer locations. We create two populations of adaptive agents, each to address one of these necessary functions. By training each agent population separate source domains, we use transfer learning to boost initial performance in the target FCP. This boost removes the need for 300 generations of agent training in the target FCP, though the source problem computation time was less than the computation time for 5 generations in the FCP.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems

General Terms

Algorithms, Management, Performance, Reliability

Keywords

Multiagent Learning, Transfer Learning, Coevolution, Logistics

1. INTRODUCTION

The use of semi tractor-trailers to move large amounts of goods from one place to another is the backbone of the economy in developed nations. In the United States, over 70% of all transportation of commercial goods was conducted by truck, dwarfing all other forms of freight transportation [2]. Trucking companies face the complex problem of routing their vehicles in such a way that they complete their contracted deliveries on-time, while spending minimal fuel and other resources. Fuel efficiency of individual vehicles has steadily increased in recent years, due to a high amount of research attention to aerodynamics and fuel efficient designs [8, 10, 27]. This, however, only addresses one side of the problem. If a company poorly assigns these delivery tasks to vehicles in its fleet, a large amount of fuel can wasted by vehicles traveling "empty miles"—miles traveled where a vehicle has no cargo [12]. Previous work on this subject centers around the Vehicle Routing Problem (VRP).

The VRP addresses the need to minimize the resources consumed in a road-vehicle-based logistics environment [6]. The original VRP creates a static customer set with a set of demands for a single good that must be satisfied by a single depot to the customer set. Each customer typically has a set window of time within which they will accept deliveries. Solutions to the VRP typically seek to minimize the number of vehicles necessary to complete the delivery set [6]. Algorithms that solve the classic VRP typically fall into one of three categories: route construction, local search, or metaheuristics [9]. Each of these solution types are typically nonadaptive and centralized, and require full observability of the system. The solutions also have very little generalizability, and solving a new problem instance will take as long as solving the first, even if the two are very similar [9].

We address this problem by framing the domain as a distributed multiagent system with adaptive agents. This puts our solution strategy in the category of metaheuristics, which are typically noted for being slow to calculate. However, we leverage the benefits of transfer learning, in which experience from one problem instance can be used to boost performance in another problem instance. By doing this, less adaptation is needed between problem instances, and acceptable performance can be attained in a new problem instance in significantly less time than solving the new system from scratch.

The classic VRP has been extended multiple times to include more realistic demands. One example includes soft time windows to the classic formulation, where a delivery may be made outside of the desired time, with a penalty assessed proportional to the time the delivery is outside of the desired window [9, 14]. Other extensions allow for the inclusion of multiple depots from which deliveries may originate [23, 15], backhauls that must be made in which the customers have goods to deliver to the depot [13], simultaneous pickups and deliveries for back hauls [7, 17], heterogeneous vehicles [25], and time-varying travel speed [28]. Each of these extensions incorporates an additional level of realism into the problem, but each also leaves out the contributions of the other problem formulations.

In collaboration with an industry partner, we identified key extensions to the VRP that best embody the day-to-day operation of a truckload or less-than-truckload carrier [16]. These extensions include (i) a fixed fleet size, (ii) soft time windows, (iii) heterogeneous vehicles, and (iv) multiple depots, which we expand such that each customer acts as a depot, creating a point-to-point delivery problem. We in-

clude *all* of these extensions in the domain considered in this work. We also include a novel extension of the VRP: (v) an elective, nonlinear tradeoff between travel time and fuel expenditures.

The tradeoff between travel time and travel cost serves to model that in many cases, there are multiple routes from one location to another that trade between time efficiency and fuel efficiency: very often a more fuel efficient route may be available, and simply take more time to traverse. The inclusion of these extensions to the Vehicle Routing Problem creates a problem domain that brings a different focus to the coordination required for a logistically-viable solution. Instead of a centralized controller that develops a set of routes, each agent must coordinate with the others such that the system performs well as a whole.

The crux of the problem then becomes controlling the vehicles in such a way that the fleet is well coordinated, to reduce "empty miles" and other wasteful fuel consumption. As such, we term our formulation of the VRP the Fleet Coordination Problem, or FCP.

While algorithms exist to address each of these extensions (i–v) individually, none have been created that incorporate them all. In this work we propose an adaptive, distributed control technique for assigning the responsibility for a given delivery event through a multiagent auction to minimize fuel consumption. By addressing the problem in this manner we can take advantage of transfer learning to allow experience gained in one problem instance to generalize to other "target" problem instances. This also allows us to train the agents initially on simple "source" problems to boost their adaptivity or performance in the target problem.

The major contributions of this work are to:

- Provide a distributed, adaptive solution strategy to a complex Fleet Coordination Problem
- Show that transfer learning allows agents trained in a simple source problem to significantly reduce required training time in the FCP
- Show that agents continue to learn through coevolution in the FCP
- Demonstrate robustness to calculation approximations and partial transfer learning

The remainder of this paper is organized as follows: Section 2 addresses background involving the VRP, transfer learning, evolutionary algorithms, and coevolution. Section 3 provides a complete description of the novel Fleet Coordination Problem addresssed in this work. Section 4 provides a treatment of the experimental methods and algorithms used in this work, including a full description of the source problems used. Section 5 contains the experimental results of this work, which show significant gains through using transfer learning in the FCP, even in the presence of calculation approximations or information loss. Section 6 draws conclusions from this work and addresses future research directions.

2. BACKGROUND

In Section 2.1, we define the Vehicle Routing Problem and describe a number of previously studied extensions to the VRP, as well as solution strategies that have been used to address these problems. In Section 2.2 we provide the relevant background on transfer learning and its previous use in various domains. Section 2.3 provides background on coevolution.

2.1 Vehicle Routing Problem

A classic version of the VRP is formalized as:

> **Definition : VRP** The classic Vehicle Routing Problem (VRP) consists of a depot D and a set of nV homogeneous vehicles $V = \{v_1, v_2, ..., v_{nV}\}$ with a common maximum load Q_{max} and maximum route length L_{max}. These vehicles must service a set of nC customers $C = \{c_1, c_2, ..., c_{nC}\}$ with demand $q_i \in \mathbb{N}$ for a good provided by depot D. Each customer must be serviced by exactly one vehicle, and each vehicle must return to the depot at the end of its route. The goal is to minimize the number of vehicles nV required to service all customer demand [6, 23].

Early approaches to the VRP included three primary methods. Direct solution approaches were only viable for small problems, while heuristic methods and methods based on the Traveling Salesman Problem were able to handle larger problem instances [6]. Since the development of these early solution strategies, work in the VRP has both focused on incorporating realistic extensions to the VRP as well as finding new solution strategies.

Two simple extensions include the Vehicle Routing Problem with Hard Time Windows and Vehicle Routing Problem with Soft Time Windows. These extensions add time bounds within which each customer may be serviced. In the case of hard time windows, no early or late deliveries are allowed. In the case of soft time windows, these deliveries are allowed, but are penalized proportionally to the deviation from the prescribed time window [9]. These variations increase the problem complexity significantly, and are readily incorporated into other VRP variations. Most VRP implementations use hard time windows as an implied constraint unless explicitly stated otherwise.

Two other extensions to the VRP include the Vehicle Routing Problem with Backhauling (VRPB) which provides each customer with a supply $s_i \in \mathbb{N}$ that must be hauled back to its originating depot, and the Vehicle Routing Problem with Simultaneous Pickups and Deliveries, which is a form of the VRPB in which the dropoff of the demanded goods and the pickup of the supplied goods must occur simultaneously (in order to minimize loading effort on the part of the customer) [13, 7]. These extensions are typically not incorporated into other variations on the VRP.

Another variant of the VRP is the use of heterogeneous, fixed-size fleets [25]. In this formulation, a fixed number of multiple types of vehicles is available to deliver goods to customers. Each different vehicle type has a unique maximum capacity and cost per unit distance traveled.

One final notable extension is the Vehicle Routing Problem with Multiple Depots, which changes the single depot D into a set of depots $D = \{d_1, d_2, ..., d_{nD}\}$. One notable treatment of this problem was carried out by Léauté et. al, who framed the problem as a Distributed Constrained Optimization Problem (DCOP) using various modern techniques to solve the DCOP such as SynchBB, DPOP, and P-DPOP before solving the resulting VRPs with a locally centralized controller [15].

In this work, we incorporate many of these into a single problem domain. The FCP, described in section 3, uses soft time windows, a heterogeneous fixed fleet, and as many depots as customers (creating a point-to-point delivery problem) as well as incorporating an extension not found in the literature: an elective tradeoff between travel speed and travel cost.

The solution strategies for various versions of the VRP have a number of shortcomings. Many of the solutions are not generalizable from one problem type to another (though some special circumstances do exist where solutions from one problem type can be generalized to another, e.g. solutions to the VRP with Soft Time Windows can be used for the VRP with Hard Time Windows if the penalty for early or late service is high enough), or even one problem instance to another of the same type [9]. Additionally, many of the algorithms are centralized, and require full system observability for calculations.

Decentralized solution strategies have been used to address some variations of the VRP [23, 15, 3]. Of these, however, some solve the problem using decentralized coordination, while others merely divide the problem into smaller VRPs and solve each of these with a centralized controller [15].

2.2 Transfer Learning

None of the solution approaches used for the various embodiments of the VRP would be suitable for the FCP without significant adjustment, or would only work on a specialized subset of problem instances within those allowed by the FCP. However, an adaptive, multiagent approach—wherein an agent senses information about its environment, reasons based on that information, and takes some action — bears a number of advantages for the FCP. First, framing the problem in a multiagent setting allows for an effectively decentralized approach with minimal communication between agents. Second, using adaptive agents can allow the use of transfer learning (TL) to leverage experience gained in one problem instance to increase performance in another problem instance [5].

At its simplest, transfer learning can allow an agent that takes actions based only on local state information to use the same policy in a different instance of an identically-formulated problems [26]. However, transfer learning can also be leveraged to use a simple training domain, or "source" to boost performance in a more complex problem domain, or "target", as long as the policies can be represented in a similar manner in both cases, and the experience gained in the source problem is valuable in the target problem.

That is, an agent trained on source problems similar to the target problem will gain benefits in performance or trainability in the target domain, but agents trained on a random task will not gain any performance benefits, and may in fact be hurt by the transfer [5, 18].

In this work we use a function approximator for each agent, in the form of a single-layer, feed-forward neural network. Such neural networks are powerful computational tools that can serve as function approximators and have been used in transfer learning in previous work [5, 21]. These neural networks have been used in applications as complex and diverse as computer vision, HIV therapy screening, and coronary artery disease diagnosis [1, 4, 22].

Success in transfer learning is typically a function of the similarity between the source problem and target problem, training time on the source problem, and validity of the agent's representation in both the source and target problems [11, 18]. In the ideal case, the agents representation fits both problems very well, and knowledge is well represented for transfer. This occurs when the states seen and actions taken are similar in both cases.

In this work we use transfer learning by producing single agent domains that act as a microcosm of the FCP, in which agents face similar, but not identical decisions. By training on these domains, described in Section 4 before directly using the developed policies into the FCP, we gain benefits both in initial performance and learning speed.

2.3 Evolution and Coevolution

To allow the agents to adapt to their environment, in hopes of increasing the performance of the system as a whole, we need a way to affect the policies with which the agents reason about which actions to take in whatever state they sense. In this work, we achieve this through the use of both evolutionary algorithms, and coevolution.

Evolutionary algorithms are a biologically-inspired computational technique in which a population of agent policies is first randomly generated, and then tested in some domain. After calculating a scalar measurement of an agent's "fitness" for each agent, those with lower fitness are replaced with slightly-altered copies of their higher-fitness counterparts. Through this random alteration and intelligent selection, system performance increases as the agents adapt to the domain to maximize their fitness calculation.

Coevolutionary algorithms leverage the concept of evolution for team-based domains. In coevolution, multiple separate populations are maintained, and are used in a shared simulation environment, where their fitness is evaluated based on how well they perform an assigned task as a member of a team made up of members from each population. An evolutionary algorithm is carried out on each population individually, such that the populations eventually produce agent policies that are well-suited in the team-based environment, to maximize the team's calculated fitness.

Coevolutionary algorithms have the potential to speed up a search through a complex space (which readily characterizes the FCP), but can often lead to a suboptimal area of the search space [19, 21]. This can be due to the agents learning to take a conservative strategy, being able to cooperate with a broader range of teammates [19, 20]. However, in this work we use an evolutionary algorithm on each agent population before using transfer learning and incorporating coevolution in the FCP (Section 4.2 – 4.3), so we have already guided the populations into favorable areas of the search space.

3. FLEET COORDINATION PROBLEM

The Fleet Coordination Problem is a variant on the classic VRP that includes the following extensions: (i) soft time windows, (ii) customer locations acting both as depots and delivery locations, (iii) a heterogeneous fleet of vehicles, (iv) a fixed fleet size, and (v) an agent-determined tradeoff between time efficiency and fuel efficiency (which can be intuitively thought of as the vehicle choosing a "speed" without much conceptual loss).

The FCP can be formally defined as:

Definition : FCP The Fleet Coordination Problem (FCP) consists of an allotted time $T = \{t|0 \leq$

$t \leq T_{end}\}$, a set of customers $C = \{c_1, c_2, ..., c_{nC}\}$ in the 3-dimensional Euclidian space, a set of nP packages $P = \{p_1, p_2, ..., p_{nP}\}$, each consisting of a set of: a weight $0 \leq w_i \leq W_{max}$, a customer origin for the package $c_j^a \in C$ located at λ_j^a, a customer to deliver the package to, $c_j^b \in C$, located at λ_j^b, the beginning and end times within which the delivery must be completed, $t_j^a, t_j^b \in T$. Each package $p_j = \{w_j, c_j^a, c_j^b, t_j^a, t_j^b\}$, must be delivered point-to-point by one of a set of nV nonhomogeneous vehicles $V = \{v_1, v_2, ..., v_{nV}\}$ which are each described by fuel efficiency, weight, and allowed cargo weight $v_i = \{\eta_i, w_i, \kappa_i\}$. Each vehicle v_i travels along each of the nK "journey legs" described by edges connecting each customer directly to each other at a unitless rate of travel $R_k \in [0, 100]$ The goal is to maximize the system level utility $\mathbb{G}$, measured as a combination of the negative total fuel consumed by all members of the fleet and their on-time performance (Equation 10).

There are a few key points to note in this formulation. First, it is possible to travel from any customer c_a to any other customer c_b directly. This is an abstraction of a road system, which would realistically pass through a customer c_c if that customer was sufficiently close to the straight-line path. Each trip from a customer c_a to c_b consists of $nK = 10$ "journey legs" ("legs" for short), regardless of the length of these legs. Agents may decide on the tradeoff between fuel efficiency and time efficiency for each leg independently, but the decision holds for the entire leg. This assumption was made so that calculations would scale relative to the number of packages to be delivered in a system, rather than the distance travelled by the vehicles in an experimental run.

Also, the customers are not assumed to be on the Euclidian plane, and in fact exist in a three-dimensional environment. The slope between any two customer locations is limited to be less than a 6% grade. This adds the complexity of uphill and downhill travel into the decision-making required by each agent and the fuel efficiency calculation [24].

The fuel cost for traveling from customer c_a to customer c_b is characterized as a sum over all of the k legs of the journey:

$$F_{tot} = \sum_{k=1}^{nK} F_{elec,k} + F_{req,k} \quad (1)$$

where

$$F_{req,k} = \eta_i \alpha_1 \delta_k (1 + w_i \sin(S_k)) \quad (2)$$

$$F_{elec,k} = \eta_i \alpha_2 \delta_k R_{j,k}^2 + \eta_i \alpha_3 \delta_k R_{j,k} \quad (3)$$

where α_{1-3} are experimental coefficients that are held constant through experimental runs, η_i are vehicle-specific fuel-efficiency parameters, δ_k is the distance of leg k, w_j is the weight of truck j, S_k is the slope of leg k, and $R_{j,k}$ is the "rate-of-travel" of truck j over leg k [24]. This $R_{j,k}$ value can be loosely interpreted as a speed of travel but is more accurately described as both a function of speed and fuel efficiency of a vehicle's chosen route.

Intuitively, these fuel expenditure equations break down to this: Equation 2 calculates the minimum cost of travel between two points, which is a function of the distance between the two points, and the slope of the road between them. Equation 3 models that the vehicles may choose to move along more or less fuel efficient paths (modeled by the linear term) at a more or less fuel efficient speed (modeled by the quadratic term) [24].

Algorithm 1 FCP Execution Algorithm

```
for j = 1 → total_packages do
  λ_a ← Package origin location
  λ_b ← Package destination
  for i = 1 → total_vehicles do
    if vehicle i is "busy" then
      Bidding agent i bids β_{i,j} = 0
    else
      Bidding agent i bids a value β_{i,j} ∈ [0, 1]
    end if
  end for
  Find highest bidder : v_win = argmax_i(β_{i,j})
  Move v_win to origin: Algorithm 2 (λ_{v_i}, λ_a)
  Increase weight of v_win by package weight w_j
  Move v_win to destination: Algorithm 2 (λ_a, λ_b)
  Decrease weight of v_win by package weight w_j
  Determine if package j was delivered within desired delivery window (Equation 6)
end for
```

The reason for the forms of these equations follows: for any journey, there is a physical absolute minimum fuel that has to be spent to complete the journey. This amount of fuel for journey k is F_{req}. Beyond that, due to the choice in driving habits of the driver, and route choices by a navigator, more fuel can be spent to get to a destination faster. This is represented by F_{elec}.

This fuel consumption model is not intended to compete with to the state of the art. This model was chosen to limit computation time, while still lending a degree of realism to the problem domain [16].

Paired with this, the rate-of-travel decisions $R_{j,k}$ also affect the time of travel between the origin and destination:

$$T_{tot} = \sum_{k=1}^{nK} \frac{\delta_k}{R_{j,k}} \quad (4)$$

where T_{tot} is the total time it takes for the vehicle to travel from the two locations, λ_a and λ_b.

It is also specifically important to note that all package deliveries must be completed; they may not be refused, and they stay in the domain until the completion of a delivery (some methods for internet routing allow for an amount of packet loss and design this into the approach; this is not viable for this problem domain).

Though we strove for applicability, This representation of the FCP is still a rough representation of reality, and cannot incorporate all facets of the real-world problem.

4. METHODS AND ALGORITHMS

We take a vehicle centric approach, wherein agents are associated with the vehicles; other agent-based distributed approaches are possible, with depots or packages themselves treated as agents, but in the FCP, choosing a vehicle-centric approach makes intuitive sense.

Because the target problem domain (the FCP, described in Section 3) is so complex, we break the agent responsi-

bilities into two categories: (i) the movement of the agent from one location of another, and (ii) the decision of which agent will be responsible for each package. We explain these decisions in detail in Section 4.1. These two responsibilities are trained on different source tasks, which are laid out in Sections 4.2 and 4.3.

4.1 Distributed Auction and Agent Populations

First, an agent must decide how much the immediate importance the vehicle will give to speed and fuel efficiency, respectively, in order that the vehicle might be mobile across the domain. This "driver" decision must be made during each leg of each journey from a customer c_a to c_b, and this directly impacts both the fuel spent on that leg, and the time spent traversing that leg (Equations 1-4).

The other agent responsibility is choosing which package pickup and delivery events each vehicle will be responsible for. To solve this portion of the problem, we use a distributed, one-shot auction in which the highest bidder takes responsibility for traveling to the customer at which the package originates, picking up the package, and delivering it to its final destination. The agents each bid a value $\beta_i \in [0, 1]$ that represents how well-suited they believe their vehicle is for handling that package.

These two tasks, though they deal with similar information — an agent must consider the distance away from a package origin both in choosing a speed and in choosing a bidding value — are very different decisions. To address this, we took the split responsibilities and assigned them to two *completely separate* agent groups. That is, instead of one agent being assigned to each vehicle, a team of two agents, consisting of one "driving" agent and one "bidding" agent, is assigned to the vehicle, to work as a team. We characterize each of these agents as a 4-input, 10-hidden unit, 1-output feed-forward neural network.

The actions of each of these agents affects the performance of their teammate as well as the system-level performance $\mathbb{G}$, in turn. Because the actions taken by the two agents are very different, however, we choose to initially train them in different source environments, to leverage the maximum possible benefit from transfer learning. These two source problems are described in the following sections.

4.2 Evolution in Driver Source Domain

To train the driving agents, we pose the simplest possible problem, such that the driver learns the effect of its actions on the outcomes as quickly as possible. The source problem that we train the driver on is characterized as follows:

Definition : Driver Source Domain (DSD) A vehicle v is placed at the location of customer c_a, and assigned a package p_j of weight w_j. It must travel to the location of customer c_b (Algorithm 2), with the goals of minimizing total fuel consumed during the journey F_{tot} (Equation 1), and arriving before a time $T_f \in T$.

Algorithm 2 Travel Algorithm

Given origin and destination locations (λ_a, λ_b)
Determine the length of each journey leg, δ_k
for $k = 1 \rightarrow$ *journey_legs* **do**
  Select rate of travel $R_{(i,k)}$
  Calculate required and elective fuel spent (Equations 2 and 3)
  Calculate total fuel spent F_{tot} (Equation 1)
  Calculate time spent T_{tot} (Equation 4)
  Mark vehicle as "busy" for the time spent.
end for

Algorithm 3 Evolutionary Process

Initialize 100 population members of neural networks with small weights.
for $g = 1 \rightarrow$ *end_generation* **do**

Simulation Step (varies with domain)
DSD: Simulate DSD $\forall$ agents i (Section 4.2)
or
BSE: Calculate β_i $\forall$ agents i (Equation 7, Section 4.3)
or
FCP: Choose nV agents at random, perform FCP (Algorithm 1); Repeat until all agents have participated once.

Fitness Step
  Calculate fitness of all agents: $U_{i,g}$ $\forall$ i (Equations 5, 8, 9 or 10)
  Sort agents from highest to lowest fitness

Selection Step (select 20 survivors)
  set $counter = 20$
  set $survive_i = 0 \; \forall \; i$
for $z = 1 \rightarrow 20$ **do**
  (select high-fitness survivors)
  With probability $(1 - \epsilon)$,
    $counter \leftarrow counter - 1$
    and set $survive_i = 1$
end for
for $z = 1 \rightarrow counter$ **do**
  (select random survivors)
  Select a random agent j with $survive_j == 0$
  Set $survive_j = 1$
end for

Mutation Step (repopulate to 100 agents)
for $Z = 1 \rightarrow total\_agents$ **do**
  if $survive_Z == 0$ **then**
    Select a parent network Y where $survive_Y == 1$
    Set all weights of $Z \leftarrow$ weights of Y
    Mutate all weights using triangular distribution of mean 0, maximum and minimum change ± 0.05
  end if
end for

end for

With only two locations, one package, and one vehicle, this embodies an extremely simple training case. The single agent makes only ten decisions (the rate of travel for each leg of the journey) before receiving feedback, allowing for the feedback to be very specific to the individual agent and much easier to learn compared to the target multiagent, long-term environment.

Specifically, the agent is a single-layer, feed-forward neural network that takes as inputs the distance to the destination

$\delta_{(i,b)}$, the slope of the next leg of the journey S_k, a measure of "time pressure" ψ_j, and the vehicle weight w_v ; and gives as outputs $R_{j,k}$, the "rate-of-travel" of the vehicle it is controlling along the kth leg of the journey. ψ_j takes on a value of 1 if the vehicle can make it to its destination at a (unit-less) rate of travel of 75[1], a higher value if the vehicle is under more time pressure for the delivery (a faster speed is necessary), and a lower value if a lower speed would still result in an on-time delivery. This was done to give the agents a sense of time that scaled correctly with the problem. It is roughly equivalent to the human intuition of "being on time", while still en route to a destination.

The driving agents face exactly these same decisions within the FCP. Though the system-level effects of their decisions are not fully expressed within this domain, the simple principles that it is generally better to be on time than late, and better to be efficient about fuel expenditures are expressed within this domain. The specifics of the training algorithm follow.

The agents are trained through an evolutionary algorithm, in which a population of 100 agents is allowed to perform on the same instance of the DSD before downselection and mutation occurs (Algorithm 3). After each agent has performed once on a given problem instance (over the course of one generation), the problem instance changes; this allows the agent population to experience a wide variety of training situations to learn robust behaviors. In each generation, each agent is assigned a fitness for generation g based on fuel spent and whether they arrived before the prescribed time limit:

$$U_{j,g} = -F_{tot} - H(g)L(j) \tag{5}$$

where the fitness of agent j in generation g is $U_{j,g}$, the fuel consumed on the journey is F_{tot}, $H(g)$ is the positive "handicap" assessed for arriving late and is a function of the generation, and $L(j)$ is calculated as:

$$L(j) = max(0, T_{arrive} - T_f) \tag{6}$$

which returns zero if package j was on time, or the measure of time the package was late.

In Equation 5, the handicap $H(g)$ is initially zero, such that the agent has an opportunity to learn the function between its actions and the fuel efficiency of the vehicle over the journey. $H(g)$ then steadily increases as g increases, so that arriving on time becomes a higher and higher priority. As g nears the final generation gN, the fuel efficiency term and on-time term achieve a rough parity in terms of importance. Changes in the size of the experimental domain would necessitate an adjustment of the $H(g)$ function to maintain this parity.

The agents learn to achieve high performance on this task through an evolutionary algorithm, which mimics the process of biological evolution; high-achieving agents are very likely to survive, and lower-achieving agents are very likely to be replaced. In this implementation, we maintain a population of 100 agents, and allow 20 agents to pass directly to the next generation. The best-performing member of the population is always maintained, and 19 additional agents are selected; for each of the 20 agents with the best fitness values, the agent is selected to survive with probability $(1-\epsilon)$, and a random agent is selected to survive with probability $\epsilon = 0.3$. The value of ϵ was chosen to encourage slower population convergence to avoid local optima.

The 20 surviving agents serve as parents for the 80 agents created for the next generation. Each new agent selects a parent agent from among the survivors, and after a step of mutation using a triangular distribution on each weight centered at zero, with maximum and minimum deviations of ± 0.05 in the neural network is, is entered into the population. This evolutionary algorithm process is outlined in Algorithm 3.

[1]75 was chosen as a "typical" rate of travel for this calculation, merely so that vehicles could "make up time" if necessary, at quadratic fuel cost. See Equation 3 for details.

4.3 Evolution in Bidder Source Domain

In a similar manner, we must form a simple source domain to train the bidding agents. In our first iteration of this, we created a source similar to the driver source domain, which instead placed multiple vehicles near a pickup location and allowed the bidding agents to create bids for each vehicle. This led to unacceptable performance: for any given delivery, bidding agents learned to bid *relatively* higher for better suited vehicles, but they did not learn to create an appropriate *absolute* bidding gradient. In fact, the agents trained on this BSD only utilize about 1% of the available bidding space, that is, $\beta_{max} - \beta_{min} \leq 0.01$. This bidding strategy did not generalize well into the FCP.

Instead of this approach, we applied domain knowledge to create a supervised learning problem. Distance, time, vehicle weight and road conditions between the vehicle's current location and the pickup location all have an impact on a vehicle's suitability for a delivery. We can combine these in a linear fashion to create a target bid equation to train the population of agents. We call this equation the Bidder Source Equation, detailed below:

Definition : Bidder Source Equation (BSE) We train agent i's bid for vehicle v (initially located at location λ_i) on package p_j using the equation:

$$\beta_{train} = 1 - k_1\delta_{(i,a)} - k_2 S_{(i,a)} - k_3\psi_j - k_4 w_v \tag{7}$$

where β_{train} is the bid, $\delta_{(i,a)}$ is the distance from the vehicle's current location to the package origin, $S_{(i,a)}$ is the average road slope between the vehicle's location and the package origin, ψ_j is a metric that characterizes the time available to make the delivery (see Section 4.2), and w_v is the weight of the vehicle. k_{1-4} are tunable parameters. Performance is not sensitive to these parameters, except that k_1 must be significantly larger than the rest.

With this equation as the source problem, we can create a random training instance and train directly on the error, in a case of supervised learning. That is, the fitness U_i of an agent i is:

$$U_i = -|\beta_i - \beta_{train}| \tag{8}$$

We allow each agent to evaluate the circumstances provided by the problem instance into a bid, and then use the same evolutionary algorithm outlined in Section 4.2 in Algorithm 3. By using the BSE, we attained bidder behaviors that used a much larger portion of the available bidding space ($\beta_{max} - \beta_{min} \approx 0.9$). This bidding range was much more appropriate to transfer into the FCP.

The justification for the form of Equation 7 is as follows. Each term in the linear combination expresses a simple fact:

for example the first term states that a vehicle closer is more suited for a delivery than one further away; the last states that a lighter vehicle is more suited than one that is heavier. The linear combination of these factors is the simplest possible form that expresses these facts. By allowing our bidders to train on this equation, we improve their performance over a random neural network. This equation does not seek to represent an ideal bidding formulation for the FCP, it merely acts as a starting point, from which the agents may deviate as the evolutionary process continues.

4.4 Coevolution in Target Domain

The agent populations, trained first in their source domains (the DSD and BSE), are then put into use in the target domain, the FCP. We choose to keep the two populations of agents separated, calculating their fitness with the same metric in each experiment, but allowing them to be evolved separately. By allowing this, and by randomly pairing the agents together in each problem instance together, we employ the concept of coevolution.

In the experiments conducted for this work, a set of 10 trucks is assigned to service 25 customers for a series of 1000 package delivery events. For each generation, we draw 10 agents from each population that have not yet been used in the current generation, pair them randomly, and assign the teams of agents associated with vehicle i fitness values based either on their local utility $\mathbb{L}_i$, or the global system utility $\mathbb{G}$. We calculate $\mathbb{L}_i$ as the negative sum of all fuel spent by that vehicle, plus a positive term for each package delivered on-time.

$$\mathbb{L}_i = \sum_{j=1}^{nP} I_i(j)[-F_{(tot,j)} - HL(j) + \phi] \tag{9}$$

where nP is the total number of packages in the problem instance, $I_i(j)$ is a function that indicates whether vehicle i was the maximum bid for package j, $F_{(tot,j)}$ is the total fuel expended delivering package j, H is the constant handicap coefficient for a late delivery, $L(j)$ is calculated by Equation 6, and ϕ is a constant "bonus" for making a delivery.

Because fuel costs are always a negative value, without the positive bonus for delivering a package, the agents quickly learn to bid low so that another agent might have to make the delivery, making the first agent earn zero fitness for that delivery event, while the other incurs the negative cost. This increases the fitness of the first agent at a cost to the system level performance. Conversely, if ϕ is set too high, all agents bid very high for every delivery, because the fuel costs incurred are dwarfed by the bonus received for completing the delivery.

We calculate the global system utility $\mathbb{G}$ as the sum of all local rewards, without the bonuses (ϕ):

$$\mathbb{G} = \sum_{i=1}^{nV} [\mathbb{L}_i] - (nP)\phi \tag{10}$$

At the system level we are concerned with the sum total of fuel spent; the package delivery bonuses merely add a constant to the global reward, which does not affect learning. Because all of the agents are scored on the overall fuel consumption, they learn not shy from taking a package that they are well-suited to deliver, making the bonus term unnecessary. The on-time delivery term remains, because this is a matter of concern to a truckload-hauling carrier: having a high on-time-percentage can be a selling point in attaining new contracts, and by changing the H term in Equation 9, we can potentially cause the agents to make many hard-time deliveries (high H) or many soft-time deliveries (H low). This $\mathbb{G}$ term will always evaluate to be negative; the agents strive to maximize the system utility, thus minimizing the fuel spent to make a delivery on time.

4.5 Approximation and Partial Transfer

In order to show that our approach to solving the FCP is not brittle to changes in experimental parameters or small changes in procedure, we choose to demonstrate results on two additional forms of complication. First, we pose a situation in which none of the drivers are trained in any source problem, while bidders are trained in the BSE. In the target problem, coevolution is allowed to proceed as normal.

Additionally, to demonstrate that a variety of function approximators could be suitable for this solution strategy, we replace the sigmoid function in the neural network we used with a gross approximation which requires far less computational power: a 3-piece linear function:

$$f(x) = \begin{cases} 0 & : x < -2 \\ 0.25x + 0.5 & : -2 \leq x \leq 2 \\ 1 & : x > 2 \end{cases} \tag{11}$$

The training of the network and weights remained the same, but for the entire training process from source to target, the piecewise linear function was used instead of the sigmoid. Note that though this function is not differentiable, we do not use a gradient-based approach in this work; evolution still functions using this approximation by mutating the weights associated with the networks and evaluating fitness.

5. RESULTS AND DISCUSSION

In this work, we used all 8 methods (detailed below) on a series of 30 randomly-generated instances of the FCP, with a fleet of 10 vehicles serving 25 customer locations, over a course of 1000 package deliveries. In training the bidders, we used values of $\{k_1, k_2, k_3, k_4\} = \{0.6, 0.1, 0.1, 0.2\}$. We created a timescale long enough that the package congestion would be low, keeping the problem instance difficulty on the low end of those available through the FCP, and allowing all deliveries to be completed while limiting vehicles to carrying out one delivery at a time. We chose to use different problem instances for our statistical trials to show the robustness in our methods and results. For each statistical run, the same problem instance was used for all methods test, and reported global system utility $\mathbb{G}$ was normalized with respect to the mean of the first generation of all trials, except full transfer learning. This was done to allow comparison across statistical trials, and to emphasize that the global system utility $\mathbb{G}$ is a *unitless* quantity that represents a combination of the fleet's fuel consumption and on-time performance (Equation 10). Error is reported as the standard deviation of the mean,[2] and in many cases is smaller than the symbols used to plot. To validate our distributed approach to the novel FCP presented in Section 3, we desired to compare the effect of using a local fitness evaluation ($U_i = \mathbb{L}_i$) and a global fitness calculation ($U_i = \mathbb{G}$) for

[2]The deviation of the mean for N statistical runs is calculated as $\frac{\sigma}{\sqrt{N}}$

coevolution in the FCP, and testing these baseline measurements against the use of full transfer learning, using both DSD and BSE source environments. Additionally, we wish to demonstrate that our approach is not brittle to lost information or approximations, and compare these results to the baseline cases, creating a set of eight experimental methods. We show results in the following scenarios:

1. No transfer learning with fitness calculated through local and global utilities (Section 5.2)
2. Full transfer learning with global and local fitness during coevolution (Section 5.2)
3. Partial transfer learning (Section 5.3)
4. Transfer learning using a linear approximation of a sigmoid (Section 5.4)

5.1 Learning From Scratch

First, as a validation of our approach and methodology for the FCP, we compared the results of training the agents in the target FCP from scratch, with no transfer learning, against the use of full transfer learning from both source problems, as outlined in Section 4. Figure 1 shows these results, which show a substantial gain in system performance over the training period, regardless of whether the local or global training signal was used. This is because of the strong coupling between the two calculations, as the global utility $\mathbb{G}$ is formulated as a sum of local rewards $\mathbb{L}_i$, with a constant term subtracted (Equation 10).

5.2 Full Transfer Learning

We note that the full transfer learning case shown in Figure 2 thoroughly outperforms learning from scratch, with initial performance that is within 10% of the best converged performance of any algorithm tested. It is important to note here that the computation time for the full transfer learning case is roughly the same as 5 additional generations of training time in the FCP (as the source problems are much simpler). Full transfer learning, using both the DSD and BSE source domains to train the driver and bidder agents,

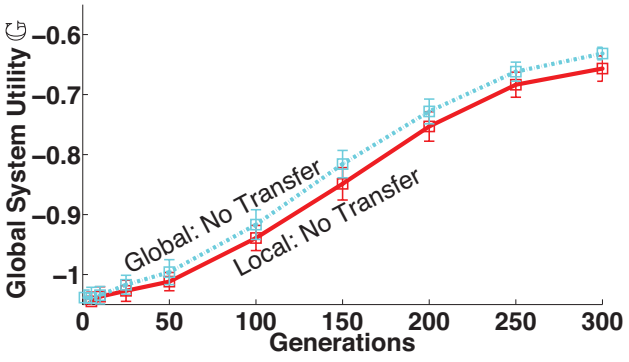


Figure 1: Comparison of agents evolved in the FCP using as their fitness evaluation: $\mathbb{G}$, the global system utility (upper, dotted line); $\mathbb{L}_i$ (lower, solid line). Note that in 300 generations, both fitness calculations lead to improvement in system performance.

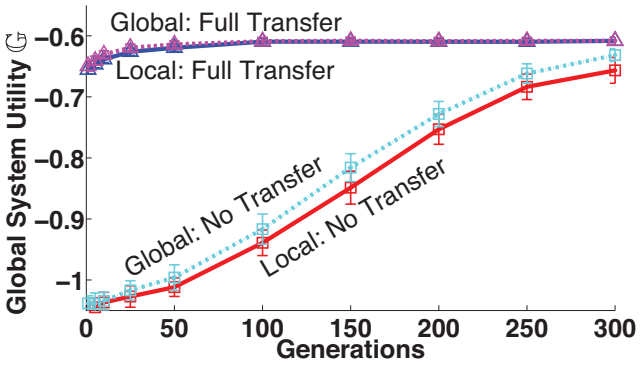


Figure 2: Performance in the FCP of agents coevolved solely in the FCP (lower lines with square markers) against agents evolved in separate source domains transferred into the FCP (upper lines with triangle markers). Note that the two full-transfer learning cases overlap significantly, and the boost in performance persists over 300 generations. Approximate computing time for the source problems in the full transfer cases was approximately 5 generations on this scale.

is extremely effective in the FCP, and results of comparable quality can be obtained in only 10% of the training time required, when compared to learning from scratch. All of these results hold true whether the local utility $\mathbb{L}_i$ or global utility $\mathbb{G}$ is used for the fitness evaluation calculation, as the resulting performance is nearly identical.

5.3 Partial Transfer Learning

We also show that our methodology in this work is robust to potential failures. First, we pose a scenario in which we allow the bidders to be trained on their source problem, but force the driving agents to learn from scratch in the FCP target environment: only one of the two agent populations benefits from transfer learning.

As seen in Figure 3 we still see some benefits both in initial performance (∼10%) and in learning speed in both the local and global reward cases. Because of the simpler mapping from actions taken to fitness calculation (the bidders are taking reasonable actions, instead of random ones as they would when learning from scratch), noise is effectively removed from the fitness calculation and the rate of performance increase is improved. The driving agents are able to learn to take reasonable actions before the bidding agents diverge from making good decisions because of the reward signals received in the FCP. It is important to note that in all of these cases, the driving agents and bidding agents are always receiving the same reward.

When we perform the same experiment in reverse, allowing the driving agents to use transfer learning, while forcing the bidding agents to start learning from scratch, we see performance that is almost at the level of full transfer learning: in the tested instances of the FCP, the drivers are capable of wasting far more fuel than poorly assigned bids. This is because the worst effect a poor bid can have is a vehicle traversing across the experimental domain, while in this formulation a driver is allowed to choose an excessive speed that wastes significantly more fuel. These results were omitted from Figure 3 for readability. In higher package-congestion

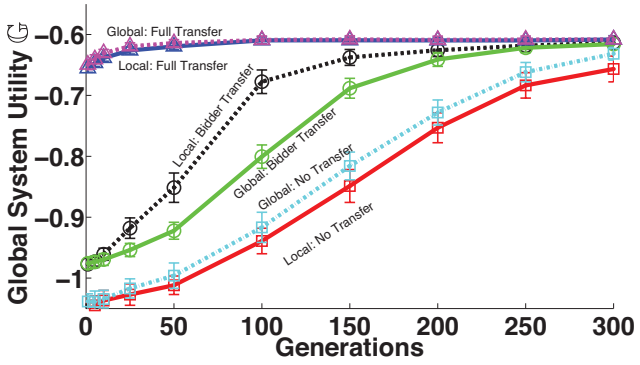


Figure 3: Experimental results in the FCP using three forms of transfer learning: no transfer (bottom, blue and red), full transfer (top, triangles), and bidder-only transfer learning (middle, green and black) which could correspond to an information loss scenario. Note that the full transfer learning cases overlap each other.

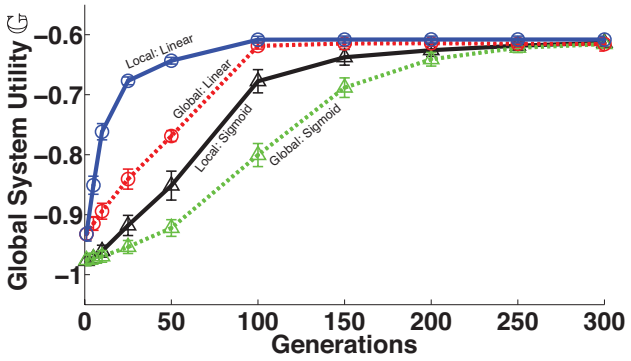


Figure 4: Experimental results in the FCP using bidder transfer learning and coevolving on the global system utility $\mathbb{G}$ or local agent utility $\mathbb{L}_i$. Because of the relatively simple problem instance, a neural network using a linear function as an approximation of the sigmoid itself actually outperformed the full-accuracy neural network.

cases or cases in which the system designer limits the speed at which the vehicles may travel to a greater degree, we expect that the bidding agents would have a stronger contribution to overall system performance.

5.4 Transfer Learning with Linear Approximations

Finally, we examine whether the neural network function approximation (which, with its many embedded sigmoids, can be very computationally expensive) is strictly necessary for our methods to work in the FCP domain. We replace the sigmoid function in each of these calculations with a 3-piece linear function with the same slope as the sigmoid at zero, seen in Equation 11.

In Figure 4, we show that using the linear approximation of the sigmoid in the FCP with bidder transfer learning actually leads to better system performance than using the neural network itself, converging to similar performance more than 100 generations faster. Here, it is important to note that the neural network using sigmoids itself is a function approximator, and the linear piecewise functions simplify this approximation.

Because the particular problem instance of the VRP shown is low-congestion enough, the linear function is able to embody all necessary information for treatment of this problem. In more congested problem instances, we expect that this would not be the case, and that the neural network using sigmoids would begin to outperform the linear approximation at some level of problem complexity or congestion.

6. CONCLUSION

In conclusion, in this work we have proposed the novel combination of VRP variations into the Fleet Coordination Problem domain. We proposed an adaptive solution strategy that leverages benefits from the fields of multiagent systems for decentralization, neural networks for function approximation, neuro-evolution for agent training, transfer learning for boosting initial performance and maintaining policy applicability over problem instances, and coevolution for simplifying the agent responsibilities and maintaining the ability for agents to retain their skills in addressing these different responsibilities in the FCP.

Though we trained on two simple source domains before placing agents in the FCP target domain, even after coevolving the agents on the combined FCP, we can still transfer this experience through a policy transfer, by maintaining the agents' policies and only changing the problem instance. Our experimental methods suggest that we can very easily take agent populations trained in one FCP and transfer their knowledge to an FCP of similar complexity with success, and work in the near future includes transferring agent experience from a simple FCP to a more complex, congested FCP instance. The FCP parameters for vehicles, packages, time, and customers in this work were chosen such that a suitable solution could definitely be found; problem difficulty can be increased by decreasing available resources to work with, or increasing demand. We expect that this "stair step" method of training agents may provide better performance in complex FCP instances than transferring straight from the original source problem to the final target problem, or learning from scratch. We seek to discover if there is a problem instance that cannot be successfully learned from scratch, which is achievable through the use of transfer learning.

Additionally, we wish to frame the FCP as a multi-objective problem, and incorporate multi-objective metrics into our treatment of the problem, including fitness shaping techniques to assist the agents in discerning their particular contribution to the system as a whole.

7. REFERENCES

[1] Amr Ahmed, Kai Yu, Wei Xu, Yihong Gong, and Eric Xing. Training hierarchical feed-forward visual recognition models using transfer learning from pseudo-tasks. *ECCV*, pages 69–82, 2008.

[2] Felix Ammah-Tagoe. Freight in america: 2006. Technical report, U.S. Department of Transportation, Washington D.C., January 2006.

[3] K. Savla Arsie, A. and E. Frazzoli. Efficient routing algorithms for multiple vehicles with no explicit

communications. *IEEE Transactions on Automatic Control*, 10(54):2302– 231, 2009.

[4] Steffen Bickel, Jasmina Bogojeska, Thomas Lengauer, and Tobias Scheffer. Multi-task learning for hiv therapy screening. *ICML*, pages 56–63, 2008.

[5] Rich Caruana. *Multitask Learning*. PhD thesis, Carnegie Mellon University, September 1997.

[6] Nicos Christofides. The vehicle routing problem. *Revue Francaise d'Automatique, d'Informatique et de Recherche Operationelle*, 10(2):55–70, February 1976.

[7] J Dethloff. Relation between vehicle routing problems: an insertion heuristic for the vehicle routing problem with simultaneous delivery and pick-up applied to the vehicle routing problem with backhauls. *Journal of the Operational Research Society*, 53:115–118, 2002.

[8] Richard Henry Distel and Richard Albert Distel. Apparatus to improve the aerodynamics, fuel economy, docking and handling of heavy trucks. Patent US 7,748,771 B2, Distel Tool and Machine Company, 2010.

[9] Miguel Andres Figliozzi. An iterative route construction and improvement algorithm for the vehicle routing problem with soft time windows. *Transportation Research Part C: Emerging Technologies*, 18(5):668–679, October 2010.

[10] Dario Guariento. System for completely closing the space between the cab and semi-trailer of an industrial or commercial vehicle, to improve the aerodynamics of the vehicle. Patent EP 1 870 320 A2, IVECO, 2007.

[11] Steven Michael Gutstein. *Transfer Learning Techniques for Deep Neural Nets*. PhD thesis, University of Texas at El Paso, May 2010.

[12] J Hine, A Barton, C Guojing, and W Wenlong. The scope for improving the efficiency of road freight transport in china. In *7th World Conference on Transport Research*, 1995.

[13] Charlotte Jacobs-Blecha and Marc Goetschalckx. The vehicle routing problem with backhauls: Properties and solution algorithms. Materials Handling Research Centre Technical Report MHRC-TR-88-13, Georgia Institute of Technology, Atlanta, 1993.

[14] Gilbert Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 1992.

[15] Thomas Léauté, Brammert Ottens, and Boi Faltings. Ensuring Privacy through Distributed Computation in Multiple-Depot Vehicle Routing Problems. In *Proceedings of the ECAI'10 Workshop on Artificial Intelligence and Logistics (AILog'10)*, 2010.

[16] Daimler Trucks NA. Personal communication.

[17] Gábor Nagy and Said Salhi. Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries. *European Journal of Operational Research*, 162:126–141, 2005.

[18] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, October 2010.

[19] Liviu Panait. Theoretical convergence guarantees for cooperative coevolutionary algorithms. *Evolutionary Computation*, 18(4):581–615, 2010.

[20] Liviu Panait and Sean Luke. Cooperative multi-agent learning: The state of the art. *Journal of Autonomous Agents and Multi-Agent Systems*, 11:387–434, 2005.

[21] Padmini Rajagopalan, Aditya Rawal, and Risto Miikkulainen. Emergence of competitive and cooperative behavior using coevolution. *GECCO*, pages 1073–1074, 2010.

[22] Daniel L. Silver and Robert E. Mercer. Sequential inductive transfer for coronary artery disease diagnosis. *International Joint Conference on Neural Networks*, 2007.

[23] Andrei Soeanu, Sujoy Ray, Mourad Debbabi, Jean Berger, Abdeslem Boukhtouta, and Ahmed Ghanmi. A decentralized heuristic for multi-depot split-delivery vehicle routing problem. *IEEE International Conference on Automation and Logistics*, 2011.

[24] Gwang-Geong Soung. Method and device for measuring slope of driving road. Patent 5,703,776, Hyundai Motor Company, Ltd., 1995.

[25] C.D. Tarantilis, C.T. Kiranoudis, and V.S. Vassiliadis. A threshold accepting metaheuristic for the heterogeneous fixed fleet vehicle routing problem. *European Journal of Operational Research*, 152:148–158, 2004.

[26] Sebastian Thrun. Is learning the n-th thing any easier than learning the first? *Advances in Neural Information Processing Systems*, pages 640–646, 1996.

[27] William Burley Uland. Under-vehicle aerodynamic efficiency improvement device. Patent Application Publication US 2005/0146161 A1, 2005.

[28] Yang-ByungPark and Sung-HunSong. Vehicle scheduling problems with time-varying speed. *Computers in Industrial Engineering*, 33(3-4):853–856, 1997.

Don't Go with the Ant Flow: Ant-inspired Traffic Routing in Urban Environments

Jörg Dallmeyer
Institute of Computer Science
Goethe University Frankfurt
P.O. Box 11 19 32
60054 Frankfurt am Main,
Germany
dallmeyer@cs.uni-frankfurt.de

René Schumann
Institute of Business
Information Systems
HES-SO
Rue de Technopôle 3
3960 Sierre, Switzerland
rene.schumann@hevs.ch

Andreas D. Lattner
Institute of Computer Science
Goethe University Frankfurt
P.O. Box 11 19 32
60054 Frankfurt am Main,
Germany
lattner@cs.uni-frankfurt.de

Ingo J. Timm
Computer Science
Department, Business
Informatics
University of Trier
54286 Trier
ingo.timm@uni-trier.de

ABSTRACT

Traffic routing is a well established optimization problem in traffic management. We address here dynamic routing problems where the load of roads is taken into account dynamically, aiming at the optimization of required travel times. We investigate ant-based algorithms that can handle dynamic routing problems, but suffer from negative emergent effects like road congestions. We propose an *inverse* ant-based routing algorithm to avoid these negative emergent effects. We evaluate our approach with the agent-based traffic simulation system MAINS^{2}IM. For evaluation, we use a synthetic and two real world scenarios. Evaluation results indicate that the proposed inverse ant-based routing can lead to a reduction of travel time.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search

General Terms

Algorithms, Management, Measurement, Experimentation

Keywords

Traffic simulation, Multiagent-Based Simulation (MABS), routing, Ant-inspired

1. INTRODUCTION

Traffic routing is a well established research and optimization problem in traffic management [6]. Most research has been done for static problems, i.e., settings where the problem structure does not change. In static problems the routing decision boils down to find the shortest path between the start and the goal point. Once a solution has been found for all routes the optimal ones can be used whenever needed. These algorithms typically are based on shortest path algorithms, like the well known A* algorithm.

The situation becomes more complex if we regard dynamic problems. In a dynamic problem, the problem structure changes while solving the problem. For routing decisions this implies that not the traveling distance has to be optimized but the traveling time [1]. Of course, a simple approach is to assume a fixed average speed that can be used for every road and to use this in the calculation of the weights of the graph. It has turned out that this simple approach often can help, but it also turned out not to be sufficient for roads which load changes in time [13, 19].

One approach to handle this limitation is to gather data to enrich the routing graph with time dependent traveling information. Based on acquired data the traveling speed on a road is extracted and can be used during route planning. In this approach, the dynamic problem is reformulated to a static one which has a larger complexity than the initial static one, as for each edge in the routing graph, time dependent traveling speed information is available. But it turned out that the data acquisition takes a lot of effort [13, 19].

Using current trends and technologies like car-2-car communication [14] and autonomic road transportation support systems [8] cars can be enabled to communicate with each other, and also with their environment. Therefore, each car can be seen as an autonomous entity, that has computing

abilities and is able to send and receive information from its current environment, and to use this information, e.g., for routing decisions.

Fortunately, ant algorithms can handle dynamic environments, so if a road is congested the following cars will take a different route if the road ahead of them is blocked. But due to the principle of following other ants in an ant-optimizing algorithm, these situations will emerge regularly, given a situation with heavy traffic, as it can be often found in urban areas. The emergence of congestions is a negative emergent behavior [28].

Heavy traffic is typical for urban areas nowadays. But we can also make another observation in these areas. There often exists a number of alternative routes, as well. In this paper, we investigate how to handle dynamic routing problems in urban areas, with high traffic and a number of alternative routes. The goal of our research is to avoid the negative effects of these emerging congestions while preserving the positive emergent behavior of ant algorithms for routing in dynamic environments. Therefore, we will change the internal reasoning of the cars to navigate to their destinations. For the evaluation of our approach we use simulation studies based on an agent-based traffic simulation system, called MAINS^{2}IM. In our evaluation, we are comparing our approach with existing alternative routing approaches.

The rest of the paper is structured as follows. In the next section we discuss related work especially from the fields of traffic simulation and ant-based routing algorithms. Then, in Section 3, we give a brief introduction into MAINS^{2}IM. Since we are focusing on routing problems, we will especially highlight routing methods of MAINS^{2}IM in Section 4. Our approach of adapting the ant algorithm to make it more suitable for urban areas, by avoiding negative emergent behavior, is described in Section 5. In the subsequent section we evaluate our approach in a series of simulation studies and discuss our results. Finally, we conclude and outline potential future research.

2. RELATED WORK

In the first part of this section, we give an introduction to related work in the field of traffic simulation. In the second part we discuss current research in the field of ant-based routing algorithms. Thereby, we focus in both subsections especially on agent-based approaches.

2.1 Traffic Simulation

The modeling of traffic is a well established field, ranging back to early work in the first half of the preceding century, e.g., [17]. Since then, different models and (later) traffic simulations have been proposed. The focuses of those range from the simulation of huge scenarios, e.g., the road traffic in Switzerland [27], using a cellular automaton based model proposed in [23], to the simulation of very small areas (e.g., [5]) with high fidelity traffic models like, e.g., [30].

Traffic simulation systems consist of models for road user behavior, as well as traffic demand models and routing methods. The road model is typically encoded in form of an annotated graph. The users' behavior is often described by their capabilities, their goal(s), and their behavior patterns, e.g., acceleration patterns. In this work, we focus on routing methods.

Gehrke and Wojtusiak present an approach for inductive learning of traffic predictions in relation to day, time and weather which leads to a higher traffic flow in an agent-based traffic simulation with agents using the predictions for route planing [16].

Vasirani and Ossowski present an agent-based approach for efficient allocation of road traffic network with help of an artificial market [29]. The approach could be shown to be efficient in a small scenario but has not yet been tested in a traffic simulation system.

A crucial point is the explicit modeling of decision making of simulated cars [2], respectively the reasoning in the simulated agents. Gawron presents an iterative algorithm for route plan optimization towards an equilibrium [15]. In a first simulation run, each simulated car chooses an optimal route. Before the next run, a portion of agents re-plan with help of the travel time knowledge gained from the previously used roads. This is done repeatedly, until an equilibrium condition is established. An analogous approach is used by Raney et al. for TRANSIMS [26]. The simulation has to be run for about fifty times in order to reach the desired state in their scenario. Thus, this approach is very time-consuming.

Bazzan and Klügl present an agent-based approach for dynamic re-planning [3]. When a car agent a perceives the occupancy of the next road on its route plan to be higher than τ, a re-routing mechanism enables a to drive around the potentially jammed area. The approach leads to a better overall performance. For simulation, frequently re-routing is time consuming. Another important aspect is at what time of the travel the re-planning has to be done. When the next road on the current route of a is jammed, a re-routing may have been much better, if done earlier.

2.2 Ant-based Routing Algorithms

As pointed out before, ant-based routing algorithms have been already investigated for traffic management [1, 4, 19]. In current work, researchers try to adapt the basic algorithm to avoid the negative emergent effects outlined before. In the following we discuss those approaches.

The approach by Alves et al. [1] is based on the equilibrium theory of traffic networks, and therefore grounded in game theory. In this approach, the cars are routed by a centralized traffic control management system. The central traffic management system collects information about the load dependent changes of the traveling times which are computed by an ant colony optimization method. Therefore, the ant-based routing is used to create information about the expected load of roads, to adapt expected traveling time information in the routing algorithm. In their approach, cars are controlled; they have no abilities to make decisions.

The need for decentralized decision making in dynamic routing problems has been pointed out by Narzt et al. [24]. Similar to our approach it is assumed that cars become smart entities that can communicate with each other and with their environment. Cars act as ants and leave a pheromone trace

along the way they travel. In the presented approach the cars use the intensity of the pheromone trace to infer the traffic density on this particular section of the road ahead. Thus, based on this pheromone information the cars can adjust their local planning model and adapt the weights of the edges, used for their routing algorithm, i.e., A*.

The idea of inferring the current traffic situation from the pheromone trace has also been suggested by Bedi et al. [4]. In their approach, Bedi et al. discuss an approach for picking the next edge to travel towards the destination. For calculating the probability for an edge they combine three factors: the traveling distance, the pheromone strength and a random factor. Based on their description, the usage of their approach focuses the individual routing problem for a specific road user, who specifies its starting and end point. With this information and the traffic infrastructure different routes are computed iteratively, trying to minimize the probability that the car will get stuck in a traffic jam.

As previous authors Krömer et al. [19] also point out the need for handling dynamic routing problems for traffic routing. They also point out that collecting real world data, can be a complex task, that often has limitations in the amount of data and the area covered. Krömer et al. present an approach to avoid the negative emergent behavior in their routing algorithm. They modify the original ant colony optimization algorithm slightly, by introducing a probability threshold. If a probability of taking an edge becomes larger than the threshold, e.g., because it has a high pheromone mark, the probability will be cut to the threshold. Thus, the probability cannot become larger than the threshold. By adding this threshold, the authors are able to gain the advantages of the ant-based routing, but also have a mean to avoid that the shortest path gets blocked due to heavy usage. In their experimentation they used realistic road infrastructure and car behavior patterns. As expected the original ant algorithm found the shortest path faster, i.e., it could converge faster to a solution, but traffic jams occurred. This effect could be softened using the ant-based routing with the probability threshold.

3. SIMULATION SYSTEM

The routing algorithms discussed in this paper have been integrated into the simulation system MAINS^{2}IM (Multi-modAl INnercity SIMulation). The simulation uses cartographic material from the *OpenStreetMap*[1] initiative in order to automatically generate a simulation graph, leading to an executable traffic simulation. The system is built on the base of the free geographical information system (GIS) toolkit *GeoTools*[2].

In order to set up a simulation, an *OpenStreetMap* (.osm) file is clipped into a user defined map section. The new file is split into logical GIS layers in relation to their type of geometry or for rendering purposes (e.g., landscape polygons, waterways, buildings, points, railways, routes and roads). In a first step, a basic graph data structure is calculated, which then is refined by several analysis and correction steps. The result of this transformation process is a graph with Edge-Informations (EI) representing roads and NodeInformations (NI) representing the connections between roads, taking into account urban traffic circumstances like, e.g., cross-walks, traffic lights, roundabouts, speed limits, numbers of lanes and priorities for the determination of the right of way.

MAINS^{2}IM provides microscopic traffic models for cars (passenger cars, trucks and buses), as well as bicycles and pedestrians. The models are discrete in time and continuous in space. One simulation iteration corresponds to one second real time.

The road users in the simulation system are modeled by simple reflex agents. Each driver-car-entity has its individual driving capabilities, e.g., different dallying behavior, acceleration, maximum velocity or rating for safety distances. When a situation occurs where multiple cars prevent each other from passing a crossing due to the right of way rules, the involved agents are able to abstain from their right of way and let another one pass the crossing. When a driver-car-entity has to wait for a certain time in front of a crossing, it may re-plan and use another route to its original destination. This is done via usage of the A* search algorithm with prohibition of the next road of the original route.

The simulation is written in Java and can be executed on a workstation computer. Detailed descriptions of the simulation system as well as case studies can be found in, e.g., [20, 9, 10] and on the corresponding website `www.mainsim.eu`.

This work deals with a modification of current routing methods for traffic simulations. Thus, the next section describes the current routing approaches in MAINS^{2}IM.

4. ROUTING METHODS

Currently, MAINS^{2}IM implements three different routing methods. Two of them can be used to identify a specific route from a starting point to a goal and one can be used to generate probabilistic routing behavior from a specific starting point. The following subsections 4.1 to 4.3 describe the approaches and subsection 4.4 discusses the presented approaches.

4.1 Precalculated Routes

In order to precalculate all possible routes in the simulation graph, an all-pairs shortest path problem has to be solved. A reasonable approach would be the Floyd-Warshall algorithm [7, pp. 629-635]. It solves the problem in $\mathcal{O}\left(|V|^3\right)$. A distance function $d\left(NI_a, NI_b\right)$ estimates the duration of travel on the edge EI between nodes NI_a and NI_b. The algorithm has one shortcoming for the problem of traffic routing: It is not able to take account of the *preceding* edge on a path. Consider the situation shown in figure 1.

The Floyd-Warshall algorithm is not capable of suppressing u-turns or turns with an over sized turning angle α, which may be unrealistic. Another method is the repeated calculation of the Dijkstra algorithm [11], once for each NI. Overall, this method also leads to computational complexity of $\mathcal{O}\left(|V|^3\right)$. During computation of the Dijkstra algorithm, our method not only stores the preceding NI on a way, but also the preceding EI and thus, overcomes the aforementioned problem.

[1] `http://www.openstreetmap.org`, accessed 03/02/12
[2] `http://www.geotools.org`, accessed 03/02/12

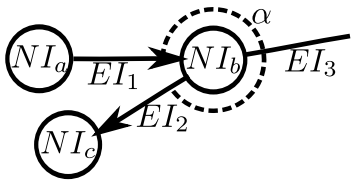


Figure 1: Consideration of turning angles

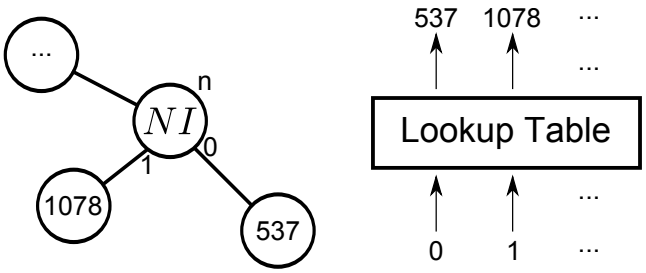


Figure 2: Lookup table for NI-ids

The computation of all ways in the simulation graph leads to the challenge to store them in the simulation system. Consider the simulation of a medium sized city with about 5,000 NIs, leading to about $5,000^2 = 25 \cdot 10^6$ paths. An efficient method in space and time is needed to store those paths. In MAINS^{2}IM a collection of all NIs (NodeInformationCollection (NIC)) is implemented in form of a list data structure. The ith element of NIC represents the NI with id i. This leads to a simple lookup when searching for a NI with a given ID. The IDs are stored as integer values.

Each NI stores a list of IDs. The ith entry of the list is the ID of the next NI_{next} node on the way to a given destination NI_{dest} with ID i. The time complexity to lookup a path in the graph is $\Theta(n)$, where n is the amount of EIs on the way.

The space complexity of this method is about $|NIC|^2 \cdot 32bit$ (integer) leading to about $95MB$ to store all paths. Coming back to figure 1, it is obvious that the amount of different IDs stored in the lists is small for each list. Each NI therefor stores a lookup table (LUT) for the IDs of its neighbors, as shown in figure 2.

The input of the table is the number of the neighbor and the result is the corresponding ID. The integer data type with the smallest amount of bits in Java is used: byte (8 bit). The approach leads to a compression ratio of about 25% without noticeable loss of time.

Note that the procedure is performed three times, because each basic type of road user (car, bicycle, pedestrian) has different routing characteristics and thus needs its own routes. Considering the previous example, the required space is reduced from about 286MB to about 72MB. The calculations are performed once and the entire graph data structure with its computed paths are stored in an external file.

The method of only using precalculated routes is suitable for static problem settings. But if multiple cars are simulated, the problem will become a dynamic one, as traveling times are changed due to other cars. Thus, the approach of precalculating routes leads to traffic jams in main streets of the road map, because the shortest paths use the fastest roads which are frequently selected by many road users. In order to obtain more heterogeneity in route choice, a probability based routing is introduced.

4.2 Probability-based Routing

In this approach, the routing of simulated road users is done probabilistically. Road users traveling under usage of this method, may have a defined starting position, but the destination of travel is not predetermined. Traffic of this kind is valuable for background traffic, influencing simulated road users with calculated routes from a defined source to a defined destination.

The approach proposed in this section is a refinement of the method stated in [9]. Let Ω_{NI} be the set of EIs, connected to NI. Each NI of the graph stores a turning probability function $p_t(EI_{\text{curr}}, EI_{\text{next}})$, for a road user of type

$$t \in \{\text{car}, \text{bicycle}, \text{pedestrian}\} \tag{1}$$

coming from EI_{curr}, giving the probability to choose EI_{next} as the next EI for travel. The function p_t holds equation 2 and 3.

$$p_t(EI_{\text{curr}}, EI_{\text{curr}}) = 0 \tag{2}$$

$$\sum_{EI \in \Omega_{NI}} p_t(EI_{\text{curr}}, EI) = 1 \;\forall\, t, EI_{\text{curr}} \tag{3}$$

With a given EI_{curr}, a route through the graph can be obtained via repeated random selection of the next EI from EI_{curr}. The function p_t is computed with help of the precalculated routes from subsection 4.1.

Each $NI \in NIC$ holds counters for all types t and each combination $EI_{\text{curr}}, EI_{\text{next}} \in \Omega_{NI}$, initialized with 0. The routes $\zeta(NI_{\text{start}}, NI_{\text{dest}})$ between all non-equal pairs of NI_{start} to NI_{dest} are identified as lists of NIs and EIs. Each route ζ is analyzed and at each $NI \in \zeta$ the corresponding counter for the connection between the current and the next EI is incremented. This is done for all types of road users. Afterwards, all counters are normalized, resulting in p_t with the conditions shown in equation 2 and 3.

The determination of p_t has a complexity of $\mathcal{O}(|NIC|^3)$, because of $\mathcal{O}(|NIC|^2)$ paths and a maximum path length of $\mathcal{O}(|NIC|)$. During simulation, the computation of a path with n NIs takes $\Theta(n)$.

Both described methods are not directly capable for respecting dynamic routing features. Thus, the next subsection discusses a method based on the well-known A* search algorithm.

4.3 A*-based Route Determination

The A* search algorithm is suited to solve the single-pair shortest path problem in graph data structures. Nevertheless, it takes more time to calculate a way using A* search online than by the other methods described above.

A* search works similar to the Dijkstra algorithm, but uses a few heuristics to speedup computation, without loss of accuracy. The assumed distance $d(NI_a, NI_b)$ may be adjusted arbitrarily during runtime of the simulation, enabling for dynamic[3] modification of the routing methods. Different kinds of simulated road users may favor different types of roads, e.g., one may prefer motorways, the other country roads. This feature brings more specific characteristic behavior to road user plans on the cost of computation time.

4.4 Discussion

We have outlined three different approaches for planning routes for simulated road users, that have already been implemented in the MAINS^{2}IM system. The first assumes static travel times for each EI and leads to traffic overloads on major roads. The second method is a more dynamic approach, but without exact steering capabilities, although the p_t may be adjusted using a point of attraction, as shown in [9]. The third approach is appropriate for dynamic routing with exact start and destination points, but will be noticeable slower in computation time in large scenarios with thousands of nodes in the graph.

The next section discusses a modification method for the edge weight function $d(NI_a, NI_b)$ in order to overcome the overloading problems of the precalculated routes.

5. ANT-INSPIRED ROUTING

The idea of ant-colony optimization is to mimic strategies observed from real ants. Ants leave behind trails of pheromones, e.g., when looking for food. When an ant has to decide, which way to choose, it chooses the way with the higher pheromone concentration more likely than the others. These pheromones have a given, typical linear vaporization rate, i.e., the trace becomes weaker in time. This leads to the emergent effect, that short routes are found in the environment. A survey is provided by Dorigo and Blum [12]. It is obvious that this algorithm can be directly applied for routing decisions, e.g., in traffic simulation, as done here.

Our approach uses a simplified *inverse* ant colony optimization. It is inverse in the sense that a strong pheromone trace will influence following cars *not* to follow their predecessors but instead to avoid this road, taking a different route to their goals.

The basic idea of this ant-inspired routing is that each road holds a "smell intensity" si. The pheromone trace is used to indicate the previous usage of a road. The higher si, the slower a car will be able to drive on the corresponding road, since the usage of the road is higher, which will slow down the traffic on this road. Each EI holds two values EI^a_{si} and EI^b_{si} for the two possible directions of travel on EI (with direction a: In the course of road, direction b: contrary).

Let $length(EI)$ be the length of the corresponding road in m and EI_{vMax} the speed limit on the road. The distance estimation function $d(NI_a, NI_b)$ between two nodes NI_a and NI_b over EI in direction dir is adjusted by a modification of the estimated travel velocity on EI, as shown in the following equation 4.

$$\begin{aligned} d(NI_a, NI_b) &= \frac{length(EI)}{v^*} \\ v^* &= \left(1 - EI^{dir}_{si}\right) \cdot EI_{vMax} \end{aligned} \tag{4}$$

The domain of EI_{si} is $[0 \cdots 1]$, leading to a maximal estimated travel velocity v^* of EI_{vMax} when there is "no smell" and a minimal $v^* = 0$, when there is a high "smell intensity".

In each simulation iteration, each EI has to adjust its values of EI_{si}, as shown in equation 5.

$$EI^{dir}_{si} = \min\left(\max\left(EI^{dir}_{si} - \kappa + \vartheta \cdot dens\left(EI^{dir}\right), 0\right), 1\right) \tag{5}$$

The value of EI^{dir}_{si} is absorbed by the subtrahend κ and increased by the traffic density in the direction dir on EI, scaled by ϑ. The result is bounded to the interval $[0 \cdots 1]$.

A road with high traffic densities results in high concentrations of EI_{si} and thus influences cars to avoid the travel over EI when calculating a route with help of the A* search algorithm. This should result in more uniformly distributed traffic loads upon the road graph and thus shorter times of travel, because of less intense traffic jams. We expect that this approach is especially useful in areas with a number of different routing alternatives, and with a high traffic density. Therefore, we consider this approach in particular useful for urban areas, in which routing becomes especially important.

We believe that this approach can be valuable if the "pheromone traces" can be stored in the environment, e.g., by the underlying traffic management system. Cars that pass the roads can then read and update the pheromone traces.

6. EVALUATION

The evaluation of the routing methods described above begins with an optimization of the parameters κ and ϑ. Afterwards, the ant-inspired routing method is evaluated on a synthetic graph and on real world cartographic material.

6.1 Parameter Optimization

The optimization of κ and ϑ is done with the method Simulated Annealing (see, e.g., [22]). We use Simulated Annealing, because of its good suitability for complex problems and its ability to avoid being stuck in local optima. The fitness of a parameter configuration is determined in the urban scenario shown in figure 3.

The performance of each parameter configuration is estimated by the average fitness of five replications. The amount of simulated cars is held constantly at 400. Whenever a car reaches its destination, a new one will be generated. Source and destination points are chosen randomly. After a settlement phase of 900 simulation iterations, a measurement phase of 3600 counts the amount of cars fit_r, that have finished their travel in replication r.

The fitness of a setting is $fit = -\frac{1}{5}\sum_{r=1}^{5} fit_r$ (negation as the optimization problem is formulated as a minimization problem). This fitness function is an implicit estimation of the driving velocities of simulated cars, because the higher

[3] *Dynamic* refers to the dynamic change of the distance estimation function and not to dynamic replanning.

Figure 3: Town of Erlensee (13,000 inhabitants), cumulated length of roads: 142km, graph consists of 937 EIs and 714 NIs.

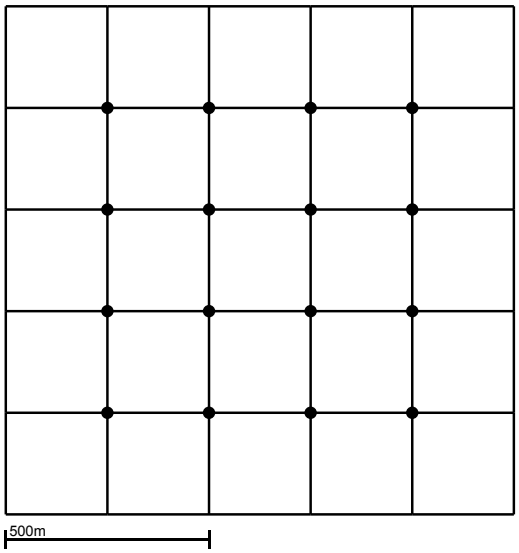


Figure 4: Synthetic graph. Circles show the positions of traffic lights.

the amount of finished cars, the more efficient the routing and the higher the average driving velocity.

The best parameter configuration with $fit = -4660.2$ in the described experiment was:

$$\kappa = 0.28077122867684745 \quad (6)$$

$$\vartheta = 2.8138856401002688 \quad (7)$$

6.2 Comparison of Routing Methods

The determined parameter configuration is used for a comparison of the ant-inspired approach with the method of A* search without enhancements and an iterative plan optimization approach.

The method of iterative route planning performs the simulation of identical cars several times and an amount of 10% of cars/agents is allowed to adjust its routing in each run, according to the travel times the agents have experienced in the preceding simulation runs. The method leads to a dynamic user equilibrium [15, 26], also discussed in Section 2. The training phase for this method is set to 50 replications, in order to enable the simulated cars to gain simple knowledge about the traffic conditions in the simulation area. The following experiment uses the iterative route planning approach for comparison. The first experiment for comparison is done in a synthetical scenario, followd by two experiments on real world road maps.

6.2.1 Synthetical Graph

For a first test of the obtained values, a highly dynamical experiment is used: A square lattice graph with 6×6 NIs. Each EI has a length of 250m and $EI_{vMax} = 13,\overline{8}m \cdot s^{-1}$.

Each NI with four EIs holds a traffic light, as shown in figure 4.

For evaluation, we generate 100 different settings with randomly generated start and goal positions for road users. Due to the stochastic nature of the used behavioral model, each setting is repeated with ten replications.

A car which enters the simulation, starts with velocity 0 $m \cdot s^{-1}$. It waits for a sufficient gap in traffic and then literally enters the road and begins acceleration. In the beginning, every simulated car stands still.

The result of each run is the average travel time of all cars, that have reached the destination after the settlement phase of 900 iterations. The measurement phase has a duration of 3,600 iterations and is extended until the last car has reached its destination.

Due to the stochastic nature of the used simulation model, the result of a setting is the average value of the results from its replications. The amount of cars is held constantly for the first replication and identical copies of the cars with the same start and goal positions are used for further replications with other seed values for the random number generator.

The average travel times per run $\bar{t}_r$ are compared. For the amounts of cars (200, 300, 400), the ant-inspired approach leads to significant[4] reductions in travel time in comparison to the A* method and exhibits lower spreadings. The values for $\bar{t}$ increase with increasing amounts of cars. In the experiment with 500 cars, the ant-inspired approach suffers from high-value outliers for $\bar{t}$, even though it still leads to the lowest travel times for 49 out of 100 runs. This indicates an problem at very high traffic densities, potentially

[4]The statistical software R [25] is used for determination of significance with help of the t-test using error level $\alpha = 0.05$.

$\#cars$	$\#runs$	A*	ant	iterative
200	100	$\bar{t} = 281.09$ $\sigma = 3.77$	$\bar{t} = 224.23$ $\sigma = 2.63$	$\bar{t} = 282.97$ $\sigma = 3.95$
300	100	$\bar{t} = 294.55$ $\sigma = 3.39$	$\bar{t} = 226.38$ $\sigma = 2.52$	$\bar{t} = 297.09$ $\sigma = 3.84$
400	100	$\bar{t} = 305.91$ $\sigma = 2.75$	$\bar{t} = 230.47$ $\sigma = 2.11$	$\bar{t} = 307.21$ $\sigma = 2.67$
500	100	$\bar{t} = 326.01$ $\sigma = 3.58$	$\bar{t} = 327.58$ $\sigma = 45.19$	$\bar{t} = 316.56$ $\sigma = 3.35$

Table 1: Average travel times $\bar{t}$ and standard deviation σ in the synthetical scenario.

$\#cars$	$\#runs$	A*	ant	iterative
200	100	$\bar{t} = 302.55$ $\sigma = 5.28$	$\bar{t} = 279.60$ $\sigma = 3.59$	$\bar{t} = 268.62$ $\sigma = 3.23$
300	100	$\bar{t} = 459.77$ $\sigma = 61.79$	$\bar{t} = 287.82$ $\sigma = 4.10$	$\bar{t} = 273.16$ $\sigma = 3.11$
400	100	$\bar{t} = 656.43$ $\sigma = 32.62$	$\bar{t} = 291.70$ $\sigma = 4.48$	$\bar{t} = 276.09$ $\sigma = 3.72$
500	100	$\bar{t} = 807.42$ $\sigma = 24.48$	$\bar{t} = 310.54$ $\sigma = 6.00$	$\bar{t} = 283.63$ $\sigma = 4.92$

Table 2: Average travel times $\bar{t}$ and standard deviation σ in graph for road map of Erlensee.

an oversteering of ϑ, leading to a total blockade of certain roads.

6.2.2 Small Real World Scenario

As a next step, we repeat the described experiments in real world scenarios. 100 start-goal settings with ten replications per run are performed in the map excerpt, shown in figure 3. The different routing methods are compared with identical start-goal settings for the cars. The results are shown in figure 5.

The plots show that both, the ant-inspired routing and the iterative route planning mechanism lead to significant lower mean travel times for all simulated traffic densities. The iterative approach results in shorter travel times than the remaining approaches. Both, the iterative and the ant-inspired approach reduce traffic jams at high traffic densitites, because of agent experiences on the one hand and bad weighting of jammed areas during route planning on the other hand. Table 2 summarizes the results of this experiment.

In order to determine the effects of the different routing methods, the results of one run are used for calculation of a road usage map. In each simulation time step, each car increments the road usage value for the EI it currently drives on. Figure 6 shows the comparison of the different routing methods on the map extract of figure 3. The first row (parts (a) to (c)) takes road usage values for all three approaches for determination of the minimum and maximum values in order to show an overall comparison. The second row (parts (d) to (f)) scales the values for each separate approach in order to deliver an insight on the distribution of traffic for the individual approaches.

$\#cars$	$\#runs$	A*	ant	iterative
500	50	$\bar{t} = 540.12$ $\sigma = 4.55$	$\bar{t} = 554.38$ $\sigma = 4.25$	$\bar{t} = 560.13$ $\sigma = 3.62$
750	50	$\bar{t} = 651.45$ $\sigma = 10.46$	$\bar{t} = 658.24$ $\sigma = 16.37$	$\bar{t} = 611.66$ $\sigma = 6.87$
1000	50	$\bar{t} = 800.22$ $\sigma = 32.62$	$\bar{t} = 809.08$ $\sigma = 4.48$	$\bar{t} = 683.88$ $\sigma = 3.72$
1500	50	$\bar{t} = 1109.39$ $\sigma = 14.76$	$\bar{t} = 1101.61$ $\sigma = 22.05$	$\bar{t} = 948.37$ $\sigma = 20.91$

Table 3: Average travel times $\bar{t}$ and standard deviation σ in graph for road map of Hanau.

Parts (a) to (c) of figure 6 show, that the maximum of road usage is dominated by the A* method and the ant-inspired and iterative approach produce less intensive traffic intensities in these areas. Parts (e) and (f) show that the ant-inspired and the iterative methods lead to a wider absolute spreading of traffic in comparison to the pure A* based routing, shown in part (d). Again this goes in line with our expectations, that the inverse ant-based algorithm can avoid the negative emergent behavior of road congestions, since the traffic is more balanced on the different routes of the road network. This effect gets facilitated if multiple alternative routes exist.

The comparison of the different methods took place in the same graph, the optimization was done in section 6.1. The next step is to take the methods to another road map in order to test for overfitting of parameter optimization.

6.2.3 Medium-sized City

The medium sized city Hanau am Main (89,000 inhabitants) with a total length of roads 548km is used for a second experiment. The resulting graph has 4,201 NIs and 5,758 EIs. The experimental setup remains identically to the preceding experiment, except that on the one hand, the amount of simulated cars is increased. On the other hand, the number of start-goal settings per traffic density is reduced to 50 due to the increased computational complexity of this scenario. Table 3 shows the experimental results.

Table 3 exhibits differences to the results of Table 2. The iterative routing mechanism leads to the lowest values of $\bar{t}$ at high traffic densities. The ant-inspired routing approach is not beneficial for this scenario. It performs comparable to the basic A* mechanism. This could be an indicator for parameter overfitting for the parameters κ and ϑ to the road map, shown in figure 3.

6.3 Discussion

The evaluation has shown that the ant-inspired routing method can lead to lower average travel times of simulated cars in a synthetic scenario (section 6.2.1), as well as in a real world simulation graph (section 6.2.2). Section 6.2.3 did not show advantages of the ant-inspired method over the A* based or the iterative routing approach. However, the iterative approach has been integrated as reference only

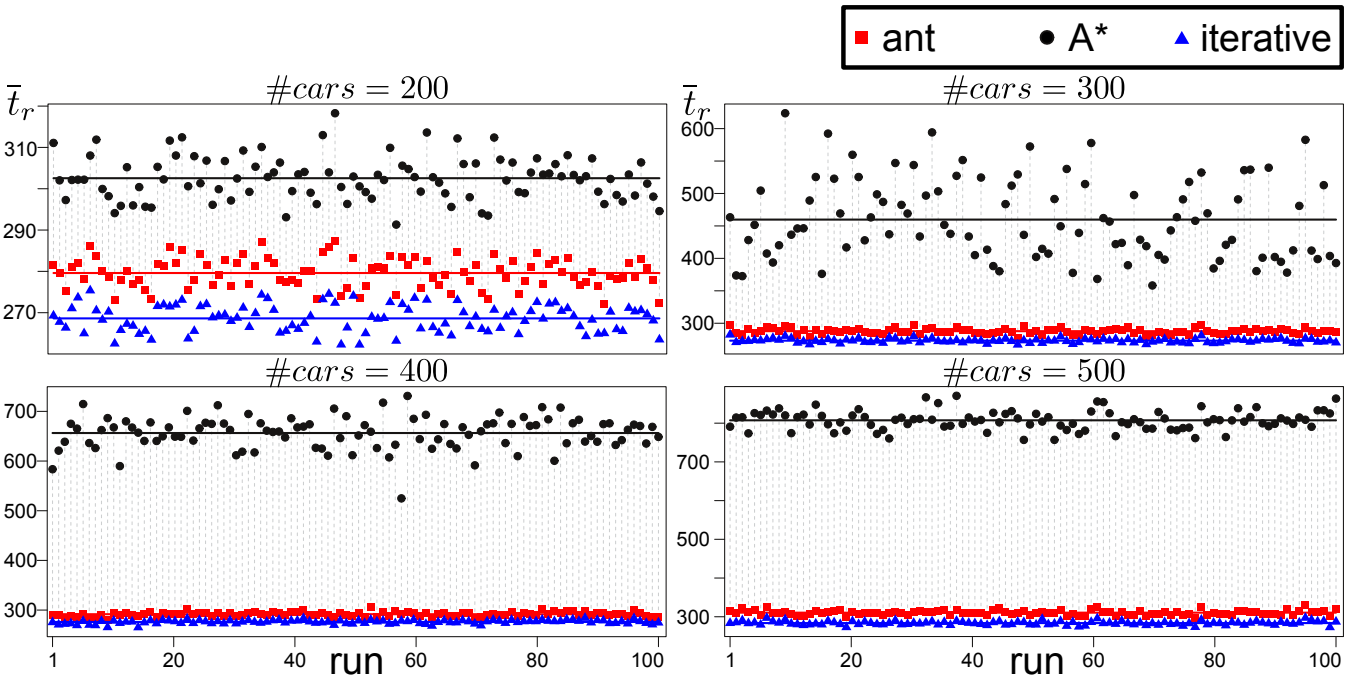


Figure 5: Evaluation on a graph for the road map of Erlensee. Horizontal lines show average values.

and not for direct comparison as it is based on individual "knowledge" of agents.

The ant-inspired method leads to a wider distribution of traffic in the simulation area. This is the basic outcome of the iterative approach. The ant-inspired routing approach has the advantage that a distribution of traffic is done without calibration runs. The amount of about 50 calibration runs, before performing the actual simulation runs is expensive and can be avoided by the approach presented in this paper. Nevertheless, it has to be mentioned, that the calibration with 50 runs is not always necessary. It would be better to do the calibration until the agents plans do not change any more, as discussed in literature.

In contrast to the approach, presented by Narzt et al. [24] our approach is not explicitly considering re-planning during travel. This is due to the focus of MAINS^{2}IM, namely the simulation of whole cities, taking account of multi modal traffic. For such large-scale scenarios, the simulation agents must not be very complex with respect to computational effort.

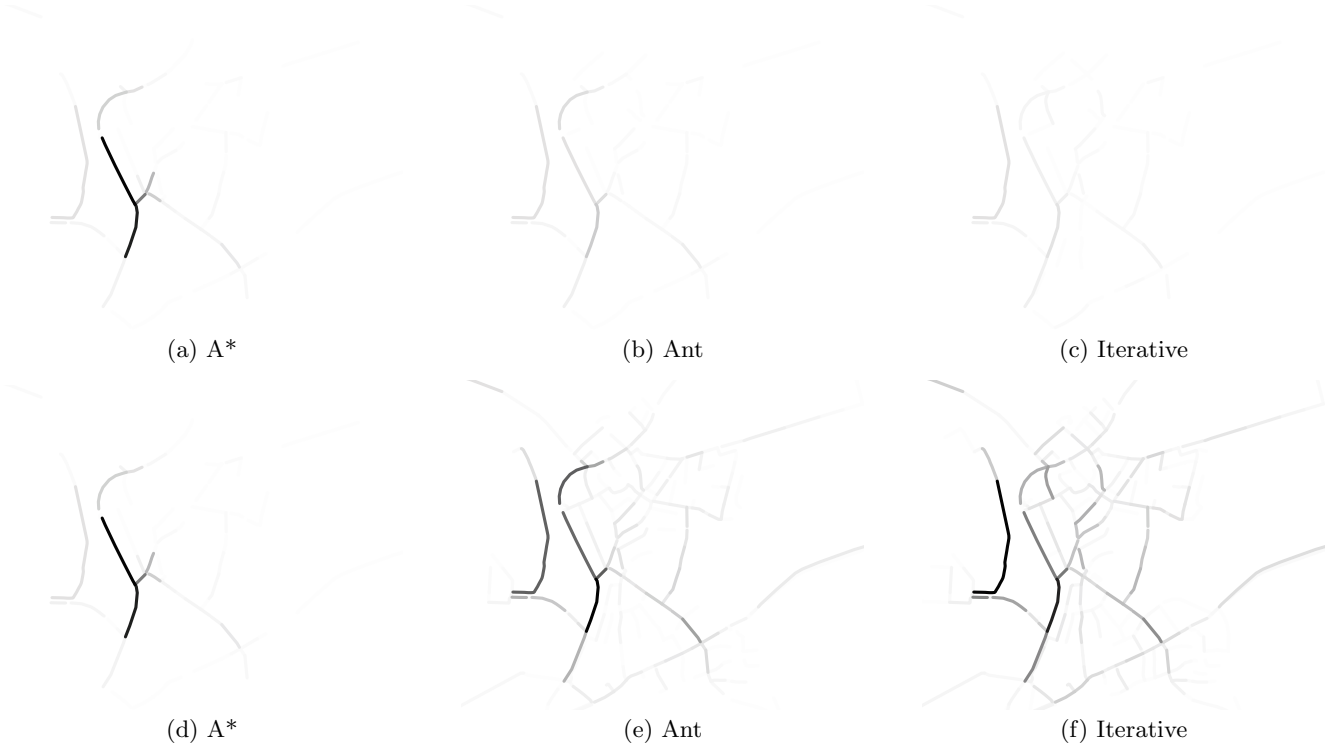
Our approach utilizes the effect that traffic jams resolve faster, when the amount of cars filling the waiting line behind the beginning of the jam decreases [21]. This is achieved by the ant-inspired routing mechanism as newly planned routes avoid road segments with high traffic.

7. SUMMARY AND PERSPECTIVES

In this paper, we have discussed the need for future routing approaches, especially in urban environments where a number of alternative routes exist and traffic is dense. In these environments a dynamic routing algorithm could help to reduce travel times. We have focused on routing approaches using the ant-optimization paradigm. While ant-based optimization algorithms offer a number of interesting features, like fast convergence to the shortest path, and self-stabilization in case of disruptions, there are some negative emergent effects. In particular this kind of optimization will produce congestions, as ants follow the most intense pheromone trail, which leads to overload situations on the road network.

To maintain the aforementioned positive effects of ant-based routing algorithms we have modified the conventional ant-based optimization approach, to have a balancing effect in the road selection. This is done with help of a simple modification of the distance function of the A* search algorithm. We have tested our modified routing algorithm in the agent-based traffic simulation system MAINS^{2}IM. Within this simulation system we could show the positive effects of our modification in a grid network. Since MAINS^{2}IM is able to generate a simulation model out of publicly available map information, we are able to transfer these results from a theoretical setting into settings based on real world map excerpts. The method's parameters have been optimized for a small scenario and benefits for another scenario have not been observed. This could indicate parameter overfitting and needs to be investigated in the future.

By using an agent-based traffic simulation system, we are able to model each road user, with a different set of attributes, leading to different characteristics, which allows us to investigate richer models. In our particular research we have applied local reasoning capabilities that cars can have in taking the routing decisions locally. This goes in line with the trend that cars become smart active entities in the



(a) A* (b) Ant (c) Iterative

(d) A* (e) Ant (f) Iterative

Figure 6: Comparison of road usage ratios (white: low usage; black: high usage). First row: Same leveling for all methods, second row: individual leveling for each method.

overall traffic management infrastructure.

The smell intensity values of the simulated roads are dependent on the time dependent traffic densities of the corresponding roads. The transferability from density to flow needs to be investigated in the future, because it is easier to measure traffic flow than traffic densities in real traffic scenarios. One aspect for future work could be to identify a set of good measurement points to assess the traffic situation. While the simulation allows us to have an easy access to the overall traffic situation, this is not possible for real traffic management systems. Those systems often have only a limited view, defined by a set of measuring and monitoring points. Based on this local view a global view has to be estimated. Therefore, the identification of measuring points becomes of special interest, to get a good estimation of the global state by local observations.

The described approach has not lead to benefits in a medium sized city. This may have two reasons: overfitting of the parameters for the small scenario or too long-ranging agent plans. If overfitting was the actual reason, the approach would not necessarily lead to advantages over the iterative routing approach with respect to computational time as situation-dependent calibration would also be needed. All mentioned routing approaches base on the offline A* search algorithm. Future research needs to investigate its use for real-time path finding algorithms like RTA* [18] in order to use the dynamics of the pheromone concentrations during the trips of the simulated cars for a dynamic routing with replanning. This should lead to better results in huge scenarios.

The ant-based routing can be studied in the simulation setting to investigate its potentials. As we have outlined before it bases on assumptions about ongoing technology trends, and therefore could be used also for travel time optimization in real world scenarios. The method needs to be investigated for robustness against influences from pedestrians and bicycles and local public transport, in the future. The effect on gas consumption and CO_2 emissions can be identified with MAINS^{2}IM and thus, will be a field of study for further investigations of the ant-based approach.

8. REFERENCES

[1] D. Alves, J. v. Ast, Z. Cong, B. D. Schutter, and R. Babuska. Ant colony optimization for traffic dispersion routing. In *Proceedings of the 13th International IEEE Conference on Intelligent Transportation Systems (ITSC 2010)*, pages 683 – 688. IEEE Press, 2010.

[2] M. Balmer. *Travel demand modeling for multi-agent traffic simulations: Algorithms and systems.* Dissertation, ETH Zürich, Switzerland, 2007.

[3] A. L. Bazzan and F. Klügl. Re-routing agents in an abstract traffic scenario. In *Proceedings of the 19th Brazilian Symposium on Artificial Intelligence: Advances in Artificial Intelligence*, SBIA '08, pages 63–72, Berlin, Heidelberg, 2008. Springer-Verlag.

[4] P. Bedi, N. Mediratta, S. Dhand, R. Sharma, and

A. Singhal. Avoiding traffic jam using ant colony optimization - a novel approach. In *International Conference on Computational Intelligence and Multimedia Applications, 2007.*, volume 1, pages 61 – 67, 2007.

[5] C. Bönisch and T. Kretz. Simulation of Pedestrians Crossing a Street. *Proceedings of the Eighth International Conference on Traffic and Granular Flow*, 8, Nov. 2009.

[6] K. J. Button and D. A. Hensher. *Handbook of Transport Systems and Traffic Control*, volume 3. Pergamon Press, 2001. ISBN 978-0-080-43595-4.

[7] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 2nd revised edition, September 2001. ISBN 0-262-03293-7.

[8] E. COST. Towards autonomic road transport support systems, 2011. `http://www.cost.esf.org/domains\_actions/tud/Actions/TU1102?`; Accessed: 02/27/12.

[9] J. Dallmeyer, A. D. Lattner, and I. J. Timm. From GIS to Mixed Traffic Simulation in Urban Scenarios. In J. Liu, F. Quaglia, S. Eidenbenz, and S. Gilmore, editors, *4th International ICST Conference on Simulation Tools and Techniques, SIMUTools '11, Barcelona, Spain, March 22 - 24, 2011*, pages 134–143. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), Brüssel, 2011. ISBN 978-1-936968-00-8.

[10] J. Dallmeyer, A. D. Lattner, and I. J. Timm. *Data Mining for Geoinformatics: Methods and Applications*, chapter GIS-based Traffic Simulation using OSM. Springer, 2012. (accepted).

[11] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959. 10.1007/BF01386390.

[12] M. Dorigo and C. Blum. Ant colony optimization theory: A survey. *Theoretical Computer Science*, 344(2-3):243–278, 2005.

[13] J. Ehmke and D. Mattfeld. Integration of information and optimization models for vehicle routing in urban areas,. In *The State of the Art in the European Quantitative Oriented Transportation and Logistics Research – 14th Euro Working Group on Transportation & 26th Mini Euro Conference & 1st European Scientific Conference on Air Transport.*, volume 20, pages 110 – 119. Procedia - Social and Behavioral Sciences., 2011.

[14] S. Eichler, J. Eberspächer, and C. Schroth. Car-to-car communication. In *VDE-Kongress 2006*, 2006.

[15] C. Gawron. An Iterative Algorithm to Determine the Dynamic User Equilibrium in a Traffic Simulation Model. *International Journal of Modern Physics C (IJMPC)*, 9(3):393–408, 1998.

[16] J. D. Gehrke and J. Wojtusiak. Traffic prediction for agent route planning. In *Proceedings of the 8th international conference on Computational Science, Part III*, ICCS '08, pages 692–701, Berlin, Heidelberg, 2008. Springer-Verlag.

[17] B. D. Greenshield. A Study of Traffic Capacity. *Proceedings of the 14th Annual Meeting Highway Research Board 14*, 468:448–477, 1935.

[18] R. E. Korf. Real-time heuristic search. *Artificial Intelligence*, 42(2–3):189–211, 1990.

[19] P. Krömer, J. Martinovic, M. Radecký, R. Tomis, and V. Snásel. Ant colony inspired algorithm for adaptive traffic routing. In *Proceedings of the Third World Congress on Nature and Biologically Inspired Computing (NaBIC 2011)*, pages 329 – 334. IEEE Press, 2011.

[20] A. D. Lattner, J. Dallmeyer, and I. J. Timm. Learning Dynamic Adaptation Strategies in Agent-based Traffic Simulation Experiments. In *Ninth German Conference on Multi-Agent System Technologies (MATES 2011)*, pages 77–88. Springer: Berlin, LNCS 6973, Klügl, F.; Ossowski, S., 2011. ISBN 978-3-642-24602-9.

[21] H. K. Lee and B. J. Kim. Dissolution of traffic jam via additional local interactions. *Physica A: Statistical Mechanics and its Applications*, 390(23-24):4555 – 4561, 2011.

[22] K. Y. Lee and M. A. El-Sharkawi, editors. *Modern Heuristic Optimization Techniques With Applications To Power Systems*. John Wiley & Sons, 2005. ISBN 978-0-471-45711-4.

[23] K. Nagel and M. Schreckenberg. A cellular automaton model for freeway traffic. In *Journal de Physique I*, volume 2, pages 2221–2229, December 1992.

[24] W. Narzt, U. Wilflingseder, G. Pomberger, D. Kolb, and H. Hörtner. Self-organising congestion evasion strategies using ant-based pheromones. *Intelligent Transport Systems, IET*, 4(1):93–102, march 2010.

[25] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2011. ISBN 3-900051-07-0.

[26] B. Raney and K. Nagel. Iterative Route Planning for Modular Transportation Simulation. In *in Proceedings of the Swiss Transport Research Conference, Monte Verita*, 2002.

[27] B. Raney, A. Voellmy, N. Cetin, M. Vrtic, and K. Nagel. *Towards a Microscopic Traffic Simulation of All of Switzerland*, pages 371–380. Computational Science - ICCS 2002 Part I, Lecture Notes in Computer Science 2329, Springer Heidelberg, 2002.

[28] I. J. Timm. *Dynamisches Konfliktmanagement als Verhaltenssteuerung Intelligenter Agenten*. DISKI. Akademische Verlagsgesellschaft Aka, Berlin, 2004.

[29] M. Vasirani and S. Ossowski. An Artificial Market for Efficient Allocation of Road Transport Networks. In *Ninth German Conference on Multi-Agent System Technologies (MATES 2011)*, pages 189–196. Springer: Berlin, LNCS 6973, Klügl, F.; Ossowski, S., 2011.

[30] R. Wiedemann. Simulation des Straßenverkehrsflusses. *Schriftenreihe des IfV*, 8, 1974.

Acknowledgement

This work was made possible by the *MainCampus* scholarship of the *Stiftung Polytechnische Gesellschaft Frankfurt am Main*.

Communication and metrics in agent convoy organization

Vincent Baines, Julian Padget
Dept. of Computer Science,
University of Bath
{v.f.baines,j.a.padget}@bath.ac.uk

ABSTRACT

Convoy formation, maintenance and dissolution is a multi-faceted problem, with different domains creating a range of constraints, some of which can be helpful, but equally that can complicate the situation. The scenario under consideration here is, in the long-term, a mix of human- and agent-controlled vehicles in a public, transportation setting. However, the focus here is on agent-controlled vehicles and the problem of "knowledge fusion", or more precisely (i) how much/how little, and (ii) what kind of inter-vehicle communication is sufficient to enable adequate individual and group situational awareness to permit the effective operation of a convoy. This can be viewed as a global problem, but it is also a local problem, as each convoy participant must weigh up the costs and benefits arising from (i) the loss of autonomy – being subject to the governance of the convoy – and, (ii) the loss of privacy – needing to communicate data and intentions to some other convoy participants. We report on the first steps, examining communication issues and strategies, in realising this scenario by means of Belief-Desire-Intention agent controllers that operate vehicles in a 3D virtual environment.

Keywords

multiagent systems, intelligent transportation systems, convoy management

1. INTRODUCTION

Constructing an Artificial Intelligence that can enable vehicles to navigate under autonomous control has been an area of research for a number of years, with output from initiatives such as the US Defense Advanced Research Projects Agency (DARPA) "Grand Challenge" (e.g. 2005 winning entry [20]) showing the promise of real vehicles fitted with arrays of sensors being able to navigate through difficult terrain.

Given the number of research efforts (e.g. [12, 16]) that are demonstrating autonomous vehicles operating on roads, a potential progression of such work is automating vehicle convoys. Indeed, the 'SAfe Road TRains for the Environment' (SARTRE) project [2] is investigating the use of autonomous vehicle control to enable "vehicle platoons" and has produced outputs identifying convoy functionality required and what may be communicated within such a platoon. However, such work is in early stages, and there does not seem to be consideration of potential benefits in communicating higher levels of SA awareness (as aiding understanding of the situation).

At the same time, Vehicle-to-Vehicle (V2V) communication is an active research area (e.g. [18]). However, the focus in V2V research tends to be on the hardware and network protocol layers, whereas our concern is what data should be communicated in order to allow vehicles to cooperate as part of a larger collective and to achieve common goals with measurable benefits e.g. improved fuel consumption, reduced journey time, etc.. We believe an open architecture is required that can enable the vehicles to pursue goals and manage their action selection through varying communication strategies. In order to produce an assessment of benefits in various approaches to V2V communication, such an architecture has been constructed.

An essential idea that has inspired our approach is that of *situational awareness* (SA) [10] and how its principles may be transferred to the domain of autonomous vehicles. The desire is to be able to capture sufficient information, at various levels of detail, about the environment, coupled with additional data pertaining to the vehicle itself, in order to be able to complement sensor-derived perceptions with higher level comprehensions about situation of the vehicle and its context.

The Belief-Desire-Intention [5] model has been adopted as an effective means to meet these requirements. BDI provides an agent based software architecture with a store of beliefs and available plans to achieve goals. These provide a loose mapping to core SA concepts (perception, comprehension, and projection), and facilitate the communication of a vehicle's current beliefs and future intentions to other vehicles. The aim is that this should augment the autonomous decision making process of other vehicles, as they will be informed of potential future events (e.g. an emergency stop occurring, and the reason for that stop) much more rapidly, rather than relying on physical sensors and beliefs inferred from that sensor feed.

The successful application of a BDI approach to convoy coordination has been demonstrated in earlier research [19] where focus was on varying convoy coordination methodologies (e.g. centralized, de-centralized, multi-agent team) and the impact on convoy split/merge activities.

We outline a range of scenarios that have been constructed to explore the impact of various V2V communication strategies, in place of or as a complement, to pure sensor approaches. The motivation here is that there are some issues

which may be best perceived at a low physical sensor level, such as that the car in front is closer than some threshold. However, some, (such as the vehicle in front speeding up in order to bring the convoy speed closer to the optimum for fuel efficiency, or the vehicle behind is leaving the convoy because it is turning off at the next junction), are clearly at the level of information, not data. The same communication strategy is not necessarily appropriate for all three of these. In the first instance, we report on the use of two approaches: data push and data pull. Push is the basic and most obvious, where vehicles publish their position data to other vehicles (without it being requested), which may even avoid the need to depend on physical sensors in some circumstances. This permits convoy members to remain informed of the position of other vehicles in the convoy, and manage their own movement based on this. Pull demands a request-response protocol, but may reduce the overall level of communication, whereby a vehicle requests information if, say, they have not received updates in a required time window, and vehicles might share their current plans and future intentions.

The remainder of the paper is set out as follows: in the next section we survey some of the large amount of related work. In section 3 we outline the simulation testbed, comprising of the Tankcoders 3D environment and the agent driving team. We have identified several scenarios that we set out in section 4, before presenting some preliminary results in section 5. We finish with some issues for future work (section 6) and conclusions (section 7).

2. RESEARCH BACKGROUND

This work attempts to combine a number of research areas in order to tackle the problem of autonomous vehicle convoys. We focus first on situational awareness (SA) in order to inform the design approach to the simulation so that a vehicle's perceptions, comprehension, and projections are accessible and observable. A number of metrics have been proposed in an attempt to measure SA, and although it is a challenging unit to quantify, the effects of incorrect or lack of SA are dominant features in many accident investigations. Hourizi [14] relates Endsley's components of SA [11] failures in understanding the current state of an aircraft, given as:

1. Failure to perceive important elements in the environment;
2. Failure to comprehend the elements that have been perceived;
3. Failure to predict the future status of those components.

Because SA can be likened to human understanding of the environment, and it is this which informs human decision making, then it follows that an incorrect or lack of SA can be the cause of incorrect actions and decisions being taken. There is little (human) self awareness as to whether (one's) SA is accurate, complete, inaccurate, or incomplete – we have very limited awareness of what we do not know. Furthermore, belief in SA is measured by the self as well, and so liable to be fallible to the (self) human in the decision making loop. If, in contrast, we place the derivation of SA within an AI context, especially within a multiagent collective, where members can contribute to others' SA, this raises the possibility that some of the weakness identified above may be addressed. Revisiting the previous excerpt of Hourizi's work with this in mind:

1. Failure to perceive important elements in the environment;

   Perceptions (e.g. obstacle detected) are passed between members of the agent collective in order to negate this issue

2. Failure to comprehend the elements that have been perceived;

   Other vehicles contribute their understanding of events; complex non-understood perceptions are referred to some other entity for resolution

3. Failure to predict the future status of those components.

   Entities within the simulation exchange their intentions and goals, adding information as to how events are likely to unfold.

The intention of combining the SA constructs with a BDI model is to address these issues and thus improve the ability of an automated system to control a vehicle, and furthermore with potential advantages over solely human control. Specifically, for the convoy scenarios being explored, members of the convoy are dependent on data exchange between their members. The aspiration is that allowing vehicles to exchange a range of, but especially higher level, data/information pertaining to their understanding of the current situation will aid members of the convoy in their action selection, improving the efficiency of the convoy as well as its ability to deal with unexpected events in the simulation.

That said, the issue of how much information should be passed between distributed agents also needs consideration. In [6], the approach proposed is to communicate only information that is needed and beneficial to other agents but it is not clear that the sender is capable of establishing these criteria. The motivation for addressing the quantity of communication is due to the cost of such communications and potential bandwidth limitations in the given scenario.

However, there is also concern regarding security and privacy: how much information should be revealed, as even some may be too much.There is a further advantage of an automated system handling such information exchange: that humans would find such communication tedious, even invasive, as well as distracting. Furthermore, humans most likely could not make good use of the information because the driving task is fairly routine. Recent [17] reports highlight that vehicle communication could lead to significant benefits in reducing motorway pile-ups.

The concept of self driving cars has been gathering increased momentum, with efforts by Google [16] to produce a self driving car, along with significant interest in developing such a concept from vehicle manufacturers. Earlier efforts took place during the series of DARPA funded challenges, and focusing on the 2007 entry of Tartan Racing, the "Perception and World Modelling" component [21], is of relevance here.

It performs "Situation Assessment" on received sensor data of tracked objects, integrates this with other knowledge of the world, and attempts to estimate the intention of this object. In relation to this SA-like concept, it is reported that the system struggles to perform well when approaching intersections and projecting future events (e.g. will a vehicle leave or join at that intersection). This provides an example of how communication of BDI constructs between vehicles could prove useful; rather than having to rely on some visual cue (e.g. an indicator light), vehicles would have been informed as to what was likely to happen at that intersection based on other vehicles belief and intention set. Allowing vehicles to communicate their planned events would potentially have the benefit that excessive braking and acceleration would be reduced, as vehicles are able to take account of expected future events rather than relying on last minute reactions.

Other work [19] has demonstrated the application of BDI to vehicle convoys in Collaborative Driving Systems (CDS), though the intention here is to explore what information exchange best supports SA (both individual and group) generation amongst the vehicles in order to improve road travel (safety, energy consumption, etc).

With this motivation and design selection in place, we turn to the construction of a suitable test bed which can support the assessment of different communication strategies, and their impact on convoy performance.

3. SIMULATION TESTBED

In order to assess the affect of various vehicle communication implementations, a testbed has been developed where a number of scenarios can be explored. To reduce the number of technological challenges faced from the outset, a simulation based approach has been selected as it offers the ability to assess performance of the system in a more controlled fashion. As the BDI component is software based, it can be tested using a simulated vehicle, allowing a base set of functionality to be established using some test scenarios.

There are a number of BDI implementations available, from which we have chosen to use Jason [3], because of its ease of extension using Java, an active support community, and the existing integration with the TankCoders virtual environment we are using for visualization of driving.

The TankCoders project [13] aimed to support research into Jason agent teams working cooperatively in a virtual environment, which it achieves by integrating Jason with a tank simulation based on the jMonkeyEngine 3D engine.

This has been revised as work has progressed, however the intention is to maintain it as being non-vehicle and non-domain specific with the aim of retaining applicability beyond the current vehicle scenario (e.g. unmanned aerial vehicles). This abstraction not only enables alternative vehicle types to be deployed within the TankCoders simulation, but decouples the high level call made from a Jason agent (e.g. `moveToXZ`) and the lower level implementation determined by the target platform (e.g. `turnWheel`, `applyTorque`, etc). This supports another objective of this work, which is to demonstrate the relevance of this research to real physical platforms as well simulated entities.

There is also the matter of building up an enhanced set of behaviours which could be considered as fundamental to the safe operation of the vehicle, an example of which is the emergency stop condition. The process of that behaviour itself resides in the agent and is not especially complicated (e.g. apply brakes, come to a complete stop, do no further actions), however by having that available, we can then consider situations in which it might be invoked. A 'bottom-line' approach to safety for any vehicle proceeding in a given direction is that, if there is some obstacle in that direction, which would be struck in the near future, then do not proceed any further in that direction. In other words, a collision avoidance behaviour. Such a behaviour also relates back to the earlier situational awareness notion, as it is a higher level inferred projection based on: (i) perceptions: obstacle detected and vehicle's current speed, (ii) comprehension: obstacle is of a significant size and will damage vehicle and (iii) projection: given current speed a collision will occur in n seconds.

With this in mind, we have developed a (naive) initial solution to the problem whereby each vehicle is placed within a collision volume that is constructed according to the vehicle's current orientation, speed, and a given time interval n. This represents the physical space that the vehicle will pass through for the next n seconds. If any object is detected within this volume, then the vehicle performs an emergency stop. Jason is able to construct this volume in the simulated environment and dynamically update as the vehicle moves, which demonstrates both a benefit of working in simulation (that the scenario can be augmented with data derived by the Jason agent, even going so far as to explain some of its SA), and the strengths of that agent in being able to derive additional data and use it to inform action selection.

Behaviours such as this can be thought of as providing primitive building blocks to allow much complex composite behaviours to be constructed. In the convoy scenario, vehicles may well follow closely behind each other, and this collision detection could get triggered if they follow too closely for some reason. In that case, the more primitive (and important) behaviour of avoiding a collision should take precedence, but the set of conditions which led to it occurring also need to be investigated in case there is a fault in how the convoy is behaving.

3.1 Agent capability

The simulation framework (elements of which are described above) provides one side of the story. The agents that are responsible for controlling the vehicles are the other. Thus, it was also necessary to develop agents that are capable of exhibiting convoy behaviour. Firstly, we have discarded the notion of a single controlling agent and replaced with a driving team, which allows us to break away from centralized control and to enable vehicles to be dynamically extended with additional capabilities as required, for example, only needing to instantiate one agent to handle convoy behaviour when the vehicle joins a convoy. The first agent used to supplement the coordinator agent is a *driver* agent, which has the primary responsibility for the vehicle's navigation. This separates off some responsibility, because the driver agent

determines a speed and asks the central agent to achieve this, and introduces flexibility, because some other agent may request the coordinator agent to reduce speed, for example, and introduces robustness, for example the ability to encapsulate alternative solutions via the BDI plan failure mechanisms. More discussion of the driving team approach appears in the next section.

Agents are able to interact with each other via Jason's communication mechanism. The infrastructure supports two communication languages (KQML and FIPA-ACL), and at present we use KQML following from the TankCoders project. The specific implementation is not the subject of interest, but rather the capability which it provides. Whilst supporting intra-vehicle agent communication (for example the driver agent asks the coordinator agent to achieve some speed), it also provide an inter-vehicle communication capability, allowing not just information related to beliefs, but also plans and goals to be sent between vehicles. Vehicles can use such information to better improve the projection element of their SA, and perhaps also modify their own plans based on the plans of other vehicles. At present, only a small set of performatives are in use – primarily `askOne`, `achieve` and `tell`. This enables vehicles to add data to other vehicles knowledge bases (e.g. inform vehicles of obstacles which they may not have detected via physical sensors), enquire as to another vehicle's beliefs (e.g. what is your position), and ask another vehicle to achieve some goal (e.g. move to a given position). These are considered as fundamental to allow vehicle groups (in this case a specific convoy) to achieve a collective goal and handle self organisation. It is the effects of varying this communication behaviour which is the subject of investigation in the convoy scenarios and which is now presented, with the specifics of the convoy agent approach presented in the results section.

Key extensions to the agents' available plans are detailed below (details regarding beliefs have been largely omitted for the sake of brevity).

Coordinator agent:

- `+!chosenSpeed(V)` – set the vehicle speed via its API; update driver agent belief with new value.
- `+!requestTurnToAngle(A)` – call vehicle API to achieve an orientation.
- `+!updateColPred(X1,Y1,Z1,X2,Y2,Z2)` – update coordinates of collision prediction volume.

Driver agent:

- `+!emergencyStop`, `+!arrivedAtDestination` – ask coordinator to achieve zero speed and to unachieve `requestTurnToAngle(_)`, abolish own desired XZ and drop desire `moveToKnownPosition`.
- `+!cruise` – ask coordinator agent to achieve `+!chosenSpeed(V)`.
- `+!applyBrakes`, `+!standardSpeed`, `+!speedUp`, `+!slowDown`, – adjust vehicle speed away from default value.
- `+!moveToKnownPosition` – using `desiredXZ(X,Z)`, if arrived then `+!arrivedAtDestination`, otherwise determine direction A to X,Z; ask coordinator agent to achieve `requestTurnToAngle(A)`, then `cruise`.

Convoy member agent:

- `+vehAheadInfo(X,Y,Z,_,_,_,_)` (Data push scenario) Using X,Y,Z of vehicle ahead, determine distance to that vehicle and ask driver agent to achieve `standardSpeed`, `speedUp` or `slowDown` to maintain convoy gap. Tell driver agent to update its belief to `desiredXZ(X,Z)` and then to achieve `moveToKnownPosition`.
- `+!convoyMgmtPlan` (Data pull scenario) At three second intervals, ask the vehicle ahead `convoyMemberInfo(X,Y,Z,_,_,_,_)`. Tell driver agent to update its belief to `desiredXZ(X,Z)` and then to achieve `moveToKnownPosition`.

4. VEHICLE CONVOY SCENARIOS

The vehicle convoy domain has been selected for a number of reasons. This topic has been attracting attention recently, with the potential benefits of vehicle platoons being reported [2]: up to twenty percent reduction in fuel consumption, ten percent reduction in fatalities, and improved driver convenience (for passenger-drivers in the vehicles where control has been ceded to the platoon). Benefits have also been claimed recently [9] with improved traffic efficiency as a key goal. In relation to environmental considerations, [15] shows that the total trip time for journeys can be significantly improved through vehicle to vehicle communication. This study also shows that if navigation systems share traffic information, then journey times can be shortened, highlighting the potential benefit that sharing simple beliefs of BDI constructs may bring.

However, the application of this research does not reside purely in vehicle convoys; rather this has been selected as a key area where vehicle behaviours (such as information sharing and common goals) lend themselves to explore the benefits of SA-like knowledge exchange in a challenging but relevant context. As such, there are a number of limitations to the scenarios in use. Firstly, there is no road model; vehicles are bounded only by physics such as collisions with other objects and terrain. Secondly, there is no traffic model; at this stage we are only considering how vehicles communicate at an intra-convoy level. This will be extended once the scenarios become more broad as we wish to build up a larger SA picture into the convoy performance (e.g. external convoy members informing of obstacles ahead, and the convoy deciding on a course of action based on this). This will add further understanding to the question being posed, as if we are addressing how much intra-vehicle communication is required, it certainly follows that non-convoy member communication (e.g. position updates from other vehicles) requires consideration.

We have chosen to modularize the structure of the driving system by developing substantial new behaviours as separate agents, rather than as additional behaviours of an existing agent. This is already clear in the initial structure where there is a coordinator agent and a driver agent. The motivation is that new behaviours can be added (or removed) by the introduction (or removal) of an agent, rather than the modification of an existing agent: it only requires that the central coordinating agent is informed of new functionality (or its loss), while the collection of agents function as in-

dividual self-interested entities under the governance of the common objective of getting the vehicle to its destination (for example).

Two further motivations for this behavioural decoupling are: (i) to keep individual agent behaviour specifications "small enough" to be maintainable and to minimise the potential impact of hard-to-identify bugs arising from the aggregation of behaviour within a single BDI reasoning engine, and (ii) to keep constituent agent reasoning cycles short enough that response times might potentially be adequately controlled for close enough to real-time behaviour. It remains to be seen how well each of these is borne out in practice.

One such additional behaviour is the agent responsible for vehicle behaviour in the convoy collective. At its most basic, on instantiation this agent is informed of the vehicles in front and behind in the convoy of which it is a member, and on receipt of the vehicle in front position data, it seeks to move to that position. Both the driver agent and convoy member agent are currently generic, so the same agent capability is embedded across all the vehicles in a convoy.

As mentioned previously, KQML is used as the communication language in this testbed. KQML is used at both intra- and inter-vehicle communication. For example to allow a driver agent to request a speed from the coordinator agent, or to allow a coordinator agent to update a driver agent on current position within simulation. At inter-vehicle level, it allows the convoy agent of vehicle 1 to ask the convoy agent of vehicle 2 for the current location of vehicle 2.

It is this communication mechanism, coupled with the intrinsic BDI data constructs, which is the topic of interest regarding the impact it has on the success and efficiency of convoy behaviour(s) in the scenario(s). The main communication strategies can be broken down in line with the BDI paradigm, that is, inter-vehicle sharing of beliefs, desires and intentions.

Our first step has been an investigation of the benefits of sharing beliefs and two convoy scenarios are presented in section 5 based on this. The two approaches differ in how the data is transmitted; the first requires all vehicles to inform other vehicles of their position at every tick of the simulation, while the second implements a request approach where each vehicle determines when to ask some other vehicle for its current position details.

At present, we are focussing on belief sharing and this has produced some initial statistics and observations on (convoy) behaviour, depending on whether the data is pushed (i.e. sent out at some tick interval to n agents), or pulled (agents request information from other agents at their chosen interval). The first implementation of a convoy has been based on each convoy member knowing the identity of the convoy member behind it, and at each simulation tick using a KQML performative `send(vehicleBehind, tell, vehicleInfrontPosition(X,Z))` to advise its position to the vehicle following. Upon receipt of a `vehicleInfrontPosition(X,Z)` belief, a convoy member establishes the goal of `moveToXZ(X,Z)` thus following the path of the vehicle in front.

5. RESULTS

The experiments presented at this point are based around five scenarios. The first three scenarios are baseline assessments of the operation of the framework, and use no convoy member agents. The fourth scenario implements the data push communication strategy, in which, data is pushed at a regular interval between convoy members, where each member passes its position to the vehicle behind. The fifth scenario implements the data pull strategy, in which data is pulled by request from a specified agent to the requestor. Precise details are given in the following section.

5.1 Convoy scenarios

The detail of each scenario, and the intention of what it should assess, is as follows:

Scenario 1: Four vehicles, with a driver agent but no destination to achieve, no convoy member agent. Assess: Baseline of physics simulation and rendering of four vehicles.

Scenario 2: Four vehicles, with a driver agent and given a destination, no convoy member agent. Assess: Impact of using the driver agent on the initial baseline.

Scenario 3: Two vehicles, with a driver agent and given destination, no convoy member agent. Assess: Impact workload of half as many vehicles and driver agents.

Scenario 4: Four vehicles, with a driver agent, where lead vehicle's driver agent is given a destination, and each vehicle has a convoy member agent based on convoy strategy 1. Assess: Initial convoy strategy dependent on high communication traffic between convoy agents.

Scenario 5: Four vehicles, with a driver agent, where lead vehicle's driver agent is given a destination, and each vehicle has a convoy member agent based on convoy strategy 2. Assess: Impact of reduced communication between convoy agents.

In the following push and pull strategies, there is an assumption that a 'convoy join' behaviour has already taken place, resulting in three vehicles following the lead vehicle. Part of this behaviour would involve determining whether the convoy is heading (at least partly) in the direction required. On joining the convoy, there is an abdication of route planning responsibility as part of the ceding of individual autonomy to the collective convoy, and instead navigation involves following the trail of the vehicle in front.

5.1.1 Data push strategy

Convoy strategy one implements the following approach:

- Only the lead vehicle knows the final destination.
- The lead vehicle's coordinator agent starts the movement by sending a message to its driver agent regarding the desired location:
  `send(driverAgent, tell, desiredXZ (500,2500))`
  followed by a message to achieve a movement to that destination:
  `send(driverAgent, achieve, moveToKnownPosition)`.
- Each vehicle in the convoy starts a convoy member agent.

- Each convoy member agent is told the vehicle's driver agent name (in order to be able to send messages regarding updated positions to move to) and the name of the convoy member agent of the vehicle behind (in order to push data to the correct vehicle).
- On every simulation update cycle, the coordinator agent tells its convoy member agent and driver agent the vehicle's new position.
- When a driver agent receives a position update, it uses this to calculate the distance remaining to the `desiredXZ` and perform any necessary actions (e.g. course corrections, or stop if at that location).
- When a convoy member agent receives a position update from its coordinator agent, it performs the data push of telling the following convoy member agent this new position.
- When a convoy member agent receives a position updates from the vehicle ahead, it tells its driver agent as a new `desiredXZ` followed by the request to achieve `moveToKnownPosition`.

5.1.2 *Data pull strategy*

Convoy strategy two implements the following approach:

1. Only the lead vehicle knows the final destination.
2. The lead vehicle's coordinator agent starts the movement by sending a message to its driver agent regarding the desired location:
   `send(driverAgent, tell, desiredXZ (500,2500))`
   followed by a message to achieve a movement to that destination:
   `send(driverAgent, achieve, moveToKnownPosition)`.
3. Each vehicle in the convoy starts a convoy member agent.
4. Each convoy member agent is told the vehicle's driver agent name (in order to be able to send messages regarding updated positions to move to) and the name of the convoy member agent for the vehicle ahead (in order to pull data from the correct vehicle).
5. Each convoy member agent starts a convoy management plan, in order to handle the data pull aspect. At present, every 3 seconds this plan uses the KQML performative `askOne` to ask the vehicle ahead's current position.
6. When a convoy member receives a reply containing the position of the vehicle ahead, it sends this to its driver agent as a new `desiredXZ` followed by the request to achieve `moveToKnownPosition`.

5.2 Scenario results

The focus of the discussion here is on scenarios four and five, as these demonstrate the affect of varying the convoy communication strategy. Scenarios one, two and three demonstrate consistent behaviour and sufficient performance of the simulation to have confidence in the output from scenarios four and five.

Figure 1 plots the position reports of each convoy member, as the convoy forms and moves from starting positions (approximately 0,2000) to the fixed destination given to the lead vehicle (500,2500). Figure 1 shows the convoy positions during the fourth scenario, i.e. convoy strategy one (data push). By comparison, in Figure 2 the position of each vehicle is shown for the fifth scenario, i.e convoy strategy two (data pull). In this simple situation where the destination of the lead vehicle is not changing, it is evident that there is little difference between the two approaches in terms of the route taken by the convoy and its member vehicles. However, it can be observed that, during the transition from start conditions to the steady state behaviour when moving towards the (non-changing) destination, vehicle 4 (starting at the farthest left in the figures) does differ between the two scenarios. This shows the impact of increasing the gap between position updates: between updates from the vehicle in front vehicle 4 has diverged slightly from the convoy direction. In this particular scenario (i.e. with a fixed destination for the lead convoy vehicle) there is no real impact as, with the next position update, the vehicle realigns to the convoy. However, in a case where the lead convoy vehicle changes route more frequently (i.e. navigation through a congested city with many intermediate destinations or waypoints) this could result in greater divergence from the convoy grouping. This also suggests where benefits may arise from obtaining higher level information from the convoy member in front (e.g. desired final location) rather than low level details (current position), and this will be the subject of investigated in future scenarios.

During the scenarios involving these two convoy strategies data was collected to capture the effect of the different approaches on the communication volume, specifically the number of percepts and the number of messages. In Figure 3 the marked contrast can be seen between strategy one (data push) and strategy two (data pull). The approach of a data push has resulted in approximately five times as many percepts being registered by vehicles two, three and four compared to the same vehicles using a data pull approach. This result is expected, as the same data (of vehicle position) is being communicated in both scenarios, however in the data pull strategy the communication frequency is lower (as the responsibility for when to ask for this data resides with the receiving vehicle and it does so every few seconds compared to at every opportunity) and as such fewer percept updates are received. In Figure 4 the number of messages transmitted for the four vehicles is shown, and a similar profile to that of Figure 3 can be seen. Compared to Figure 3 however, data points do not begin until approximately 20 seconds have elapsed. This is due to how the simulation starts; as soon as the environment is instantiated (and the vehicle agents created) they begin receiving percepts. However, until the vehicles begin moving and using the convoy strategy there will be no exchange of messages, and this only occurs after the simulation has been fully initialised (approximately 20 seconds into the data capture).

In both figures it can be seen that vehicle 1 follows the same profile across both scenarios. This provides an expected correlation, as in both convoy strategies the lead vehicle is performing a role where its communication is significantly different to the rest of the convoy. The lead vehicle receives no external position updates as it is not following any other vehicle, instead only sending data to the vehicle behind. This would seem to confirm that there are no other sources of percept generation (e.g. mass broadcast of position data to all convoy members rather than to the specified target vehicle), which confirms the observed results are indeed due to

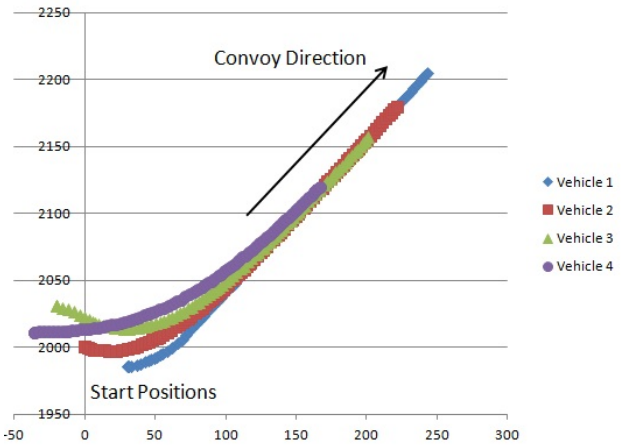


Figure 1: Vehicle positions with convoy strategy one

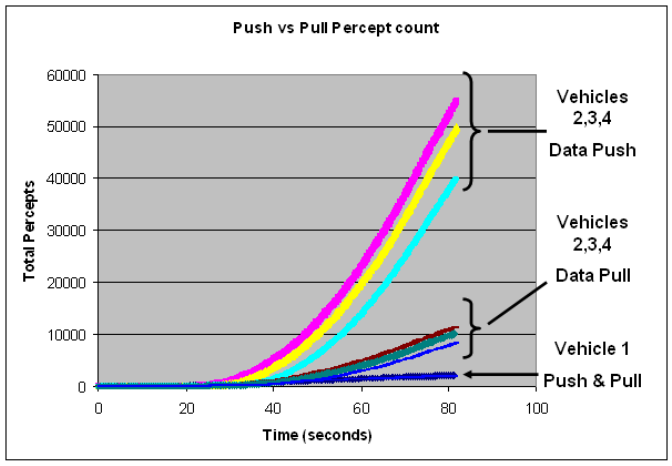


Figure 3: Vehicle percept updates

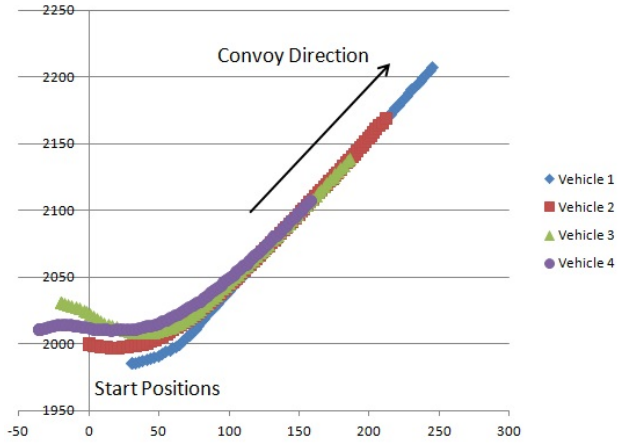


Figure 2: Vehicle positions with convoy strategy two

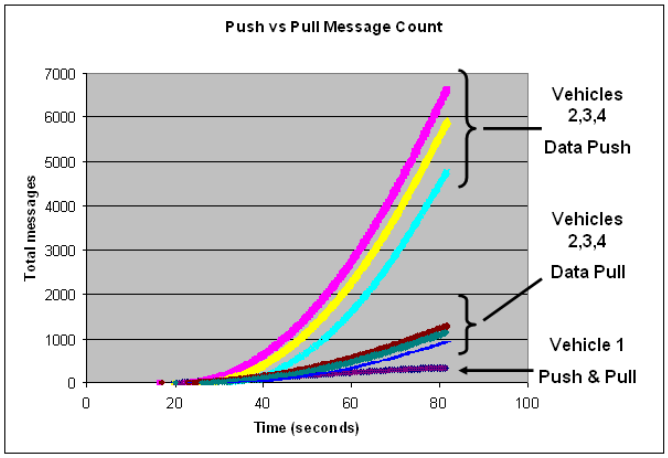


Figure 4: Vehicle message counts

the variation in convoy strategy rather than other factors.

Although no specific performance metrics have been developed yet, there was a notable impact on the simulation during scenario five, as the frames per second rate dropped from approximately 19fps to 12fps. The system performance (measured by frame rate) during these two scenarios is shown in Figure 5, where this difference in performance can be seen. With the performance of the simulation dropping to such levels, we conclude that the resources consumed by communication are impacting the ability of the system to carry out computation. Video capture of the two convoy strategies is available in mp4 format, for scenario 4 (data push) at http://people.bath.ac.uk/vb216/dataPush.mp4 and for scenario 5 (data pull) at http://people.bath.ac.uk/vb216/dataPull.mp4.

In both videos it can be seen that the frame rate differs from that shown in Figure 5, due to the increased load on the system of capturing the video stream. However, it can be seen that there is still a performance difference between the two scenarios, with data-pull outperforming data-push.

If the simulation performance drops much further, it has been observed that unexpected and unpredictable agent behaviour occurs and convoy behaviour breaks down. This issue is one of the motivations for the decoupling of software components discussed in the next section.

6. FUTURE WORK

At present Jason is quite tightly integrated with the TankCoders platform, which is good in some respects for performance, although we have already experienced stochastic behaviour arising from tracing that has further obscured the issues we were attempting to observe. In the next phase of our work, despite some concern over the performance impact of the introduction of middleware, we wish to decouple the various components for four reasons:

1. We seek to avoid a repeat of the problem cited above, that monitoring perturbs the system further.
2. Experience elsewhere has taught us that large numbers of agents on a single Jason instance can be problematic, so we would like to be able to connect multiple Jason instances to a single TankCoders environment. In addition, this would permit driver teams to be lo-

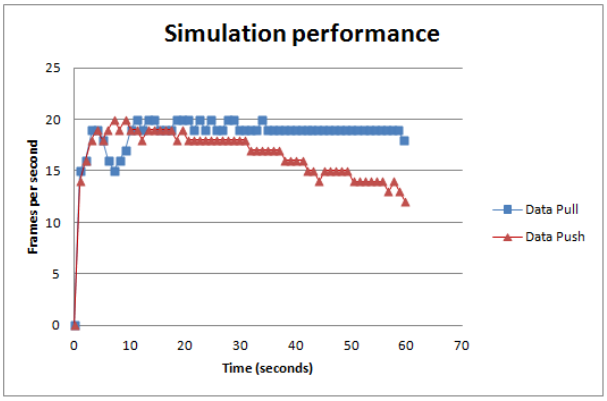


Figure 5: System performance variation

cated anywhere on the Internet, not just on the same machine as the virtual environment.

3. A critical feature of the next phase is the introduction of normative framework [8] to capture the rules of the convoy in the form of an externally reference-able entity that governs the behaviour of individual teams, as well as subsequently exploring interaction between convoy instances [7] to handle operations such as merging, splitting and passing through one another. Previous experience [1] of its integration, encourages us to decouple the normative framework from the agent platform.
4. Finally, useful though working with the TankCoders environment is, the harshest environment is the physical world and so we wish to substitute physics models of vehicles with simple robot vehicles, in this case LEGO Mindstorms platforms carrying android mobile phones as communication devices.

In pursuit of these goals, we are currently developing the means for the various components identified above to communicate using the Extensible Messaging and Presence Protocol (XMPP). XMPP is in widespread use underpinning internet messaging systems, but it is equally applicable for inter-program communication and for the collection of sensor data (our initial application). Thus, by taking an event-oriented view of the world and treating each of the above components as event consumers and producers in conversations enabled by XMPP, it is relatively straightforward to achieve the desired decoupling.

Further work is also necessary on the simulation system itself. Some refinement is required on the generation of system level metrics, such as those presented in Figures 3 and 4, in order to improve the efficiency of data collection and ensure there is minimal impact on system performance. An extension is also planned to provide a breakdown of the message and percept types being communicated, in order to understand further what is being exchanged between agents. As the scenarios grow more complex, this is expected to be essential in order to follow the interactions occurring between the vehicles and their agents.

The scenarios also need to be extended to add both realism and challenge to the vehicle agents. A more complex convoy route is required to more clearly demonstrate merits between differing convoy strategies, and also to identify strengths and weaknesses of varying communicated data (e.g. beliefs vs intentions). As the complexity increases, so too will the likelihood of calling upon the ability of Jason to handle plan failures, as unexpected situations and interactions occur.

With this in place, more advanced metrics measuring the impact of varying convoy strategies are needed. Two already under development are fuel management and convoy route deviation measures. The first involves the integration of a simplified engine model into the simulation, such that inefficiencies (e.g. high engine revs, excessive acceleration and braking) in the drive of the vehicle will be reflected in the fuel consumption. Such work will also introduce the ability to explore competing objectives between agents (e.g. fuel management requiring a slow speed to conserve fuel, convoy agent requiring a high speed to maintain convoy position). The second, convoy route deviation measure, is to extend the results being produced which produced Figure 1 and Figure 2. This will produce a metric indicating how well the convoy is performing in geographic cohesiveness and highlight deviations from its route.

A major future development is to utilise a normative framework within the system and to capture a reasonable set of both legal governance and societal convention into this architecture. The design and implementation of the normative solution will require significant effort. Work has been presented in [1] demonstrating a methodology for the utilisation of institutional models of governance in open systems. This raises a number of questions which will need consideration, such as whether an individual agent should refer its action selection to a normative control, or does a normative agent model actions at an individual vehicle level, how will the convoy be regulated, and are certain actions allowable but involve a punishment mechanism? The work of [1] also demonstrates the feasibility of integrating BDI (and specifically Jason) with institutional models. The work of Bradshaw [4] also touches on the notion of potential actions vs permitted actions, raises the question of how some adjustable autonomy will be managed (e.g. action selection when in convoy vs action selection when driving as individual). Some larger scenarios are likely to be required to investigate these questions, and the effort of both this and the normative framework itself positions this work in a more ready state for transition into a real-world domain.

As discussed earlier, another aspect of development is the decoupling of this study from the TankCoders-jMonkeyEngine simulation in order to connect it with a real sensor-actuator capability. This process is underway, with integration to an XMPP based sensor framework in early stages. This will allow the simulation of a vehicle instance from TankCoders to be replaced with a real vehicle, passing geographical data back to Jason and responding to Jason agent requests. Work is progressing to formalise the specifics of message exchange format, and this will then form the basis of a ratification of the V2V communication strategy by introducing real world limitations, e.g. latency, bandwidth.

We plan to experiment with an Android device coupled with a real world platform (a remote control car with an IOIO breakout board) that provides a sensor suite from the android device (orientation, position) coupled with an appropriate actuator. In addition, work is in progress to couple this system to the LEGO Mindstorms platform, with a fuller set of functional XMPP message being developed.

7. CONCLUSIONS

This work demonstrates that a usable simulation framework has been constructed, capable of supporting the next phase, which will focus on the investigation of benefits in BDI type message exchange to support of vehicle convoy behaviour. Results to date are:

1. Demonstration of a working simulation with Jason Belief-Desire-Intention agent controlled simulated vehicles.
2. An agnostic control design, where agents are not specific to a vehicle (e.g. weight, size, power), or type (e.g. locally simulated vehicle, or remote XMPP vehicle).
3. An initial convoy scenario exploring the performance of a 'data push' of vehicle positions.
4. A second convoy scenario exploring the performance of a 'data pull' of vehicle positions.
5. An initial suite of metrics to measure aspects of system and convoy performance.

Having established our foundations, we will now work towards more credible vehicle scenarios. Following this, the integration of a normative framework will be explored such that the governance of this vehicle collective is established. Finally, the applicability to real platforms will be demonstrated through the use of remote physical vehicles.

8. REFERENCES

[1] Tina Balke. *Towards the Governance of Open Distributed Systems: A Case Study in Wireless Mobile Grids*. PhD thesis, University of Bayreuth, November 2011. Available via http://opus.ub.uni-bayreuth.de/volltexte/2011/929/. Retrieved 20120109. Also available as ISBN-13: 978-1466420090, published by Createspace.

[2] C. Bergenhem, Q. Huang, A. Benmimoun, and T. Robinson. Challenges of platooning on public motorways. In *17th World Congress on Intelligent Transport Systems*, 2010.

[3] R. H. Bordini, J. F. Hübner, and M Wooldridge. *Programming multi-agent systems in AgentSpeak using Jason*. Wiley, 2007.

[4] Jeffrey M. Bradshaw, Paul J. Feltovich, Hyuckchul Jung, Shriniwas Kulkarni, William Taysom, and Andrzej Uszok. Dimensions of adjustable autonomy and mixed-initiative interaction. *Computational Autonomy*, 2969:17–39, 2004.

[5] M. E. Bratman, D. J. Israel, and M. E. Pollack. Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4:349Ű355, 1988.

[6] J. Chen. Distributed coordination of autonomous agents by communicating on a need-to-know basis. Master of engineering, Massachusetts Institute of Technology, 2003.

[7] O. Cliffe, M. De Vos, and J. A. Padget. Specifying and reasoning about multiple institutions. In *Coordination, Organizations, Institutions, and Norms in Agent Systems Lecture Notes in Artificial Intelligence*. Springer, 2007.

[8] Owen Cliffe, Marina De Vos, and Julian Padget. Answer set programming for representing and reasoning about virtual institutions. In Katsumi Inoue, Ken Satoh, and Francesca Toni, editors, *CLIMA VII*, volume 4371 of *Lecture Notes in Computer Science*, pages 60–79. Springer, 2006.

[9] F. Dressler, F. Kargl, J. Ott, O. Tonguz, and L. Wischhof. 10402 abstracts collection and executive summary – inter-vehicular communication. In *Inter-Vehicular Communication*, number 10402 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2011. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany.

[10] M. R. Endsley. Toward a theory of situation awareness in dynamic systems. *Human Factors*, 37:32–64, 1995.

[11] Mica R. Endsley. Toward a theory of situation awareness in dynamic systems. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 37(1):32–64, 1995.

[12] Freie Universitat Berlin. Autonomous car navigates the streets of Berlin. http://autonomos.inf.fu-berlin.de/news/press-release-92011, September 2011. accessed October 8th, 2011.

[13] Germano Fronza. Simulador de um ambiente virtual distribuido multiusuario para batalhas de tanques 3d com inteligencia baseada em agentes bdi. http://campeche.inf.furb.br/tccs/2008-I/2008-1-14-ap-germanofronza.pdf, July 2008.

[14] R. Hourizi. *Awareness beyond mode error*. PhD thesis, University of Bath, 1999.

[15] I. Leontiadis, G. Marfia, D. Mack, G. Pau, C. Mascolo, and M. Gerla. On the effectiveness of an opportunistic traffic management system for vehicular networks. In *IEEE Transactions on Intelligent Transportation Systems*. IEEE Intelligent Transportation Systems Society, 2011.

[16] J. Markoff. Google cars drive themselves, in traffic. http://www.nytimes.com/2010/10/10/science/10google.html, October 2010. accessed October 8th, 2011.

[17] Katia Moskvitch. 'Talking' cars could reduce motorway pile-ups. http://www.bbc.co.uk/news/technology-14125245, July 2011. accessed October 8th, 2011.

[18] M. Raya, P. Papadimitratos, and J.-P. Hubaux. Securing Vehicular Communications. *IEEE Wireless Communications*, 13:8–15, October 2006.

[19] Brahim Chaib-draa and Simon Hallé. A collaborative driving system based on multiagent modelling and simulations. *Journal of Transportation Research Part C (TRC-C): Emergent Technologies*, 13(4):320–345, 2005.

[20] Sebastian Thrun and Mike Montemerlo et al. Stanley: The robot that won the DARPA Grand Challenge. *Journal of Field Robotics*, 23(9):661–692, 2006.

[21] C. Urmson and W. Whittaker. Self-driving cars and the urban challenge. *IEEE Intelligent Systems*, 23:66–68, March 2008.

An Agent-Based Approach to Pedestrian and Group Dynamics: Experimental and Real World Scenarios

Giuseppe Vizzari, Lorenza Manenti*
Complex Systems and Artificial Intelligence research center, Università degli Studi di Milano–Bicocca, Milano, Italy
{vizzari,manenti}@disco.unimib.it

Kazumichi Ohtsuka, Kenichiro Shimura
Research Center for Advanced Science & Technology, The University of Tokyo, Japan
tukacyf@mail.ecc.u-tokyo.ac.jp, shimura@tokai.t.u-tokyo.ac.jp

ABSTRACT

The simulation of pedestrian dynamics is a consolidated area of application for agent-based models: successful case studies can be found in the literature and off-the-shelf simulators are commonly employed by decision makers and consultancy companies. These models, however, generally do not consider the explicit representation of pedestrians aggregations (groups), the related occurring relationships and their dynamics. This work is aimed at discussing the relevance and significance of this research effort with respect to the need of empirical data about the implication of the presence of groups of pedestrians in different situations (e.g. changing density, spatial configurations of the environment). The paper describes an agent-based model encapsulating in the pedestrian's behavioural specification effects representing both traditional individual motivations (i.e. tendency to stay away from other pedestrians while moving towards the goal) and a simplified account of influences related to the presence of groups in the crowd. The model is tested in a simple scenario to evaluate the implications of some modeling choices and the presence of groups in the simulated scenario. Moreover, the model is applied in a real world scenario characterized by the presence of organized groups as an instrument for crowd management. Results are discussed and compared to experimental observations and to data available in the literature.

Categories and Subject Descriptors

I.6 [**Simulation and Modeling**]: Applications

General Terms

Experimentation

Keywords

pedestrian and crowd modeling, interdisciplinary approaches

1. INTRODUCTION

Agent–based approaches to the simulation of complex systems represent a relatively recent but extremely successful application area of concepts, abstractions, models defined in the area of autonomous agents and multi–agent systems (MAS). Agent-based models have been adopted to model complex systems in very different contexts, ranging from social and economical simulation to logistics optimization, from biological systems to traffic. Large groups and crowds of pedestrians represent a typical example of complex system: the overall behavior of the system can only be defined in terms of the actions of the individuals that compose it, and the decisions of the individuals are influenced by the previous actions of other pedestrians sharing the same space. Sometimes the interaction patterns are *competitive*, since pedestrians may have conflicting goals (i.e. they might wish to occupy the same spot of the shared environment), but *collaborative* patterns can also be identified (e.g. leave room to people getting off a subway train before getting on). The overall system is characterized by self-organization mechanisms and emergent phenomena.

Despite the complexity of the studied phenomenon, the relevance of human behaviour, and especially of the movements of pedestrians, in built environment in normal and extraordinary situations, and its implications for the activities of architects, designers and urban planners are apparent (see, e.g., [3]), especially considering dramatic episodes such as terrorist attacks, riots and fires, but also due to the growing issues in facing the organization and management of public events (ceremonies, races, carnivals, concerts, parties/social gatherings, and so on) and in designing naturally crowded places (e.g. stations, arenas, airports). Computational models for the simulation of crowds are thus growingly investigated in the scientific context, and these efforts led to the realization of commercial off-the-shelf simulators often adopted by firms and decision makers[1]. Models and simulators have shown their usefulness in supporting architectural designers and urban planners in their decisions by creating the possibility to envision the behavior of crowds of pedestrians in specific actual environments and planned designs, to elaborate what-if scenarios and evaluate their decisions

*Crystals Project, Centre of Research Excellence in Hajj and Omrah (Hajjcore), Umm Al-Qura University, Makkah, Saudi Arabia.

[1]see `http://www.evacmod.net/?q=node/5` for a significant although not necessarily comprehensive list of simulation platforms.

with reference to specific metrics and criteria. Despite the substantial amount of research efforts this area is still quite lively and we are far from a complete understanding of the complex phenomena related to crowds of pedestrians in the environment: one of the least studied and understood aspects of crowds of pedestrians is represented by the implications of the presence of groups [6]. In particular, little work in the direction of modeling and simulating relatively large groups within a crowd of pedestrians encompassing some form of validation (either quantitative or qualitative) against real data can be found in the literature.

The main aim of this work is to present motivations, fundamental research questions and directions, and results of an agent–based modeling and simulation approach to the multidisciplinary investigation of the complex dynamics that characterize aggregations of pedestrians and crowds. In particular, in this paper we will present an agent–based model of pedestrians considering groups as a first–class abstraction influencing the behaviour of its members and, in turn, of the whole system. The model has been tested (i) in a schematic situation that has also been analyzed by means of field experiments to characterize the implications of groups in the overall pedestrian dynamics and (ii) in a real world scenario in which pedestrians were organized in large groups for sake of crowd management.

The paper breaks down as follows: the following section will set the present work in the state of the art of pedestrian and crowd modeling and simulation, with specific reference to recent works focusing on the modeling and implications of groups. Section 3 will introduce the model that was adopted in an experimental scenario, described in section 4, and in a real world scenario, described in section 5. The scenarios will be described and the achieved results will be discussed. Conclusions and future developments will end the paper.

This work is set in the context of the Crystals project[2], a joint research effort between the Complex Systems and Artificial Intelligence research center of the University of Milano–Bicocca, the Centre of Research Excellence in Hajj and Omrah and the Research Center for Advanced Science and Technology of the University of Tokyo. The main focus of the project is on the adoption of an agent-based pedestrian and crowd modeling approach to investigate meaningful relationships between the contributions of anthropology, cultural characteristics and existing results on the research on crowd dynamics, and how the presence of heterogeneous groups influence emergent dynamics in the context of the Hajj and Omrah. The implications of particular relationships among pedestrians in a crowd are generally not considered or treated in a very simplistic way by current approaches. In the specific context of the Hajj, the yearly pilgrimage to Mecca that involves over 2 millions of people coming from over 150 countries, the presence of groups (possibly characterized by an internal structure) and the cultural differences among pedestrians represent two fundamental features of the reference scenario. Studying implications of these basic features is the main aim of the Crystals project.

[2]http://www.csai.disco.unimib.it/CSAI/CRYSTALS/

2. RELATED WORKS

A comprehensive but compact overview of the different approaches and models for the simulation of pedestrian and crowd dynamics is not easily defined: scientific interdisciplinary workshops and conferences are in fact specifically devoted to this topic (see, e.g., the proceedings of the first edition of the International Conference on Pedestrian and Evacuation Dynamics [23] and consider that this event has reached the fifth edition in 2010). A possible schema to classify the different approaches is based on the way pedestrians are represented and managed. From this perspective, pedestrian models can be roughly classified into three main categories that respectively consider pedestrians as *particles subject to forces*, particular *states of cells* in which the environment is subdivided in Cellular Automata (CA) approaches, or *autonomous agents* acting and interacting in an environment.

The most successful particle based approach is represented by the *social force model* [9], which implicitly comprises fundamental proxemical [8] concepts like the tendency of a pedestrian to stay away from other ones while moving towards his/her goal. Proxemics essentially represents a fundamental assumption of most modeling approaches, although very few authors actually mention this anthropological theory [26, 12].

CA based approaches can be roughly classified in ad-hoc approaches for specific situations (like the case of bidirectional flows at intersections described in [4]) and general approaches, whose main representative is the floor-field approach [21], in which the cells are endowed with a discretized gradient guiding pedestrians towards potential destinations.

While particle and CA based approaches are mostly aimed at generating quantitative results about pedestrian and crowd movement, agent based approaches are sometimes aimed at the generation of effective visualizations of believable crowd dynamics, and therefore the above approaches do not necessarily share the same notion of realism and validation. Works like [1] and [10] essentially extend CA approaches, separating the pedestrians from the environment, but they essentially adopt similar methodologies. Other approaches like [15, 24] are more aimed at generating visually effective and believable pedestrians and crowds in virtual worlds. Other approaches, like [17], employ cognitive agent models for different goals, but they are not generally aimed at making predictions about pedestrian movement for sake of decision support.

A small number of recent works represent a relevant effort towards the modeling of groups, respectively in particle-based [14, 28] (extending the social force model), in CA-based [20] (with ad-hoc approaches) and in agent-based approaches [18, 19, 25, 13] (introducing specific behavioral rules for managing group oriented behaviors): in all these approaches, groups are modeled by means of additional contributions to the overall pedestrian behaviour representing the tendency to stay close to other group members. However, the above approaches only mostly deal with small groups in relatively low density conditions; those dealing with relatively large groups (tens of pedestrians) were not validated against real data. The last point is a crucial and critical

element of this kind of research effort: computational models represent a way to formally and precisely define a computable form of theory of pedestrian and crowd dynamics. However, these theories must be validated employing field data, acquired by means of experiments and observations of the modeled phenomena, before the models can actually be used for sake of prediction.

3. GA-PED MODEL

We will now briefly introduce a model based on simple reactive situated agents based on some fundamental features of CA approaches to pedestrian and crowd modeling and simulation, with specific reference to the representation and management of the simulated environment and pedestrians; in particular, the adopted approach is discrete both in space and in time. The present description of the model is simplified and reduced for sake of space, reporting only a basic description of the elements required to understand its basic mechanisms; an extended version of the model description can be found in [2].

3.1 Environment

The environment in which the simulation takes place is a lattice of cells, each representing a portion of the simulated environment and comprising information about its current state, both in terms of physical occupation by an obstacle or by a pedestrian, and in terms of additional information, for instance describing its distance from a reference point or point of interest in the environment and/or its desirability for pedestrians following a certain path in the environment.

The scale of discretization is determined according to the principle of achieving cells in which at most one pedestrian can be present; traditionally the side of a cell is fixed at 40 or 50 cm, respectively determining a maximum density of 4 and 6.5 pedestrian per square meter. The choice of the scale of discretization also influences the length of the simulation turn: the average speed of a pedestrian can be set at about 1.5 meters per second (see, e.g., [27]) therefore, assuming that a pedestrian can perform a single movement between a cell and an adjacent one (according to the Von Neumann neighbourhood), the duration of a simulation turn is about 0.33 seconds in case of a 50 cm discretization and 0.27 in case of a finer 40 cm discretization.

Each cell can be either vacant, occupied by an obstacle or by a specific pedestrian. In order to support pedestrian navigation in the environment, each cell is also provided with specific floor fields [21]. In particular, each relevant final or intermediate target for a pedestrian is associated to a floor field, representing a sort of gradient indicating the most direct way towards the associated point of interest (e.g., see Fig.1 in which a simple scenario and the relative floor field representation are shown). The GA-Ped model only comprises *static* floor fields, specifying the shortest path to destinations and targets. Interactions between pedestrians, that in other models are described by the use of *dynamic floor fields* [16], in our model are managed through the agent perception model.

3.2 Pedestrians

Pedestrians in the GA-PED model have a limited form of autonomy, meaning that they can choose were to move according to their perception of the environment and their goal, but their action is actually triggered by the simulation engine and they are not thus provided with a thread of control of their own. More precisely, the simulation turn activates every pedestrian once in every turn, adopting a random order in the agent selection: this agent activation strategy, also called *shuffled sequential updating* [11], is characterized by the fact that conflicts between pedestrians are prevented.

Each pedestrian is provided with a simple set of attributes: $pedestrian = \langle pedID, groupID \rangle$ with *pedID* being an identifier for each pedestrian and *groupID* (possibly null, in case of individuals) the group the pedestrian belongs to. For the applications presented in this paper, the agents have a single goal in the experimental scenario, but in more complex ones the environment could be endowed with multiple floor fields and the agent could be also characterized by a *schedule*, in terms of a sequence of floor fields and therefore intermediate destinations to be reached.

The behavior of a pedestrian is represented as a flow made up of three stages: *sleep*, *movement evaluation*, *movement*. When a new iteration starts each pedestrian is in a sleeping state. The system wakes up each pedestrian once per iteration and, then, the pedestrian passes to a new state of movement evaluation. In this stage, the pedestrian collects all the information necessary to obtain spatial awareness. In particular, every pedestrian has the capability to observe the environment around him, looking for other pedestrians (that could be part of his/her group), walls and other obstacles, according to the Von Neumann neighbourhood. The choice of the actual movement destination between the set of potential movements (i.e. non empty cells are not considered) is based on the elaboration of an utility value, called *likability*, representing the desirability of moving into that position given the state of the pedestrian.

Formally, given a pedestrian belonging to a group g and reaching a goal t, the *likability* of a cell $c_{x,y}$ is defined as:

$$li(c_{x,y}, g, t) = w_t \cdot goal(t, (x, y)) + w_g \cdot group(g, (x, y)) - w_o \cdot obs(x, y) - w_s \cdot others(g, (x, y)) + \epsilon. \quad (1)$$

where the functions *obst* counts the number of obstacles in the Von Neumann neighbourhood of a given cell, *goal* returns the value of the floor field associated to the target t in a give cell, *group* and *other* respectively count the number of members and non-members of the group g, ϵ represents a random value. Group cohesion and floor field are positive components because the pedestrians wish to reach their destinations quickly, while staying close to other group members. On the contrary, the presence of obstacles and other pedestrians have a negative impact as a pedestrian usually tends to avoid them. A random factor is also added to the overall evaluation of the desirability of every cell.

In the usual floor field models, after a deterministic elaboration of the utility of each cell, not comprising thus any random factor, the utilities are translated into the probabili-

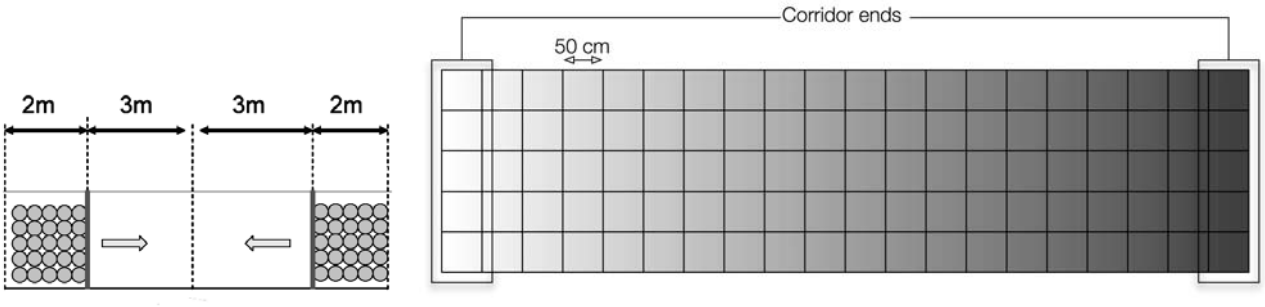


Figure 1: Schematic representation of a simple scenario: a 2.5 by 10 m corridor, with exits on the short ends and two sets of 25 pedestrians. The discretization of 50 cm and the floor field directing towards the right end is shown on the right.

ties that the related cell is selected as movement destination. This means that for a pedestrian generally there is a higher probability of moving towards his/her destination and according to proxemic considerations, but there is also the probability, for instance, to move away from his/her goal or to move far from his/her group. In this work, we decided to include a small random factor to the utility of each cell and to choose directly the movement that maximizes the agent utility. A more thorough comparison of the implications of this choice compared to the basic floor field approach is out of the scope of this paper and it is object of future works.

4. EXPERIMENTAL SCENARIO

The GA-Ped model was adopted to realize a set of simulations in different starting conditions (mainly changing density of pedestrians in the environment, but also different configurations of groups present in the simulated pedestrian population) in a situation in which experiments focused at evaluating the impact of the presence of groups of different size was being investigated.

4.1 Experiments

The environment in which the experiments took place is represented in Fig. 1: a 2.5 by 10 m corridor, with exits on the short ends. The experiments were characterized by the presence of two sets of 25 pedestrians, respectively starting at the two ends of the corridor (in 2 by 2.5 m areas), moving towards the other end. Various cameras were positioned on the side of the corridor and the time required for the two sets of pedestrians to complete their movement was also measured (manually from the video footage).

Several experiments were conducted, some of which also considered the presence of groups of pedestrians, that were instructed on the fact that they had to behave as friends or relatives while moving during the experiment. In particular, the following scenarios have been investigated: (i) single pedestrians (3 experiments); (ii) 3 couples of pedestrians for each direction (2 experiments); (iii) 2 triples of pedestrians for each direction (3 experiments); (iv) a group of six pedestrians for each direction (4 experiments).

One of the observed phenomena was that the first experiment actually required more time for the pedestrians to complete the movement; the pedestrians actually learned how to move and how to perform the experiment very quickly, since the first experiment took them about 18 seconds while the average completion time over 12 experiments is about 15 seconds.

	Indiv.		Couples		Triples		Groups of 5	
Den.	Sp.	Fl.	Sp.	Fl.	Sp.	Fl.	Sp.	Fl.
0,4	1,54	0,62	1,55	0,62	1,47	0,59	-	-
0,8	1,33	1,06	1,41	1,12	1,32	1,05	1,14	0,91
1,2	1,14	1,37	1,19	1,43	1,12	1,35	0,98	1,18
1,6	0,95	1,52	0,99	1,59	0,93	1,49	0,83	1,32
2,0	0,73	1,46	0,78	1,56	0,74	1,47	0,66	1,32
2,4	0,41	0,98	0,41	0,99	0,44	1,06	0,42	1,00
2,8	0,22	0,60	0,23	0,64	0,25	0,70	0,24	0,66
3,2	0,13	0,42	0,14	0,46	0,16	0,50	0,14	0,46

Table 1: Simulation Results: values on average speed (meters per second) and flow (persons/m·s), considering different densities (persons per square meter) of pedestrians and different configurations of groups.

The number of performed experiments is probably too low to draw some definitive conclusions, but the total travel times of configurations including individuals and pairs were consistently lower than those not including groups. Qualitative analysis of the videos showed that pairs can easily form a line, and this reduces the friction with the facing group. Similar considerations can be done for large groups; on the other end, groups of three pedestrians sometimes had difficulties in forming a lane, retaining a triangular shape similar to the 'V' shaped observed and modeled in [14], and this caused a total travel times that were higher than average in two of the three experiments involving this type of group.

4.2 Simulation Results

We applied the model described in Sect. 3 to the previous scenario by means of an agent-based platform based on GA-Ped approach. A description of the platform can be found in [5]. We employed the gathered data and additional data available in the literature to perform a calibration of the parameters, essentially determining the relative importance of (a) the goal oriented, (b) general proxemics and (c) group proxemic components of the movement choice. In particular, we first identified a set of plausible values for the w_t and w_o

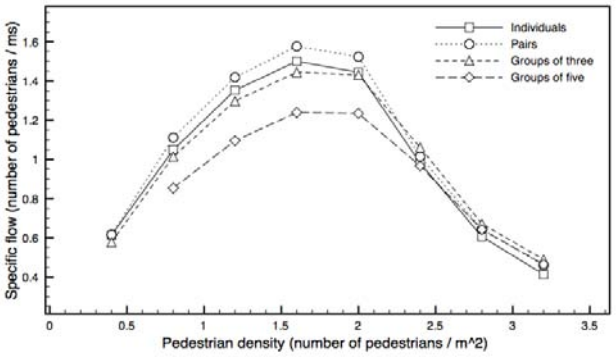


Figure 2: Fundamental diagram for different configurations of pedestrian based on simulation results in Table 1.

parameters employing experimental data regarding a one-directional flow. Then we employed data from bidirectional flow situations to further tune these parameters as well as the value of the w_g parameter: the latter was set in order to achieve a balance between effectiveness in preserving group cohesion and preserving aggregated measures on the overall pedestrian flow (an excessive group cohesion value reduces the overall pedestrian flow and produces unrealistic behavior).

We investigated the capability of our model to fit the fundamental diagram proposed in the literature for characterizing pedestrian simulations [22] and other traffic related phenomena. This kind of diagram shows how the average velocity of pedestrians varies according to the density of the simulated environment. Moreover, we wanted to distinguish the different performance of different agent types, and essentially individuals, members of pairs, groups of three and five pedestrians over a relatively wide spectrum of densities. To do so, we performed continuous simulations of the bidirectional pedestrian flows in the corridor with a changing number of pedestrians, to alter their density. For each density value displayed in the graph shown in Figure 2 is related to at least 1 hour of simulated time.

The achieved fundamental diagram represents in qualitatively correct way the nature of pedestrian dynamics: the flow of pedestrians increases with the growing of the density of the corridor unit a critical value is reached. If the system density is increased beyond that value, the flow begins to decrease significantly as the friction between pedestrians make movements more difficult.

An overview on the results of the simulations are shown in Table 1 in which values on average speed and flow, considering different densities of pedestrians and different configurations of groups are presented.

The simulation results are in tune with the experimental data coming from observations: in particular, the flow of pairs of pedestrians is consistently above the curve of individuals. This means that the average speed of members of pairs is actually higher than the average speed of individuals. This is due to the fact that they easily tend to form a line, in which the first pedestrian has the same probability to be stuck as an individual, but the follower has a generally higher probability to move forward, following the path "opened" by the first member of the pair. The same does not happen for larger groups, since for them it is more difficult to form a line: the curves related to groups of three and five members are below the curve of individuals for most of the spectrum of densities, precisely until very high density values are reached. In this case, the advantage of followers overcomes the disadvantage of offering a larger profile to the counter flow and the combined average velocity is higher than that of individuals.

5. REAL WORLD SCENARIO

5.1 Environment and observations

The model was also adopted to elaborate different what-if scenarios in a real world case study. In particular, the simulated scenario is characterized by the presence of a station of the Mashaer line, a newly constructed rail line in the area of Makkah. The goal of this infrastructure is to reduce the congestion caused by the presence of other collective means of pilgrim transportation (i.e. buses) during the Hajj: the yearly pilgrimage to Mecca that involves over 2 millions of people coming from over 150 countries and some of its phase often result in congestions of massive proportions. In this work, we are focusing on a specific point of one of the newly constructed stations, Arafat I. One of the most demanding situations that the infrastructure of the Mashaer Rail line must be able to sustain is the one that takes place after the sunset of the second day of the pilgrimage, which involves the transport of pilgrims from Arafat to Muzdalifah. The pilgrims that employ the train to proceed to the next phase of the process must be able to move from the tents or other accommodation to the station in an organized flow that should be consistent with the movement of trains from Arafat to Muzdalifah stations. Since pilgrims must leave the Arafat area before midnight, the trains must continuously load pilgrims at Arafat, carry them to Muzdalifah, and come back empty to transport other pilgrims.

The size of the platforms was determined to allow hosting in a safe and comfortable way a number of pilgrims also exceeding the potential number of passengers of a whole train. Each train is made up of 12 wagons, each able to carry 250 passengers for a total of approximately 3000 persons. In order to achieve an organized and manageable flow of people from outside the station area to the platforms, the departure process was structured around the idea of waiting–boxes: pilgrims are subdivided into groups of about 250 persons that are led by specific leaders (generally carrying a pole with signs supporting group identification). The groups start from the tents area and flow into these fenced queuing areas located in immediately outside the station, between the access ramps. Groups of pilgrims wait in these areas for an authorization by the station agents to move towards the ramps or elevators. In this way, it is possible to stop the flow of pilgrims whenever the number of persons on the platforms (or on their way to reach it using the ramps or elevators) is equal to the train capacity, supporting thus a smooth boarding operation.

Three photos and a schematic representation of the real

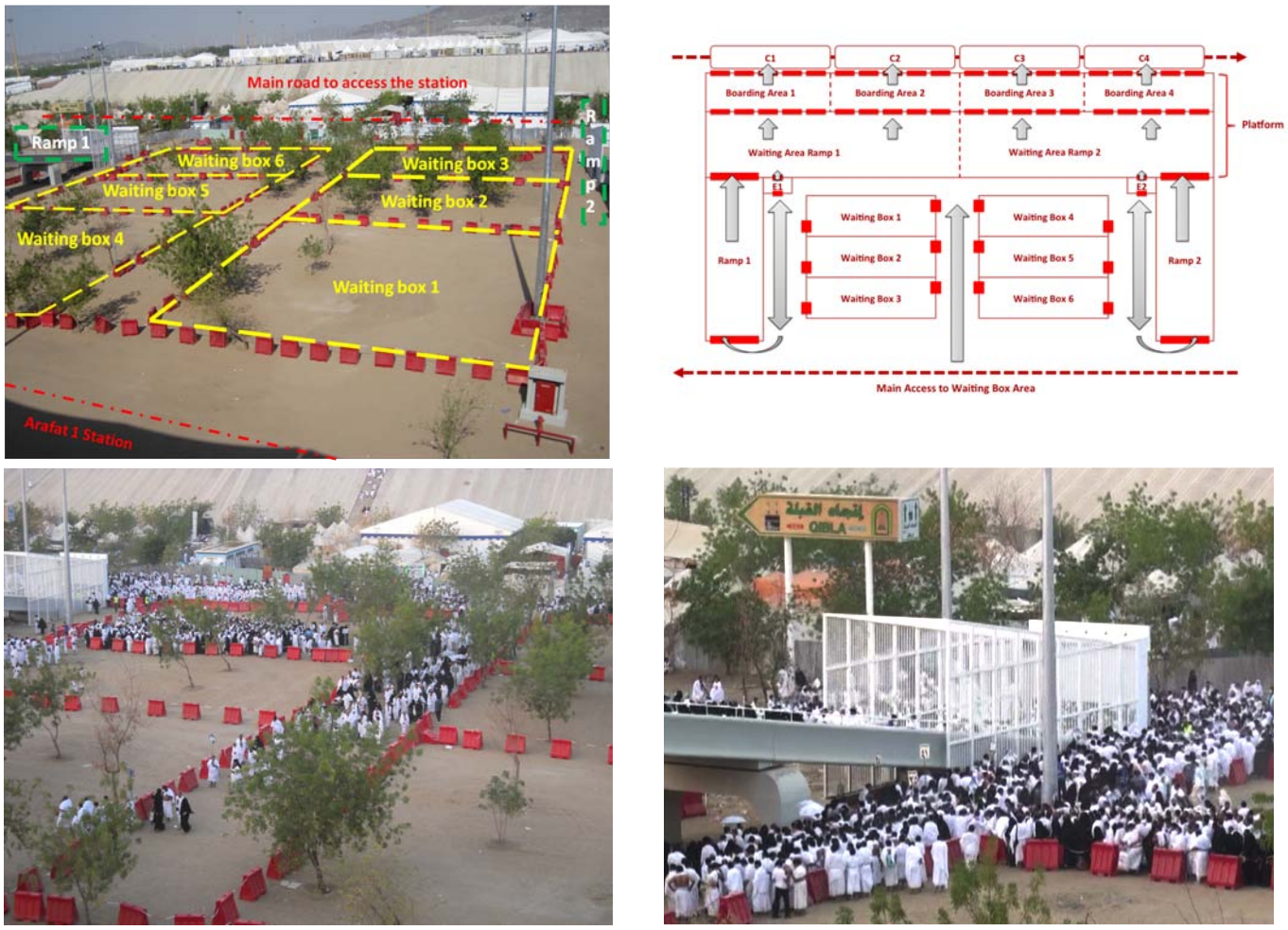


Figure 3: Photos and a schematic representation of the real world scenario and the related phonomena.

world scenario and the related phenomena are shown in Figure 3: the bottom right photo shows a situation in which the waiting-box principle, preventing the possibility of two flows simultaneously converging to a ramp, was not respected, causing a higher than average congestion around the ramp. This anomaly was plausibly due to the fact that it was the first time the station was actually used, therefore also the management personnel was not experienced in the crowd management procedures.

5.2 Simulation Results

Three different scenarios were realized adopting the previously defined model and using the parameters that were employed in the previous case study: (i) the flow of a group of pilgrims from one waiting box to the ramp; (ii) the simultaneous flow of two groups from two different waiting boxes to the same ramp; (iii) the simultaneous flow of three groups of pilgrims, two as in the previous situation, one coming directly from the tents area. Every group included 250 pilgrims. The goal of the analysis was to understand if the model is able to qualitatively reflect the increase in the waiting times and the space utilization when the waiting box principle was not respected.

The environment was discretized adopting 50cm sided cells and the cell space was endowed with a floor field leading towards the platform, by means of the ramp. The different speed of pedestrians in the ramp was not considered: this scenario should be therefore considered as a best case situation, since pilgrims actually flow through the ramp more slowly than in our simulation. Consequently, we will not discuss here the changing of the travel time between the waiting boxes and the platform (that however increased with the growth of the number of pilgrims in the simulated scenario), but rather different metrics of *space utilization*. This kind of metric is tightly related to the so called *level of service* [7], a measure of the effectiveness of elements of a transportation infrastructure; it is also naturally related to proxemics, since a low level of service is related to a unpleasant perceived situation due to the invasion of the personal (or even intimate) space.

The diagrams shown in Figure 4 report three metrics describing three different phenomena: (i) a situation in which an agent in a cell of the environment was willing to move but it was unable to perform the action due to the excessive space occupation; (ii) a situation in which an agent actually moved from a cell of the environment; (iii) the "set sum"

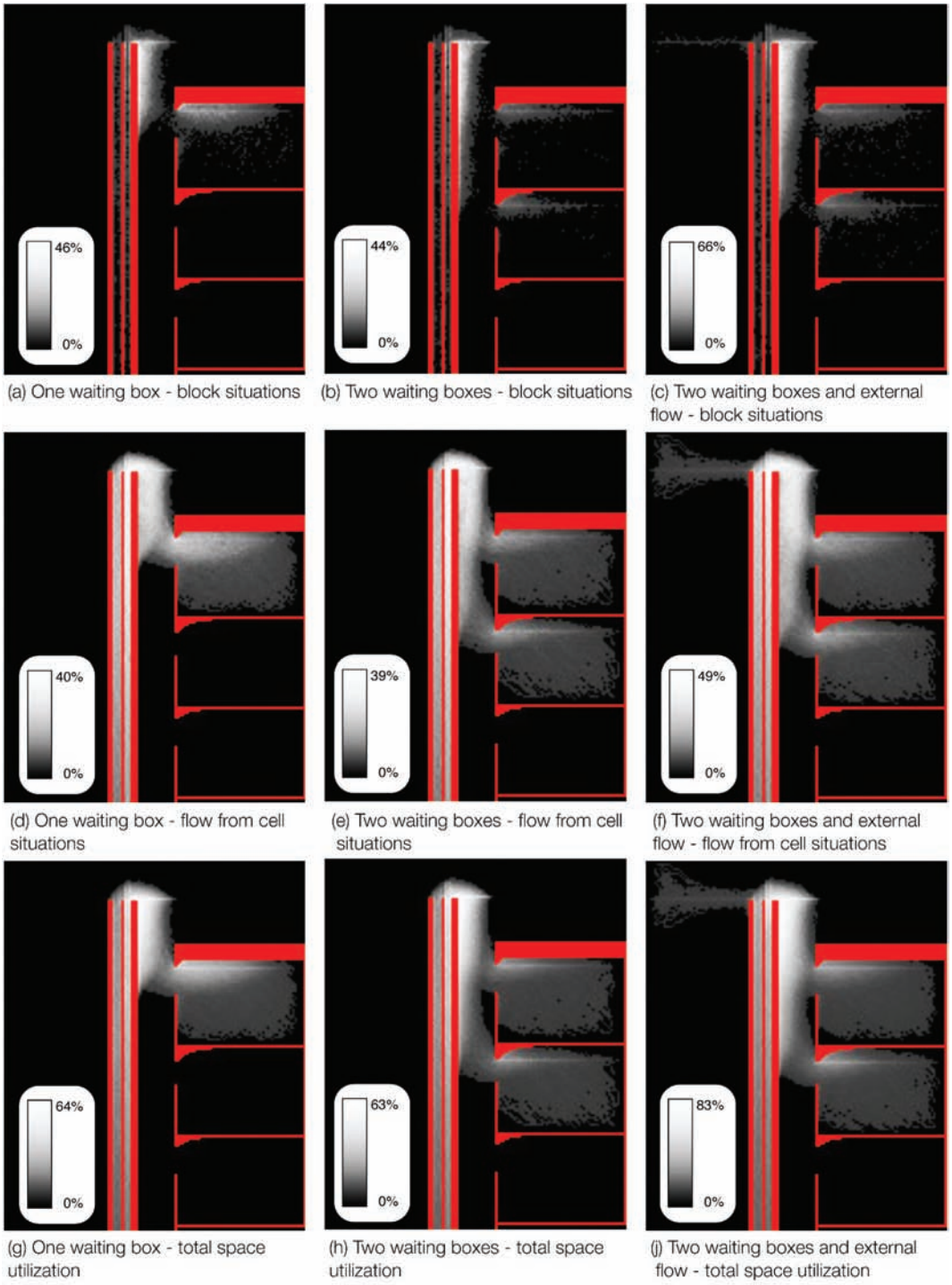


Figure 4: Space utilization diagrams related to the three alternative simulated scenarios.

of the previous situations, in other words, the situations in which a cell was occupied by agent, that either moved out of the cell or remained stuck in there. More precisely, diagrams show the relative frequency of the above events on the whole simulation time. The three metrics are depicted graphically following the same approach: the background color of the environment is black and obstacles are red; each point associated to a walkable area (i.e. a cell of the model) is painted in a shade of gray according to the value of the metric in that specific point. The black color is therefore associated to point if the environment in which the related metric is 0; the white color is associated to the point in which the metric assumes the highest value in the scenario (also shown in the legend). For instance, in all diagrams in the third row the points of space close to the ramp entrance are white or light gray, while the space of the waiting area from which

the second group starts is black in the first column, since the group is not present in the related situation and therefore that portion of space is not actually utilized.

The difference between the first and second scenario is not apparent in terms of different values for the maximum space utilization metrics (they are actually slightly lower in the second scenario), but the area characterized by a medium-high space utilization is actually wider in the second case. The third scenario is instead characterized by a noticeably worse performance not only from the perspective of the size of the area characterized by a medium-high space utilization, but also from the perspective of the highest value of space utilization. In particular, in the most utilized cell of the third scenario, an agent was stuck about 66% of the simulated time, compared to the 46% and 44% of the first and second scenarios.

This analysis therefore confirms that increasing the number of pilgrims that are simultaneously allowed to move towards the ramp highly increases the number of cases in which their movement is blocked because of overcrowding. Also the utilization of space increases significantly and, in the third situation, the whole side of the ramp becomes essentially a queue of pilgrims waiting to move towards the ramp. Another phenomenon that was not highlighted by the above diagrams is the fact that groups face a high pressure to mix when reaching the entrance of the ramp, which is a negative factor since crowd management procedures adopted in the scenario are based on the principle of preserving group cohesion and keeping different groups separated. According to these results, the management of the movement of group of pilgrims from the tents area to the ramps should try to avoid exceptions to the waiting box principle as much as possible.

6. CONCLUSIONS

The paper has discussed a research effort aimed at investigating the implication of the presence of groups in pedestrian and crowd dynamics. In particular, the paper has shown a sample situation in which data coming from experimental observations were used to calibrate and validate a simulation model that correctly captures some aspects of the impact of groups in the overall system dynamics. The validation was performed considering both travel times and other data gathered in actual experiments and also by comparing the achieved fundamental diagram with existing results from the literature. In addition, a real-world case study was also described: this work considered a train station in which different policies for crowd management were compare adopting space utilization metrics. The achieved results are in tune with observations carried out on the field and the model is able to reproduce phenomena related to group behaviours in pedestrian simulation.

Future works are aimed at modeling and simulating more complex group structures, such as hierarchical group structures (e.g. families, friends, elderly with accompanying persons inside larger groups) and their implications on overall system dynamics, validating results both quantitatively and qualitatively with specific reference to the morphology assumed by the group in medium and high density situations.

7. REFERENCES

[1] S. Bandini, S. Manzoni, and G. Vizzari. Situated cellular agents: a model to simulate crowding dynamics. *IEICE Transactions on Information and Systems: Special Issues on Cellular Automata*, E87-D(3):669–676, 2004.

[2] S. Bandini, F. Rubagotti, G. Vizzari, and K. Shimura. An agent model of pedestrian and group dynamics: Experiments on group cohesion. In R. Pirrone and F. Sorbello, editors, *AI*IA*, volume 6934 of *Lecture Notes in Computer Science*, pages 104–116. Springer, 2011.

[3] M. Batty. Agent based pedestrian modeling (editorial). *Environment and Planning B: Planning and Design*, 28:321–326, 2001.

[4] V. J. Blue and J. L. Adler. Cellular automata microsimulation of bi-directional pedestrian flows. *Transportation Research Record*, 1678:135–141, 2000.

[5] A. Bonomi, L. Manenti, S. Manzoni, and G. Vizzari. Makksim: Dealing with pedestrian groups in mas-based crowd simulation. In G. Fortino, A. Garro, L. Palopoli, W. Russo, and G. Spezzano, editors, *WOA*, volume 741 of *CEUR Workshop Proceedings*, pages 166–170. CEUR-WS.org, 2011.

[6] R. Challenger, C. W. Clegg, and M. A. Robinson. Understanding crowd behaviours: Supporting evidence. `http://www.cabinetoffice.gov.uk/news/understanding-crowd-behaviours`, 2009.

[7] J. J. Fruin. *Pedestrian planning and design*. Metropolitan Association of Urban Designers and Environmental Planners, 1971.

[8] E. T. Hall. *The Hidden Dimension*. Anchor Books, 1966.

[9] D. Helbing and P. Molnár. Social force model for pedestrian dynamics. *Phys. Rev. E*, 51(5):4282–4286, May 1995.

[10] C. M. Henein and T. White. Agent-based modelling of forces in crowds. In P. Davidsson, B. Logan, and K. Takadama, editors, *Multi-Agent and Multi-Agent-Based Simulation, Joint Workshop MABS 2004, New York, NY, USA, July 19, 2004, Revised Selected Papers*, volume 3415 of *Lecture Notes in Computer Science*, pages 173–184. Springer–Verlag, 2005.

[11] H. Klüpfel. *A Cellular Automaton Model for Crowd Movement and Egress Simulation*. Phd thesis, University Duisburg-Essen, 2003.

[12] L. Manenti, S. Manzoni, G. Vizzari, K. Ohtsuka, and K. Shimura. Towards an agent-based proxemic model for pedestrian and group dynamic. In A. Omicini and M. Viroli, editors, *WOA*, volume 621 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2010.

[13] S. Manzoni, G. Vizzari, K. Ohtsuka, and K. Shimura. Towards an agent-based proxemic model for pedestrian and group dynamics: Motivations and first experiments. In Tumer, Yolum, Sonenberg, and Stone, editors, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems – Innovative Applications Track (AAMAS 2011)*, pages 1223–1224, 2011.

[14] M. Moussaïd, N. Perozo, S. Garnier, D. Helbing, and G. Theraulaz. The walking behaviour of pedestrian social groups and its impact on crowd dynamics. *PLoS*

ONE, 5(4):e10047, 04 2010.

[15] S. R. Musse and D. Thalmann. Hierarchical model for real time simulation of virtual human crowds. *IEEE Trans. Vis. Comput. Graph.*, 7(2):152–164, 2001.

[16] K. Nishinari, A. Kirchner, A. Namazi, and A. Schadschneider. Extended floor field ca model for evacuation dynamics. *Special Issue on Cellular Automata of IEICE Transactions on Information and Systems*, E84-D, 2003.

[17] S. Paris and S. Donikian. Activity-driven populace: A cognitive approach to crowd simulation. *IEEE Computer Graphics and Applications*, 29(4):34–43, 2009.

[18] F. Qiu and X. Hu. Modeling group structures in pedestrian crowd simulation. *Simulation Modelling Practice and Theory*, 18(2):190 – 205, 2010.

[19] R. A. Rodrigues, A. de Lima Bicho, M. Paravisi, C. R. Jung, L. P. Magalhães, and S. R. Musse. An interactive model for steering behaviors of groups of characters. *Applied Artificial Intelligence*, 24(6):594–616, 2010.

[20] S. Sarmady, F. Haron, and A. Z. H. Talib. Modeling groups of pedestrians in least effort crowd movements using cellular automata. In D. Al-Dabass, R. Triweko, S. Susanto, and A. Abraham, editors, *Asia International Conference on Modelling and Simulation*, pages 520–525. IEEE Computer Society, 2009.

[21] A. Schadschneider, A. Kirchner, and K. Nishinari. Ca approach to collective phenomena in pedestrian dynamics. In S. Bandini, B. Chopard, and M. Tomassini, editors, *Cellular Automata, 5th International Conference on Cellular Automata for Research and Industry, ACRI 2002*, volume 2493 of *Lecture Notes in Computer Science*, pages 239–248. Springer, 2002.

[22] A. Schadschneider, W. Klingsch, H. Klüpfel, T. Kretz, C. Rogsch, and A. Seyfried. Evacuation dynamics: Empirical results, modeling and applications. In R. A. Meyers, editor, *Encyclopedia of Complexity and Systems Science*, pages 3142–3176. Springer, 2009.

[23] M. Schreckenberg and S. D. Sharma, editors. *Pedestrian and Evacuation Dynamics*. Springer–Verlag, 2001.

[24] W. Shao and D. Terzopoulos. Autonomous pedestrians. *Graphical Models*, 69(5-6):246–274, 2007.

[25] J. Tsai, N. Fridman, E. Bowring, M. Brown, S. Epstein, G. A. Kaminka, S. Marsella, A. Ogden, I. Rika, A. Sheel, M. E. Taylor, X. Wang, A. Zilka, and M. Tambe. Escapes - evacuation simulation with children, authorities, parents, emotions, and social comparison. In Tumer, Yolum, Sonenberg, and Stone, editors, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems – Innovative Applications Track (AAMAS 2011)*, pages 457–464, 2011.

[26] J. Was. Crowd dynamics modeling in the light of proxemic theories. In *ICAISC (2)*, pages 683–688, 2010.

[27] A. Willis, N. Gjersoe, C. Havard, J. Kerridge, and R. Kukla. Human movement behaviour in urban spaces: Implications for the design and modelling of effective pedestrian environments. *Environment and Planning B*, 31(6):805–828, 2004.

[28] S. Xu and H. B.-L. Duh. A simulation of bonding effects and their impacts on pedestrian dynamics. *IEEE Transactions on Intelligent Transportation Systems*, 11(1):153–161, 2010.

Micro-Simulation Model for Assessing the Risk of Car-Pedestrian Road Accidents

Gennady Waizman
Department of Geography and Human Environment,
Tel Aviv University
Ramat Aviv
Tel Aviv, 69978, Israel
+972-54-2012193
gennadyw@post.tau.ac.il

Shraga Shoval
Department of Industrial Engineering and Management,
Ariel University Center of Samaria
Milken Campus,
Ariel, 40700, Israel
+972-3-9066325
shraga@ariel.ac.il

Itzhak Benenson
Department of Geography and Human Environment,
Tel Aviv University
Ramat Aviv
Tel Aviv, 69978, Israel
+972-3-6409896
bennya@post.tau.ac.il

ABSTRACT
The data on traffic accidents clearly points to "Black Spots" that continually cause a high rate of accidents. However, road safety research is still far from understanding why this particular place on a road is risky. The reason is the deficit of knowledge of how pedestrians and drivers interact when facing a potentially dangerous traffic situation, and in the lack of an integrated framework that relates the data on human behavior to real-world traffic situations. We tackle the problem by developing SAFEPED, a multi-agent microscopic 3D simulation of cars' and pedestrians' dynamics at the black spot. SAFEPAD is a test platform for evaluating experimentally estimated drivers' and pedestrians' behavioral rules and estimating accident risks in different traffic situations. It aims to analyze disadvantageous design of the Black Spot and to assess alternative architectural solutions.

Categories and Subject Descriptors
I.6.5 Computing Methodologies, Simulation and Modeling, Model Development

General Terms
Algorithms, Design, Reliability, Experimentation, Human Factors, Standardization

Keywords
Traffic accidents, Black Spot, agent-based modeling, spatially-explicit modeling

1. INTRODUCTION

1.1 Micro-simulation of road accidents between the cars and pedestrians: from static to dynamics view

Accident statistics reveal factors of risk and establish the dependencies of accident rates on the characteristics and parameters of roads, cars, pedestrians, traffic and the environment of the accident location [1, 2, 3]. However, statistical models are inherently static and, thus, unable to reflect the chain of events that cause an accident [4]. The static view of the accident may explain the persistent fraction of the "black spot" - seemingly regular road locations with an unexpectedly high and stable accident rate [5] – but cannot be used for assessing the consequences of changes in the infrastructure or traffic conditions at the location.

Treatment of a specific black spot is typically based on an engineers' insight of the local conditions. The effectiveness of the safety measures is confirmed by comparing the accident rates before and after the implementation of safety measures. Successful implementations are usually reported, such as the installation of the several hundred countdown signals at the crossings in San Francisco, that reduced the number of pedestrian injuries caused by crashes with vehicles by 52% [6]; or the system for detecting pedestrians approaching a crosswalk zone and warning the drivers of pedestrian presence [7].

However, failures are often not reported. Traffic engineers lack tools for assessing the proposed safety measures, and say nothing about their economic justification. Safety measures are costly, while their success is not guaranteed. As a result, urban decision-makers have essential difficulties when deciding on changes in traffic regulations and infrastructure, even when the location is identified as a black spot.

The development of a dynamic simulation model of traffic accidents at a black spot provides a solution to this problem. Using this model, the chain of events (based on the behavior of the vehicles and pedestrians) that caused the accident can be investigated. This paper presents the pilot version of such a model.

1.2 Field studies of the accident micro-dynamics

Last decade a series of large-scale studies aimed at developing reliable indicators of vehicle pre-crash conditions were performed within the framework of the Intelligent Transportation Systems program of the U.S. Department of Transportation. The research focused on "last second" urgent maneuvering, and resulted in significant amounts of data collected during real-time observations of driver behavior and car movement [8, 9, 10, 11], as well as during simulator-based driving [12, 13]. On-road data includes kinematic characteristics of the vehicle, real-time measurements of the distance to the other objects, and video of driver's behavior. Laboratory experiments aimed to study drivers' behavior in potential accident scenarios, such as a lane-change maneuver.

The above studies provided important information on vehicle-vehicle interaction in pre-accident and accident situations. However, vehicle – pedestrian interaction were beyond the focus of the program, therefore the recorded number of vehicle –

pedestrian incidents and behavior of the participants, was low [10, 12].

In parallel, computer-based analysis of the videos taken on the roads became popular and provided essential knowledge on pedestrian decision-making when cars were approaching, as well as in more complex situations. These studies are employed for developing static, discreet choice models that describe the probability of road crossing or other action based on the distance to approaching car, or its velocity and road geometry [14, 15, 16, 17, 18]

1.3 Modeling car-pedestrian conflict

Usually agent-based (AB) models focus on either vehicle or pedestrian traffic and avoid combining these two flows within the same model. The major reasons are inherent behavioral differences between pedestrians and drivers in regard to route choice and compliance with traffic regulations. Popular models of car traffic, such as VISSIM, PARAMICS, SUMO or Aimsun [19] use an intentionally simplified view of pedestrians. Models of crowding specify pedestrian interactions but ignore details of vehicle traffic [15].

The model of pedestrians' disobedience to traffic laws at the crosswalks [20] is a rare example of a dynamic model of car-pedestrian interactions. It is based on Cellular Automata and combines the vehicle flow sub-model of Nagel-Schreckenberg [21] with the pedestrian sub-model. However, Cellular Automata's view of space inherently restricts agents' movement to relatively large cells introduced for describing vehicle flows and is too rough for microscopic representation of pedestrian motion.

In this paper, we propose SAFEPAD – a high-resolution, spatially explicit dynamic simulation model as a tool for forecasting the effects of changes in traffic environments. SAFEPAD is based on the continuous representation of space and the objects' movements, and in this respect follows the recent approaches and achievements in robotic algorithms for motion planning and collision avoidance. It is a spatially-explicit agent-based model that explicitly represents spot infrastructure and moving objects in fine 3D detail, and operates at a time resolution of 1/20 of a second. Behavioral rules of SAFEPAD agents – vehicles and pedestrians – are based, when possible, on the experimental data.

2. SAFEPED, the Agent-Based model of car-pedestrian interactions

AB techniques provide the basis for modeling vehicular-pedestrian conflict [22, 23]. By dynamically simulating the behavior of every car and pedestrian (represented by the precise 3D models) within a precise 3D model of the spot infrastructure, the researcher is able to record agents' actions and their outcome (e.g., an accident). This model identifies risk factors and investigates the effectiveness of proposed safety measures.

The advantages of the AB approach for modeling and studying traffic accidents are numerous. Results of experiments on the behavior of participants can be directly interpreted in terms of agents' behavioral rules, which can be used by the simulation model to assess an infinite number of scenarios with different numbers of cars and pedestrians of various kinds, and behaviors and in various environmental and architectural settings. The frequency and severity of accidents can then be quantitatively projected for any situation. The goal of our research is to develop the AB model of car-pedestrian interaction at a specific spot as a tool for assessing, planning and engineering decisions of road safety. The user of SAFEPED can change the 3D geometry of the spot and characteristics of the traffic flow, and then assess whether the proposed changes will decrease accident rate and severity.

The motion behavior rules of the SAFEPED agents follow the robotic approach to real-time motion planning and maneuvering for vehicles and pedestrians. These rules account for basic imperfections of human visual perception, limitations in pedestrian locomotion and car mobility, and are based on the robotic algorithms of motion in a dynamic environment proposed by Fiorini and Shiller [24].

SAFEPED is a working prototype that works at a high time resolution of 1/20 of a second. At each time step, agents, considered in a random order or priority, decide on their motion behavior for the next time step and perform it.

2.1 The 3D presentation of the spot

SAFEPED is built on precise 3D representation of the Black spot's land surface and infrastructure including road borders, parked cars, pedestrian crossings, buildings, trees, traffic lights and signs (Figure 1). Combined with the orthophoto, this provides realistic representation of the spot geometry.

Figure 1: SAFEPAD model scene showing agents' trajectories

2.2 SAFEPED agents and their behavior

SAFEPED simulates movement of both drivers and pedestrians, acting in a 3D environment. Drivers and pedestrians behave autonomously according to a set of probabilistic behavioral rules. Each agent, driver or pedestrian, is assigned an agent's profile that includes height, width, velocity, steering and acceleration/deceleration capabilities.

2.2.1 Agents' motion at a macro-level:

Each SAFEPAD agent tries to maintain the desired velocity, and aims to follow a predefined trajectory, shown in Figure 1 as a blue dashed line for a vehicle and red dashed line for pedestrian. However, it is often impossible to follow the trajectory because of other moving and stationary objects. In this example, driver and pedestrian agents react, not necessarily adequately, to the behavior of the other autonomous agents when they see them. The agent, driver or pedestrian, decides whether to deviate from the trajectory to the left or to the right, accelerate, decelerate or even stop, and returns to the trajectory should the road conditions make it possible.

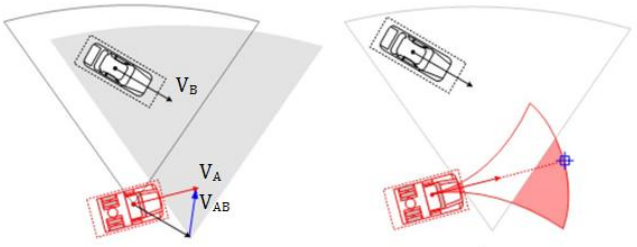
We choose the trajectory-based approach in order to reduce generating accident situations in which drivers or pedestrians

follow potentially dangerous paths. An agent enters the site at the end of one of the predefined trajectories and follows it, trying to maintain the desired velocity while taking into account the other agents and environmental elements (Figure 1). In addition, at every intersection of agents' trajectories, SAFEPED makes it possible to set decision-making priorities that reflect traffic rules and agreements. An agent moving along the continuous green path has priority over an agent moving along the continuous red path (Figure 1). When two agents, one on the continuous green path and the other on the continuous red path, approach the point of intersection of their trajectories and take account of each other (according to their movement decision rules), *both agents know* that an agent on the green path would act before the agent on the red path. Note that this includes the case when the agent on the green path decides that the agent on the red path is moving too fast, and rather than risk a potential collision, the agent on the green path decides to stop and give a way to the other agent. If the trajectories of two agents intersect and priorities are not assigned, both agents know there are no priorities (i.e. the order of their actions in the simulation will be random).

2.2.2 Agents' micro-behavior behavior in conflict situations

Road safety demands motion planning in dynamic environments, where cars and pedestrians should avoid dynamic and static obstacles. This is far more complex than the static problem and, in this case, robotics uses velocity space instead of the standard 3D space (referred to as "configuration space" in robotics). The problem of avoiding one or many mobile or immobile obstacles is treated directly in the velocity space, providing the trajectory which satisfies an optimization criterion. In our model, agents, drivers, and pedestrians follow robotic motion planning algorithms for dynamic environments. We employ the version of this algorithm that is proposed by Fiorini and Shiller [24]. One of the advantages of this algorithm is its applicability to a set of objects that essentially vary in their inherent velocities, vehicles and pedestrians in our case.

The algorithm considers Velocity Obstacle (VO) - the set of all velocities of a moving object that will result in a collision with another moving object at some moment in time, assuming that the other object maintains its current velocity. In our model, the concept of VO is applied for computation of avoidance maneuvers; accelerating/decelerating cars, and pedestrians that follow curvilinear trajectories (Figure 2).



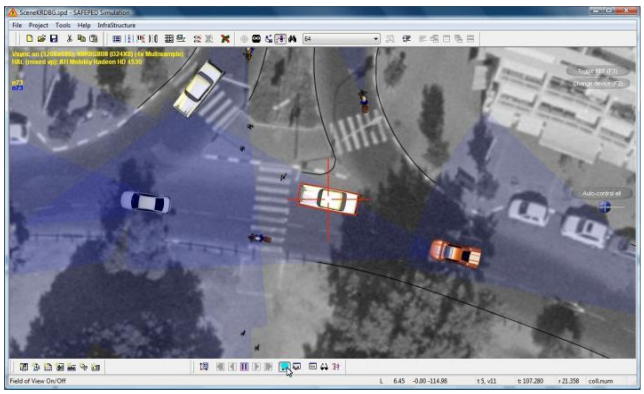
(a) (b)

Figure 2: An example of the avoidance maneuver algorithm as implemented in SAFEPED.

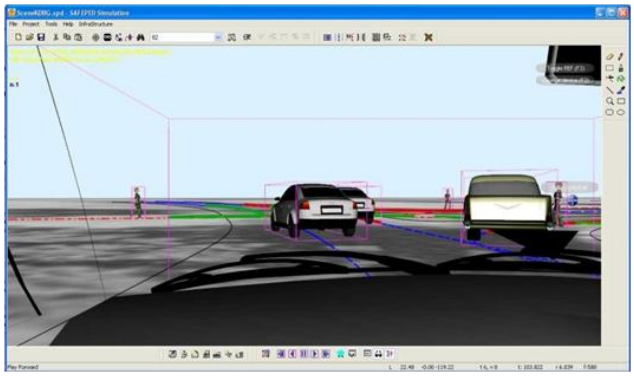
In Figure 2a, the red car is moving at a velocity of V_A, the black car at V_B and the red car is trying to avoid collision with the black car by changing its velocity. The white sector in Figure 2a denotes the set of *relative* velocities V_{AB} of the red car relative to the black car that will result in a collision. The white sector is constructed in the configuration space, taking into consideration the physical dimensions of each car (represented by the radius of the circumference circles of each car). The gray sector denotes the domain of the *absolute* velocities of the red car that leads to collision with the black car. The gray sector is a simple transformation of the white sector along V_B. In Figure 2b, the red domain denotes the set of available velocities of the red car, constrained by maximal possible acceleration of the car that guarantees no collision. This sector is constructed by subtracting the velocity obstacle domain that results in a collision (the gray sector) from the domain of all possible maneuvers of the red car. The blue point denotes a safe avoidance velocity for the red car that does not require a change in the car direction. If accident avoidance demands acceleration or deceleration that is beyond the human and car abilities, the red domain vanishes, and accident occurs.

2.2.3 Agents' vision

SAFEPED agents see the 3D environment within the "view cone" of up to 180^o angle (Figure 3a). In the pilot version of SAFEPED, agents are unaware of traffic lights, and this feature has yet to be added. We interpret the human visual system as a pinhole camera. The 3D shape (currently minimal 3D box) of each object within the view cone is projected on the retinal plan of the agent's "eye" (Figure 3b).



(a)



(b)

Figure 3: SAFEPED scene with the agents' view cones (blue); the car marked by cross is chosen for follow up (a); 3D visibility in the SAFEPED, the driver's view from the car (b).

Based on this information, an agent detects objects close to the line of sight, defines which objects are obscured by others, and to what degree. Objects that are fully obscured for 3 seconds become invisible to the agent, and the agent does not react to them. These objects are currently represented by parallelepipeds; a more precise representation of the objects by mesh technique is currently in development.

2.3 SAFEPED output and performance

All agents' actions are continuously recorded at every time step, and can be replayed. Possible types of accidents (head-on collision, one-sided collision, car-pedestrian collision, etc.) are defined and instantaneously checked. The model keeps track of agents' location, set of available velocities, eyesight behavior, decisions on velocity, distance to other agents, and acceleration/deceleration.

SAFEPED analysis of a typical site considers up to a hundred simultaneously moving agents. Even at a finest resolution of the spot and the agents' 3D geometry, we did not encounter any computational difficulties with the pilot version of the SAFEPED.

The first version of the SAFEPAD is ready for evaluation. For a general view see http://www.youtube.com/watch?v=ia3W8oiTVYw&feature=related. Our formalization of visibility is given by http://www.youtube.com/watch?v=6KFcfFREIt8&feature=related, and http://www.youtube.com/watch?v=axWEGNetpM0 illustrates a traffic accident.

3. Experiment with an obscured car

Following is an experiment with SAFEPAD that aims at testing agents' micro-motion algorithm in potentially risky situations.

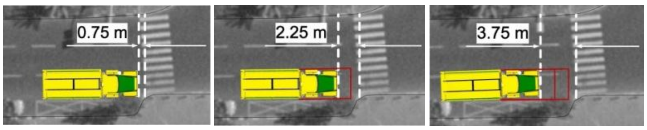
3.1 Experimental setup

The experimental setup is presented in Figure 4: the pedestrian crosses a multi-lane street on a non-regulated crossing.

Figure 4: Experimental setup: high truck A is stopped in the lane adjacent to the sidewalk and obscures the view of both the pedestrian and of approaching car B

High truck A is stopped in the lane closest to the sidewalk and obscures the pedestrian's view. Vehicle B approaches the crosswalk from the second lane and the view of the driver is obscured too. US Transportation Agency publication describes this situation as follows: "The pedestrian entered the traffic lane at midblock in front of standing or stopped traffic and was struck by another vehicle moving in the same direction as the stopped traffic" [25]. According to [3] multiple threat crashes comprise 17.6 percent of pedestrian crashes on marked crosswalks.

The actual road crossing between Weizmann St. and Moshe Sharet St. in Tel Aviv, Israel was chosen for constructing the 3D representation of a junction infrastructure. We investigate the emergence of the accident situations for three different locations of the obscuring high truck: at a distance of 0.75, 2.25 and 3.75 m from the crosswalk (Figure 5).



(a) (b) (c)

Figure 5: Three experimental situations: high truck parks at a distance of 0.75m (a), 2.25m (b) and 3.75m (c) from the crosswalk

We investigated the risk of contact between the car and the pedestrian, such as the pedestrian being hit by the car's right fender (Figure 6), as dependent on velocities and attention times of the car and pedestrian. According to [26] we set the pedestrian reaction time as 0.28 ±0.07 sec and driver reaction time as 0.70-0.75 sec. [27]. This includes all components of reaction, e.g. movement time of ~0.2 sec required to lift the foot from the accelerator and then to touch the brake pedal.

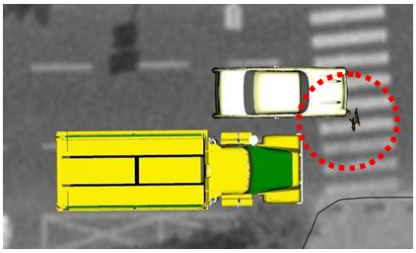


Figure 6: Car's right fender hits pedestrian when truck parks at 0.75m distance from the crosswalk.

3.2 When can each participant avoid the accident on its own?

Let us investigate the conditions in which pedestrian and driver may take control of the situation and are capable of avoiding a collision, even if the other participant chooses the worst line of action.

We start with a pedestrian that does not look around and crosses the street at a high speed of 6km/h (Table 1). In case of a truck stopped at 0.75 m form the crosswalk, the driver succeeds in noticing the pedestrian and stops safely when the truck's speed is lower than 12 km/h in case the pedestrian reacts slowly, and 13 km/h in case the pedestrian reacts fast. The reaction time of the pedestrian is based on estimates presented in [26] - 0.28 ±0.07 sec, and we used 0.28 - 0.07 = 0.21 sec and 0.28 + 0.07 = 0.35 sec as a reaction time for "fast" and "slow" pedestrian. Similarly, when the truck is located 2.25m and 3.75m from the crosswalk, the driver is able to stop if his/her speed is below 24-28km/h.

Table 1. Driver full control speed in case of inattentive pedestrian crossing at 6 km/h

Pedestrian's Reaction	Distance between truck and crosswalk		
	0.75 m	2.25 m	3.75 m
Slow	12 km/h	24 km/h	26 km/h
Fast	13 km/h	25 km/h	28 km/h

Let us now consider an ignorant driver driving at a speed of 50 km/h. To avoid an accident in case of a truck at 0.75 m, a slow reacting pedestrian must walk at 3.7 km/h or slower, while a fast reacting pedestrian can walk at speeds up to 4.6 km/h (Table 2).

For the two other positions of obscuring truck, a pedestrian walking at any reasonable speed is capable of detecting the car and stopping.

Table 2. Pedestrian full control speed in case of inattentive driver at 50 km/h

Pedestrian's Reaction	Distance between truck and crosswalk		
	0.75 m	2.25 m	3.75 m
Slow	3.7 km/h	5.1 km/h	Above 6.0 km/h
Fast	4.6 km/h	6.0 km/h	Above 6.0 km/h

Let us now focus on the most dangerous situation of close-by obscuring truck and investigate the case when, in order to avoid an accident, both the driver and pedestrian have to react to each other, i.e., when the driver's speed is above 12-13 km/h.

3.3 The situation in which both participants have to be careful

Figure 7 presents the maximal safe speeds for the car and pedestrian in the case of inattentive and attentive agents, as obtained in the model for the obscuring truck at a distance 0.75 m. As can be seen from the chart, attentive agents can move faster and avoid the accident. Pedestrian reaction is very important in this case. Slowly-reacting attentive pedestrian will be in danger if the car's speed is above 20 km/h, while the fast-reacting pedestrian is in danger if the car's speed is above 35 km/h. Note that to avoid a crash regardless of the car's speed, the pedestrian should not walk faster than 2 km/h.

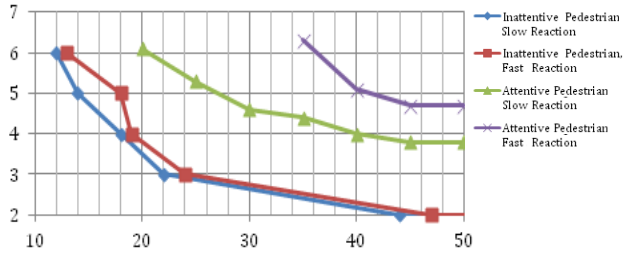


Figure 7: Maximal safe speeds for car and pedestrian with obscuring truck at a 0.75m distance

The crash is a qualitative event and, to be really safe, one needs to include essential margins to the estimates presented in Tables 1, 2 and in Figure 7. Let us estimate these margins.

3.4 Safe avoidance of crash

The situation in which the driver and pedestrian successfully avoided an accident by passing each other at a distance of 5 cm can be hardly considered safe. The human view of safe resolution of the accident demands a significant distance between the car and pedestrian during the entire period of their interaction.

In our experiments with SAFEPAD we have chosen 0.5 m as "minimal safe" distance between the car and pedestrian. We investigate only the case of a truck at 0.75 m, and present the worst case for a slowly reacting pedestrian. As can be seen from Table 3, the safe speeds are essentially lower than those that are required in order to avoid the accident.

To conclude, our model study confirms the importance of advanced stop lines on the road before crosswalk as an accident prevention measure. The simulations demonstrate that the distance between the advanced stop line and the crosswalk should be about 2m, higher than the intuitive estimate of the 1.5 m as proposed by [28].

Table 3. Minimal distance between the car and slowly reacting pedestrian, truck at the distance of 0.75m from the crosswalk

Pedestrian's speed, km/h	Car's speed, km/h						
	50	45	40	35	30	25	20
5.5	crash	crash	crash	crash	crash	crash	0.17
5.0	crash	crash	crash	crash	crash	0.06	0.16
4.5	crash	crash	crash	crash	0.10	0.19	0.29
4.0	crash	crash	crash	0.08	0.08	0.26	0.36
3.5	0.04	0.04	0.06	0.10	0.18	0.28	*0.47*
3.0	0.10	0.09	0.11	0.20	0.26	0.36	*0.45*
2.5	0.22	0.20	0.47	0.71	1.05	1.08	*0.47*
2.0	0.94	1.14	1.00	0.99	1.01	1.02	*0.42*

Shaded cells – unsafe speeds
Italic – pedestrian can stop and avoid accident on his/her own

4. Discussion

The proposed SAFEPED model is unlimited in "measuring" vehicular-pedestrian interaction in scenarios with a wide range of agents' behavior. High temporal and spatial resolution of the SAFEPAD, similar to that of driver simulators and real-time in-car equipment, provides high potential for combining it with field studies [8, 9, 10, 11, 12, 13]. SAFEPED can serve as a tool for assessing accident risks at specific spots, and can identify measures to decrease these risks.

By direct assignment of human-based behavioral rules to the model agents, SAFEPED is capable of implementing arbitrarily cognitive-perceptual parameters of drivers' and pedestrians' behavior, including strategic and tactical behavioral components.

References

[1]. Campbell, B.N., Najm, W. G. et al, 2003, Examination of Crash Contributing Factors Using National Crash Databases, National Highway Traffic Safety Administration, US Department of Transportation, DOT- HS 809 664, DOT-VNTSC-NHTSA-02-07

[2]. Chang, D., 2008, National Pedestrian Crash Report. : National Highway Traffic Safety Administration, US Department of Transportation,, DOT HS 810 968, Washington, DC

[3]. Zegeer, C.V., Stewart J.R., Huang H.H., et al, 2005, Safety Effects of Marked Versus Unmarked Crosswalks at Uncontrolled Locations, Final Report and Recommended Guidelines, US Department of Transportation, Federal Highway Administration, HRT-04-100, 112 .

[4]. Archer, J. 2005, Indicators for the traffic safety assessment and prediction and their application in micro-simulation modeling, PhD Thesis, Royal Institute of Technology, Sweden.

[5]. Gitelman, V., Balasha D., et al, 2012, Characterization of pedestrian accidents and an examination of infrastructure measures to improve pedestrian safety in Israel, Accident Analysis and Prevention, 44(1), 63-73

[6]. Markowitz, F., Sciortino, S., Fleck, J.L., Yee, B.M., 2006, Pedestrian Countdown Signals: Experience with an Extensive Pilot Installation, Institute of Transportation Engineers Journal, 76(1), 43-48

[7]. Hakkert A.S., Gitelman V., Ben-Shabat E., 2002, An evaluation of crosswalk warning systems: effects on

pedestrian and vehicle behavior, Transportation Research F, 5(4), 275-292

[8]. Kiefer, R. J., Cassar, M.T, et al, 2003, Forward Collision Warning Requirements Project: Refining the CAMP Crash Alert Timing Approach by Examining Last-Second Braking and Lane Change Maneuvers Under Various Kinematic Conditions, National Highway Traffic Safety Administration, US Department of Transportation, DOT HS 809 574

[9]. Klauer, S.G., Dingus, T. A., et al, 2006a, The Impact of Driver Inattention On Near-Crash/Crash Risk: An Analysis Using the 100-Car Naturalistic Driving Study Data, National Highway Traffic Safety Administration, US Department of Transportation, Field Experiment DOT HS810 594

[10]. Klauer, S. G., Dingus, T. A., et al, 2006b, The 100-Car Naturalistic Driving Study, Phase II - Results of the 100-Car Field Experiment National Highway Traffic Safety Administration, US Department of Transportation, DOT HS 810 593

[11]. FHWA, 2007, Evaluation of the Volvo Intelligent Vehicle Initiative Field Operational Test, Final Report, Safety Administration, US Department of Transportation.

[12]. Smith, D. L., Najm, W. G., Lam, A. H., 2003, Analysis of braking and steering performance in car-following scenarios, Proceedings of 2003 SAE world congress, SAE technical paper series 01-0283.

[13]. Najm, W., Smith, D., 2004, Modeling driver response to lead vehicle decelerating, SAE Technical Paper Series, 01-0171, 1-10,

[14]. Sun, D., Ukkusuri, S.V.S.K., Benekohal, R.F., Waller, S.T., 2003, Modeling of motorist pedestrian interaction at uncontrolled mid-block crosswalks. Proceedings of the 82d Annual Meeting of the Transportation Research Board, 03-3340

[15]. Papadimitriou, E., Yannis, G., Golias, J., 2009, A critical assessment of pedestrian behaviour models, Transportation Research F, 12(3), 242-255

[16]. Wang, T, Wu, J, et al, 2010, Study of pedestrians' gap acceptance behavior when they jaywalk outside crossing facilities, 13th International IEEE Conference Intelligent on Transportation Systems, 1295-1300, 19-22

[17]. Pratoa, C.G., Gitelman, V., Bekhor, S., 2012, Mapping patterns of pedestrian fatal accidents in Israel, Accident Analysis and Prevention, 44(1), 56-62

[18]. Robin, T., Antonini, G., Bierlaire, M., Cruz, J., 2009, Specification, estimation and validation of a pedestrian walking behavior model, Transportation Research B, 43(1), 36-56

[19]. Ishaque , M. M., R. B. Noland, 2008, Behavioural Issues in Pedestrian Speed Choice and Street Crossing Behaviour: A Review, Transport Reviews 28(1), 61-85

[20]. Zhang, Y., Duan, H., 2007, Modeling Mixed Traffic Flow at Crosswalks in Micro-Simulations Using Cellular Automata. Tsinghua Science And Technology, 12(2), 214-222

[21]. Nagel, K., Schreckenberg, M., 1992, A Cellular Automaton Model for Freeway Traffic, Journal de Physique I, 2, 2221-2229

[22]. Benenson, I. and P. M. Torrens, 2004, Geosimulation: automata-based modeling of urban phenomena .London, Wiley.

[23]. Benenson, I., S. Aronovich and S. Noam, 2005, Let's Talk Objects: Generic Methodology for Urban High-Resolution Simulation, Computers, Environment and Urban Systems, 29, 425-453.

[24]. P Fiorini, Z Shiller Motion planning in dynamic environments using velocity obstacles1998, The International Journal of Robotics Research 17(7), 760-772

[25]. FHWA, 2009, Crash-type manual for pedestrians, Highway Safety Research Center, University of North Carolina, FHWA-RD-96-104

[26]. Hoogendoorn, S.P., Daamen, W., Landman., R, 2005, Microscopic calibration and validation of pedestrian models: cross-comparison of models using experimental data. In Waldau, N., Gattermann, P., Knoflacher, H, Schreckenberg, M, Eds, Pedestrian and evacuation dynamics, 253-265

[27]. Green, M., 2000, How Long Does It Take to Stop? - Methodological Analysis of Driver Perception-Brake Times, Transportation Human Factors, 2(3), 195-216

[28]. Redmon, T., 2011, Evaluating Pedestrian Safety Countermeasures, Public Roads Magazine, 74(5), FHWA-HRT-11-003

Modeling the conscious behavior of drivers for multi-lane highway driving

Yi Luo and Ladislau Bölöni
Dept. of Electrical Engineering and Computer Science
University of Central Florida
Orlando, Florida
yiluo@mail.ucf.edu, lboloni@eecs.ucf.edu

ABSTRACT

The current state of the art in simulating highway driving extensively relies on models using formulas similar to those describing physical phenomena such as forces, viscosity or potential fields. While the parametrization of these formulas can account for the limitations of the driver (such as reaction delay), they are badly suited for modeling conscious behavior. In this paper we describe our simulation architecture which uses an agent-based model to represent the conscious tactical and strategic behavior of the agent. This model will act as a high level input to a state-of-the-art virtual physics model which models the physical vehicle and the subconscious aspects of the driver behavior.

The concrete aspects of driving modeled in this paper are the strategic lane preferences of the drivers, with a special attention to the optimal lane positioning for a safe exit. We have used the model to simulate the traffic on Orlando's Highway 408. The results match well with the real world traffic data. The increased simulation detail can be applied to crash prediction and the control of intelligent transportation system devices, such as variable speed limits.

1. INTRODUCTION

Existing microscopic traffic simulation models heavily rely on mathematical formulas similar to those describing various physical phenomena: forces, viscosity, potential fields and so on. We will call these *virtual physics* models. Over the course of the last fifty years there was a gradual shift from formulas relying on fluid dynamics towards the individual treatment of the vehicle as a particle subject to a collection of forces.

These models have been proved predict well the integrative, long term parameters of the traffic, such as throughput or average speed in congested traffic. These values are highly useful for making long-term decisions such as highway planning. Their level of detail, however, is insufficient to model events depending on specific driver decisions – such as the incidence of crashes. On the other end of the spectrum, we find purely agent based simulators such as the NetLogo [9] based `http://ccl.northwestern.edu/netlogo/models/Traffic2Lanes`. These efforts are successful as proofs of concepts, yet their realism and simulation accuracy is arguably lower than state of the art virtual physics models.

Our work is centered on improving the accuracy of microscopic highway simulation through agent based modeling of the *conscious* aspect of the driver behavior. These type of models are sometimes called "nanoscopic" traffic simulations [6, 3]. Lower level behavior, such as the vehicle physics, the driver's reflexive action, and those aspects of the driver's behavior which have been learned to the point of becoming automated will be handled by the virtual physics model augmented to allow for the integration of the agent based component. For the starting point of the contributions described in this paper see [5].

The conscious part of the driver's behavior can be classified into strategic and tactical behavior. Strategic behavior involves decisions which are planned for the overall success of the drive (safe and fast arrival to the destination). Examples involve route planning, joining or leaving convoys, and choosing the appropriate highway lanes. Tactical behavior includes actions taken to achieve short term advantages: overtaking a slow moving vehicle, escaping from a dangerous situation, increasing the distance from an erratically moving vehicle and so on. Our technical approach will be to separate the behavior of the driver into three simulation modules, as described in Figure 1.

The virtual physics model models the physics of the vehicle as well as those aspects of the driver which are either reflexive (such as emergency braking) or learned to the point of becoming sub-conscious (such as lane following and keeping a constant distance from the car in front). Our current model is based on [5], but we shall investigate other models as well.

The agent model models the conscious cognition of the human driver. This includes both strategic planning (which exit to take, which lane to prefer for long distance driving) and tactical (the decision to join a convoy or overtake a slow moving car). The agent model will receive input from the environment (including sensor data, signaling data, vehicle-to-vehicle and vehicle-to-infrastructure communication). The agent model acts *through* the virtual physics model, by tem-

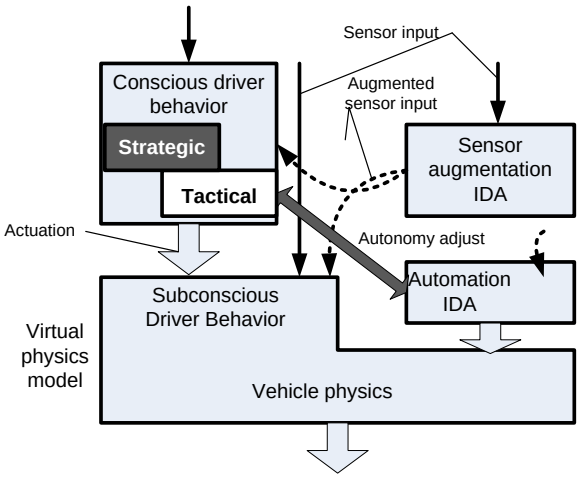


Figure 1: Overall architecture of the simulation model integrating the virtual physics model, the human agent and the automation model.

porarily changing its parameters, which, when the action is finished, will return to default values.

The automation model which models the action of the driver assist technologies, such as intelligent cruise control, emergency brakes, lane following and others. This component *replaces* the virtual physics model with a separate control system. The transitions between the virtual physics and the automation model need to model the real world transition of control between driver control and automation.

Due to space limitations, this paper will concentrate on a single, important aspect of strategic behavior, the planning, decision and execution of lane changes. For an even more concrete focus, we describe in detail the planning for a safe exit from a congested highway - which requires a number of lane changes ahead of the exit. It was found that about 10% of the crashes occurring on highways are *sideswipe crashes* while about 11% of them are *angle crashes* [7]. Both types are associated with lane changes (the reminder of the crashes are mostly rear-end crashes). Modeling the mechanics of this process is of a major importance as it can predict traffic simulations with high crash risk.

The reminder of this paper is organized as follows. Section 2 describes the virtual physics models which are the baseline of the contribution described in this paper. Section 3 describes the model through which the agent's preferences for specific lanes are enacted. Section 4 introduces a probabilistic model of success for lane changes. In Section 5 we apply the model to the problem of safe exit/merge from highways. We apply our work on the simulated traffic of Orlando's Highway 408 in the real world traffic data. We conclude in Section 6.

2. VIRTUAL PHYSICS-BASED MODELS

As shown in Figure 1, our agent-based driver model is closely integrated with and acts through the virtual physics model. To motivate this architecture, and to provide the foundation for the presentation of the agent model, we will briefly describe a collection of technologies which together are a good sample of the state of the art in virtual physics based models. These components will be used in our system to model vehicle physics and subconscious driver behavior. The virtual physics model has three main components: a time-continuous car following model, a lane change model and a human driver model.

2.1 Car following models

Car following models describe the behavior of a car on a single lane highway. Most such models calculate the acceleration or deceleration of the car though a formula of the following general pattern:

$$\frac{dv_i(t)}{dt} = f(\Delta x_i, v_i, \Delta v_i) \tag{1}$$

where $\Delta x_i = x_{i+1}(t) - x_i(t)$ is the distance between the vehicle and its immediate leader, and $\Delta v_i = v_i(t) - v_{i+1}(t)$ is the approaching speed. The specific formula we choose to use is the one introduced by Treiber et al. [10]:

$$\frac{dv_i(t)}{dt} = a\left[1 - \left(\frac{v_i}{v_0}\right)^4 - \left(\frac{\delta(v_i, \Delta v_i)}{\Delta x_i}\right)^2\right] \tag{2}$$

where a is the maximum acceleration of the vehicle, v_0 is the desired speed, and $\delta(.)$ is the desired distance from the leading vehicle. This distance depends on a number of parameters through the following formula:

$$\delta(v_i, \Delta v_i) = \Delta x_{min} + v_i T + \frac{v_i \Delta v_i}{2\sqrt{ab}} \tag{3}$$

where Δx_{min} is the minimum distance in case of congestion ($v_i = 0$), T is the safe time headway which models the buffering time of the driver, and b is the comfortable deceleration, which couldn't be less than -9 m/s^2 on a dry road.

Let us now discuss the intuitions behind this formula. On a free road, the instant acceleration changes from the maximum acceleration a (when the vehicle is still $v_i = 0$) to 0 (when the vehicle reaches its desired speed $v_i = v_0$). If a vehicle follows a leader with a negligible approaching speed ($\Delta v_i \approx 0$), the term $v_i T$ in Equation 3 dominates such that the vehicle maintains a distance $v_i T$ from the leader.

In the situation when the vehicle approaches the leader with a high speed, the last term $v_i \Delta v_i / 2\sqrt{ab}$ dominates and the formula dictates a deceleration. The most extreme case is when the vehicle moves with its desired speed v_0 and observes a still obstacle at the distance of x_i. To avoid a collision, the vehicle must brake with deceleration $-b$ when it reaches a distance of $\Delta x_i = v_i^2/2b$. Indeed, this is exactly what the model predicts:

$$\frac{dv_i(t)}{dt} = -a\left(\frac{\delta}{\Delta x_i}\right)^2 = -a\frac{\left(\frac{v_i \Delta v_i}{2\sqrt{ab}}\right)^2}{\Delta x_i^2} = -\frac{v_i^4}{4b\Delta x_i^2} = -b \tag{4}$$

The car following model, defined in this way is considered *collision free*.

2.2 Lane changing models

Our baseline model extends the car following model with the lane change model described by Kesting et al. [2]. This model assumes that lane changes happen instantaneously: for a shift to the left lane, a vehicle which has been previously in the middle lane, at time t disappears from the middle lane and appears in the left lane. This opens the possibility that a car, coming from behind in the new lane with a higher speed can not break sufficiently quickly and collides with the lane changing car. The model assumes that it is the responsibility of the lane changing car to ensure that the rear left vehicle $j-1$ has sufficient buffer distance such that it can decelerate before hitting the lane changing vehicle

$$\hat{a}_{j-1}(t) \geq -b_{max} \tag{5}$$

If this condition is not satisfied, the vehicle concludes that it is not safe to change lanes.

The second feature of the lane changing model is the analysis of the motivations to change lanes, and the "politeness of the drivers". We assume that the goal of the vehicles is to achieve their desired speed, which implies a certain desired acceleration $\hat{a}_i$. The motivation of the driver to change lanes is such that it can achieve this acceleration (which, we assume, is not achievable in the current lane). However, the changing of lanes might also trigger accelerations in the other vehicles: for instance, it allows the current follower to accelerate, and it might force the new follower to brake.

The notion of *politeness* models the fact that the driver might consider the accelerations of the other vehicles as well when taking a decision to change the lane. The politeness parameter p specifies how much does the vehicle discount the other vehicles' desired acceleration compared to its own. A value $p = 0$ indicates an impolite, fully selfish driver which does not care about other drivers (however, it still considers the safety criteria). The vehicle i will decide to change the lane if the following inequality is verified:

$$(\hat{a}_i + p \cdot (\hat{a}_{j-1} + \hat{a}_{i-1}) - (a_i + p \cdot (a_{i-1} + a_{j-1})) \geq \Delta p_{th} \tag{6}$$

where Δp_{th} is the politeness threshold. The left hand side is the difference between the new accelerations $\hat{a}_i$, $\hat{a}_{j-1}$ and $\hat{a}_{i-1}$ if the vehicle i successfully changes into the target lane and the old accelerations a_{i-1} and a_{j-1} if it doesn't change lane. The intuition is that the vehicle favors to change lane only when the advantage of the action is greater than the disadvantage it exerts to its neighboring vehicles. However, because the vehicle i can not obtain the parameters (T, v_0, a, b) for its successor $i-1$ and $j-1$, the utility of lane change can only be calculated by vehicle i's own parameters.

2.3 Human driver model in the virtual physics approach

A human driver is in some aspects "less capable", but in other aspects "more capable" than the abstract driver envisioned in the models considered up to this point. State of the art microscopic traffic models consider some aspects of the human driver such as reaction time, fatigue and cognitive limitations and integrate them in the equations of the virtual physics model.

For instance, our baseline model inspired from Treiber et al. [11] implements the following aspects. First, we consider the fact that humans can not perform an indefinite number of decisions per unit of time. This is modeled by considering a time step Δt. At every time step Δt the drivers observe the traffic and make a decision about acceleration. This acceleration value will remain constant for the next interval Δt:

$$\begin{aligned} v_i(t + \triangle t) &= v_i(t) + \dot{v}_i(t)\triangle t \\ x_i(t + \triangle t) &= x_i(t) + v_i(t)\triangle t + \frac{1}{2}\dot{v}_i(t)\triangle t^2 \end{aligned} \tag{7}$$

Another aspect of the human behavior modeled is the reaction time T' necessary to reason about the traffic situation and make decisions accordingly. This can be achieved by substituting in Equation 1 the current state $(\Delta x_i, v_i, \Delta v_i)$ at time $t - T'$. If $t - T'$ falls between two simulation steps, then it will be adjusted as:

$$x(t - T') = \beta x_{t-n-1} + (1 - \beta)x_{t-n} \tag{8}$$

2.4 A critique of virtual physics models

Virtual physics integrate physical aspects (such as maximum acceleration a and maximum breaking b) with psychological aspects such as desired speed v_0, and even cognitive limitations such as the reaction time t_r. A well tuned virtual physics model can provide a good simulation of the overall flow of the traffic. It can not, however, model well the details of specific situations.

For instance, the model presented above assumes that the only justification for a lane change is to achieve a more favorable acceleration. This obviously covers only short term behavior, but even then, it fails to account for some important aspects of driver behavior, such as the preference to overtake on the left side or the tendency to return to the preferred lane after overtaking. The model completely ignores strategic lane change behavior, such as merging into traffic, moving to a preferred lane, positioning to the right lane for a forking highway and the preparation for exit.

Let us consider the issue of politeness as described in the model above. A driver might act politely towards cars which are trying to merge into the traffic from a merging lane which is shortly terminating. The same driver might aggressively pursue its goal of changing lanes when this is necessary for him to make the desired exit. The problem is not with the physical expression of the politeness, but with the fact that this politeness is modulated by higher level cognitive acts, which can not be modeled as forces.

3. STRATEGIC LANE CHANGE BEHAVIOR

Many highway simulation models assume that the lane change decision is based on a near-term optimization criteria. The vehicles will change lanes if they can get closer to their desired speed. This, of course is only true under the ideal assumption of an infinitely long highway, with no road signs or obstacles and drivers who have no preconceived ideas about the traffic lanes.

In a real world traffic, especially for highways traversing cities, however, there are a number of considerations which affect this behavior:

- **Entrances:** the drivers enter the highway on the rightmost lane which often serves as a temporary merging lane. The drivers need to merge into traffic before the lane ends.
- **Exits:** when drivers exit the highway, they need to position themselves to the appropriate exit lane (usually one or two rightmost lanes, but occasionally a leftmost lane). Depending on the traffic, the approaching maneuver must be started long before the exit.
- **Avoid the rightmost lane.** If the highway has more than two lanes, and there is a zone with many entrances and exits, then most drivers prefer not to drive on the rightmost lane, to avoid interference with cars entering and exiting the highway.
- **Leftmost lanes as high speed lanes.** The leftmost lane is usually deemed a high-speed lane and is avoided by vehicles which drive slower by choice or necessity (such as trucks). Vehicles which are pushing the posted speed limits, however, are preferring the leftmost lane.
- **Lane number variations.** The number of lanes on the roads changes with the location. Lanes terminate, new lanes are added in busy areas. The termination of lanes is usually signalled ahead.
- **High occupancy vehicle lanes.** Some highways designate the rightmost lane as a high occupancy vehicle lane. This would naturally be a preference for qualifying vehicles, but it also requires the traversal of many other lanes for entrance and exit.

Beyond the conditions imposed by the highway configuration, the lane change behavior also depends on the *strategies* of the individual drivers. Some drivers might try to reduce the number of lane changes, while others make them every time it might offer a short term advantage. Some drivers prefer to position themselves to the correct exit lane long time ahead, while others might wait to the last minute to move towards the exit. Some drivers prefer the leftmost lane, while others try to avoid it and prefer middle lanes.

In this paper we introduce a framework which models the static and dynamic lane preferences of the drivers. The framework integrates with the virtual physics based models described in the previous section - it does not replace but augments them. The preference model does not eliminate the optimization for the desired speed from the sources of driver decision. For instance, in an open highway with the planned exit far away, speed optimization might trump the preferences for certain lanes. When approaching the desired exit, however, positioning to a preferred lane gradually takes priority.

This agent-based model of traffic simulation allows us to study aspects of traffic which are impossible with previous models. Examples of the kind of questions we can answer are:

- Are highway exits which are close to each other a help or hinder to the smoothness of traffic?
- How does a left exit changes the shape of traffic?
- Do drivers which wait for the last moment to move for the exit lane help or hinder traffic? What about their performance (time to destination?) Their safety? Other's safety? Overall driving comfort?
- Do drivers who prefer the inside lane move faster?

We start by defining our notion of utility of a lane. The first idea would be to use the left hand side of Formula 6 as the utility metric. This value, however, can be negative: it's range is $[-C, C]$ where

$$C = (a + b_{max})\,(1 + p) \tag{9}$$

We need, however, a strictly positive utility metric for the further definitions. To achieve this, we add C to the formula. Thus the utility of the current, left and right lanes will be defined as:

$$\begin{aligned} U_c &= \Delta p_{th} + C \\ U_l &= (\hat{a}_i + p \cdot (\hat{a}_{j-1} + \hat{a}_{i-1}) - (a_i + p \cdot (a_{i-1} + a_{j-1})) + C \\ U_r &= (\hat{a}_i + p \cdot (\hat{a}_{h-1} + \hat{a}_{i-1}) - (a_i + p \cdot (a_{i-1} + a_{h-1})) + C \end{aligned}$$

The preference model modifies the virtual physics model by assigning the preference value $W_c \in [0.0, 1.0]$ to the lanes of the road. The preference values are assigned to the individual lanes based on a longer term planning process. The virtual physics model will consider the *weighted utilities* of the lanes $U_c^w = W_c \cdot U_c$ and so on.

This way, the vehicle might not move to a low priority lane even if that would confer a temporary advantage. Yet, the agent's behavior would still retain the smoothness associated with the virtual physics model. When all the lanes have the same preference, the behavior reverts to the basic virtual physics model.

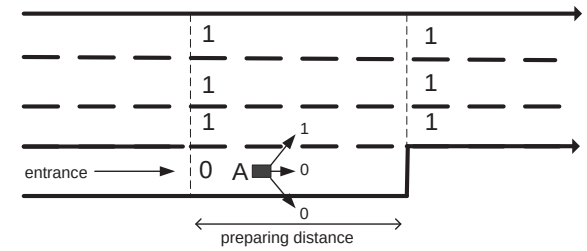
The preference weights are directly associated to the lanes of the highway, yet the vehicle needs to make decisions *one lane change at a time*. Thus the vehicle occasionally needs to accept a decrease in utility in order to reach a preferred lane after more lane changes.

To resolve this problem, we define the lane change preferences as follows. W_c is the preference of the vehicle's current lane. W_l and W_r are the *maximum* of all the preferences to the left and right of the vehicle, respectively.

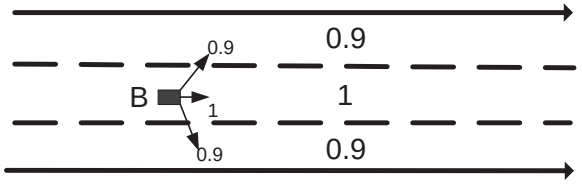
Let's now consider some examples of the use of the preferences by the agent:

i) When *entering* the highway, the agent will set the preference of its terminating entrance lane to zero. This will cause it to move to the highway's continuing lanes as soon as it is safe (see Figure 2(a)).

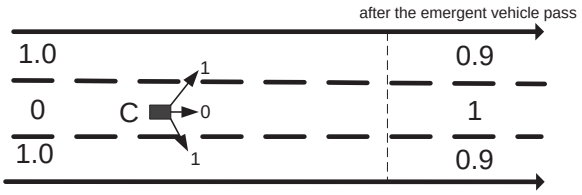
ii) When *driving* on the highway, the vehicle will assign higher preference to the lanes it prefers driving on. The



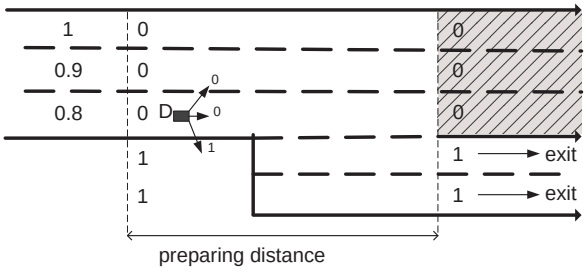
(a) Entering



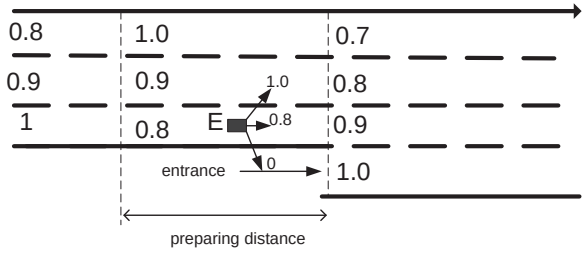
(b) Driving



(c) Give a way



(d) Exit



(e) Avoid busy lane

Figure 2: The agent tries to evaluate the preferences of lane changes.

preference gradients will be, however, milder. This allows the other components of the simulation to override this behavior, if significant advantage is to be gained - or if the tactical maneuver requires it (see Figure 2(b)).

iii) When the vehicle needs to *"give a way"* to a police car or emergency vehicle, it will set the specific lane(s) to zero preference, which will force it to move to one of the non-zero preference lane as soon as it is safe. Once the emergency vehicle has passed, the vehicle resets its lane preferences to the previous ones (see Figure 2(c)).

iv) If the vehicle prepares to *exit*, it will modify the lane preferences to prefer the exit lane. Note that this does not mean that the vehicle will immediately change to the exit lane, as a number of other safety conditions need to be satisfied for each lane change(see Figure 2(d)).

v) *Avoiding entering lanes.* Let us consider a vehicle which is on inter-city routes prefers to drive on the rightmost lanes. These lanes, however become extremely busy before and after exits with cars which are entering and exiting the highway. Thus many drivers prefer to the left side of the road around the exit to their default preferences after the merge finished. This shows in Figure 2(e). Note that this preference has again relatively mild gradient, and can be overwritten by other considerations.

3.1 Modeling lane changes under different conditions

The default virtual physics model assumes lane changes happened as the result of an *opportunity*. In real life, however, there are cases where the vehicle is *forced* to change lanes, even if the change doesn't improve the utility. For instance, the agent needs to give up its fast left lane in order to exit, or it needs to give way to an emergency vehicle.

These situations are modeled in our simulator by setting the preference of the current lane to zero. Even in these cases, although the vehicle wants to change lanes, it might not be able to, because the safety conditions will not be satisfied.

We define the *time to change lane* the amount of time between the moment when the weighted utility dictates a lane change until the moment when the lane change is safely accomplished. If the vehicle is not able to change the lane before the utilities are reversed (or the lane ends), we say that the vehicle *missed the lane change*.

To mimic the behavior of human agents in such situations, we introduced a *speed adaptation technique*. The safety condition for lane change is more likely to be satisfied if the vehicle modifies the speed such that it matches the one of the desired destination lane. Thus, under situations of *forced lane change*, the vehicle will change its desired speed to the current speed of the neighboring vehicle in the destination lane. If the vehicle needs to cross several lanes (as the case of the exit) it will change its desired speed in steps, always adapting it to the speed of the next destination lane. Once the forced lane change situation is terminated, the desired speed of the vehicle reverts to the one dictated by the virtual physics model.

4. A PROBABILISTIC MODEL OF SUCCESS FOR LANE CHANGES

Many drivers prefer to drive during most of the journey on the faster lane on the left side of the highway. To finish the journey, however, they need to exit from the rightmost lane. Thus, for most drivers, exiting the highway is a maneuver which requires several consecutive forced lane changes. In situations of heavy traffic, this can represent a significant safety risk.

Different drivers approach the problem of exit differently. Some prepare a long time ahead, moving towards the rightmost lanes. This, however, increases congestion on those lanes. Others remain on the fast lanes until the last moment – this however, requires several successive lane changes with very little room for error.

In this section, we describe in detail the considerations of preparation to exit, based on the lane change preference model introduced in the previous section. Similar considerations apply for the case when a vehicle needs to merge from a lane which soon will terminate.

Probability of successful lane change: The need to prepare in advance for exit is due to the fact that a driver who intends to perform a lane change might not be able to execute it for a certain amount of time.

The difficulty of the lane change depends on the local density of the vehicles in the target lane D_i and the average speed difference between the vehicle and the neighboring vehicles in the target lane ΔV_i. An experienced driver can estimate $Pr(t, D_i, \triangle V_i)$ - the probability that it can successfully change lanes in time t for a specific value of the density and speed difference. For the purpose of our simulation, we have collected this data by identifying lane change events in the simulator logs. The probability was extracted from the histograms of the time it took to actually perform the lane change.

Probability of successful exit If the vehicle is currently n lanes away from the exit lane, it will need to successfully execute n lane changes before exit. The driver needs to start its exit preparations at such a time / distance ahead so that it can successfully exit with a certainty (high probability).

In the rest of this paper we will use 90% for this probability value. This value requires some explanation, as it appears to be low: it would imply that 10% of the drivers will miss their exits. In reality, only a much smaller number of misses happen. What will happen in practice is that either (a) some of the other drivers will change their behavior such that they allow the vehicle to exit or (b) the vehicle will move even if the safety conditions are not satisfied. Note that case (b) does not immediately imply a crash, only a dangerous situation.

Let us now analyze how a driver can calculate the preparation time necessary for a safe exit with 90% certainty. Suppose we have $Pr(t_s, D_i, \triangle V_i)$- the probability of a single lane change which is finished at time t when the next lane i has density D_i and speed difference $\triangle V_i$. In general, if the agent tries to change from lane i to j in time n, the probability that it can succeed is

$$Pr(i,j,n) = \begin{cases} \sum_{t=1}^{n-(j-i)+1} Pr(t, D_{i+1}, \triangle V_{i+1}) Pr(i+1, j, n-t) & i < j \\ \sum_{t=1}^{n-(i-j)+1} Pr(t, D_{i-1}, \triangle V_{i-1}) Pr(i-1, j, n-t) & i > j \\ 1 & i = j \end{cases} \quad (10)$$

The probability of successful change across multiple lanes can be calculated through a recursive algorithm. As the probability of successful exit is monotonically (but not linearly) increasing with the time of exit preparation, we can find the minimum preparation time necessary to achieve any given successful exit probability through binary search in the space of calculated probabilities.

For a driver it is usually easier to tie the exit preparation to a specific distance to the exit rather than to a specific time to exit, as the current distance to the exit is usually easy to estimate from the information on the road signs. The "time to prepare" can be converted into "distance to prepare" by simply estimating the average speed of the vehicles on the lanes separating the vehicle from the exit lane.

Using these algorithms we can envision a fictional *optimal exit model*. This driver would first observe the relative speeds and densities in all the lanes which separate the vehicle from the exit lane. Then, using the calculations outlined above, the driver would be able to calculate the optimal time when it needs to start its exit maneuver (for a specific value of safe exit probability).

5. EXPERIMENTAL RESULTS

5.1 Simulation parameters

For the experimental study we have run experiments using our simulator which implements the virtual physics and agent models. The agent model also includes a number of tactical behavior components not discussed in this paper (such as communication through signaling), which ensures a higher accuracy and realism of the overall simulation. The experiments have been performed on a detailed, lane-by-lane model of a 22.13 mile stretch of Highway 408. Inflow and outflow information was acquired from the statistics of the expressway authority[1]. The vehicle inflow was modeled as a Poisson traffic, matching the specified average inflow rate. The statistical data, however, does not provide an explicit mapping between the point where a specific vehicle enters and leaves the highway. Thus, for our model, we choose exit points for the vehicle stochastically, with the probability that the vehicle entering at entrance i will have a destination at exit j being:

$$Pr(j) = \frac{Out(j)}{Out(j) + \sum_{k>j} Out(k) - \sum_{l>j} In(l)} \quad (11)$$

where $In(l)$ is the inflow rate of entrance with label l, and $Out(k)$ is the outflow rate of exit with label k. The denominator in the Equation 11 is the total number of vehicles which will pass or exit the location. However, the selection

[1]http://www.expresswayauthority.com/Corporate/ about-Statistics/HistoricalTraffic.aspx

Table 1: Default parameters of the simulation

Parameter	Symbol	Value
simulation step	Δt	$0.1s$
maximum deceleration	b_{max}	$5.0m/s^2$
vehicle length	x_{length}	$4m$
minimum distance	Δx_{min}	$2m$
acceleration	a	$1.5m/s^2$
desired deceleration	b	$2.0m/s^2$
headway time	T	$1.5s$
desired speed	v_0	$105km/h \pm 20\%$
politeness	p	0.5
politeness threshold	Δp_{th}	0.2
visibility range	$x_{visibility}$	$400m$
reaction time	T'	$0.4s$
lane change time	t_{lane}	$2.0s$

probability is calculated with the assumption that the vehicle doesn't exit before j, so we need to normalize them as

$$Pr(i,j) = \prod_{i<m<j} (1 - Pr(m))Pr(j) \qquad (12)$$

To simulate the highway in the rush hour, we increase the inflow and outflow rate by the *flow ratio*. The parameters of the simulation are summarized in Table 1.

For the following experiments we will study two different types of vehicle behavior with the same virtual physics model but different agents. The SIG agent does not change the speed of the vehicle when trying to change lane. In contrast, the VAR agent is changing its desired speed to match the destination lane, according to the technique described in Section 3.1.

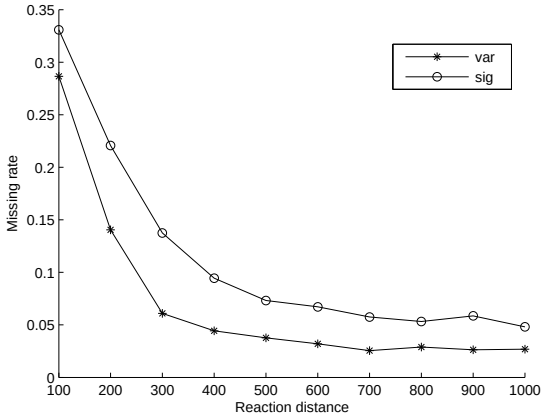
5.2 Rate of exit misses function of the exit preparation distance

In this experiment, we study the rate of the exit misses (or, in a different interpretation, of the dangerous exits) in function of the distance where the vehicles start their preparation for exit by changing their lane preferences to prefer the exit lane (as in Figure 2(d)).

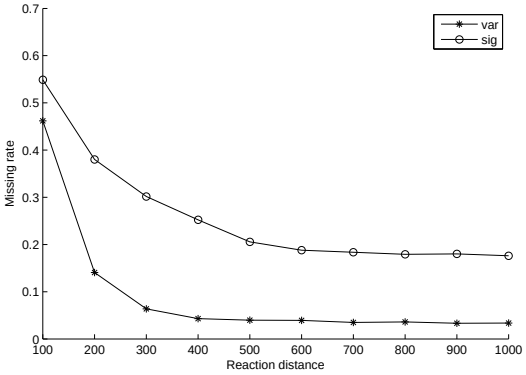
Figure 3(a) shows rate of exit misses for the two agents SIG and VAR for regular traffic on Highway 408. We find that for both agent types the miss rate decreases with the distance, but in general the VAR agent has a lower miss rate.

Figure 3(b) shows the same measurements for rush hour traffic (with the inflow and outflow increased five times). The conclusions from the normal traffic situation extend to this scenario as well. The rate of exit misses of the VAR agent did not change significantly, on the other hand the miss rate of the SIG agents is much higher, and it cannot be reduced below about 20% even with early preparation.

We conclude that the technique of adapting the speed to the target lane is a major component of safe driving under high traffic conditions. While this might appear as a commonsense advice for an experienced driver, it is an observation



(a) Normal



(b) Rush hour

Figure 3: The rate of exit misses function of the preparation distance with normal inflow and outflow rate 3(a), and during rush hour 3(b).

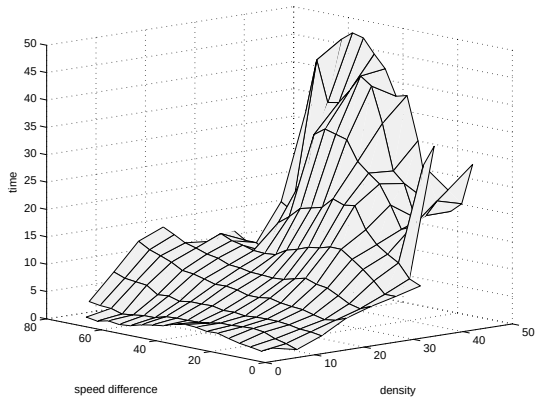
which does not appear in purely virtual physics based models, yet it emerges naturally when that model is augmented with an agent-based conscious behavior simulator.

5.3 Average lane change time

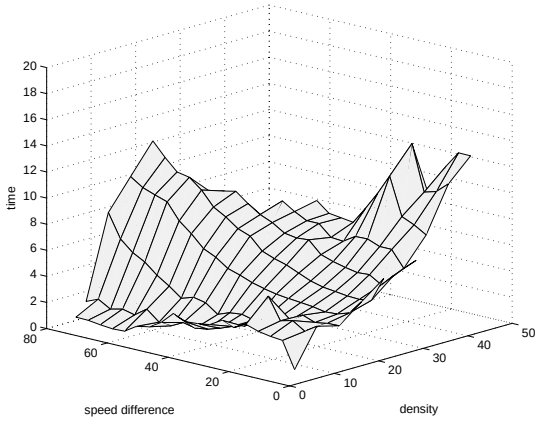
In this series of experiments we studied how long it takes for a SIG or VAR agent to perform a single lane change under various traffic situations. We assumed a very long preparation distance (1000m) and for each lane change forced by the strategic agent behavior we logged the traffic situation and the time to succeed t_s. Thus, the log does not contain the opportunistic lane changes dictated by the virtual physics model. To gather all possible local traffic situations, we run a set of simulations with different flow ratios.

In Figure 4(a) (SIG) and Figure 4(b) (VAR), we divided the density and speed difference into small ranges and plotted the average time to succeed function of density and speed difference.

The first conclusion we can reach from these graphs is that



(a) SIG



(b) VAR

Figure 4: Average lane change time for the SIG and VAR agent in various traffic situations.

both the speed difference and the density affect the time to change lanes. As expected, the time for the VAR agent is consistently shorter than for the SIG agent, reconfirming the validity of the speed adaptation strategy. For example, when the density is 30 vehicles per km, and the speed difference is 20 km/h, it takes 17.49s to do a lane change. However, if the agent adapts the desired speed, it only takes 6.94s to change a lane.

Another insight is that if the vehicle density is low, the speed difference has little effect on the lane change time, because the agent can simply let the high speed vehicle pass and change into the next lane before the new one comes. In the high density lane, however, as the speed difference increases, it needs to wait a long time before the safety condition is satisfied. On the other hand, with the same speed difference, the more vehicles in the agent's next lane, the more time it needs to take for a single lane change.

We conjecture that an experienced driver has an intuitive

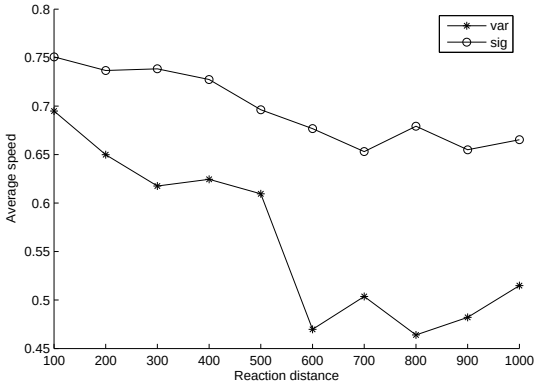


Figure 5: The average speed compared to the desired speed for arrived vehicles during rush hour on Highway 408.

understanding of the values of these graphs (choosing the graph which corresponds to his own driving style, SIG or VAR). What this means that given a specific traffic condition, the driver can estimate the time it will take to change lanes. This estimation will serve as input for the next experiment.

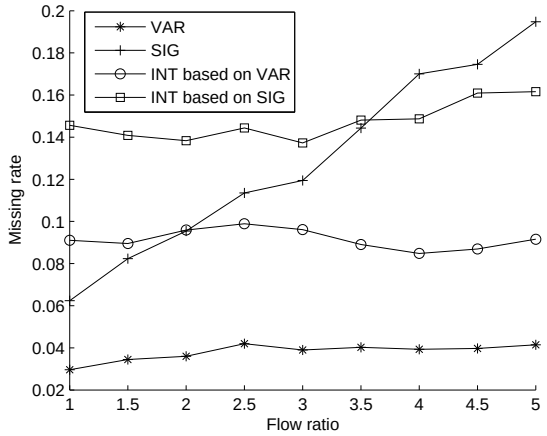
5.4 Adaptive preparation distance

It appears that the safest choice is to choose a VAR type agent as a sufficiently long preparation distance such that the risk of missing the exit is minimized. Unfortunately, such an agent would loose performance. Figure 5 shows the average speed for all arrived vehicles compared to their desired speed. The average speed of the VAR agent is significantly lower, which translates to longer trip times. Some of this is the unavoidable cost of safety. However, by maintaining the same preparation distance both under easy and difficult conditions, the agent is unnecessarily loosing performance.

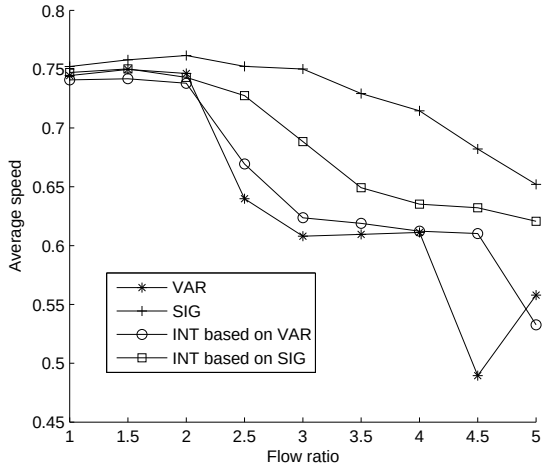
Figure 6 plots the missing rate as well as the averaged speed in the function of the inflow ratio. We compare four strategies: SIG and VAR with a fixed preparation distance of 600m, and their "intelligent" variants with adaptive preparation distance INT-SIG and INT-VAR. We find that, as expected, the adaptive strategies have a more "flat" diagram, allowing us to choose our preferred compromise between performance and risk.

6. CONCLUSIONS

Agent-based modeling can contribute significantly to the accuracy of microscopic highway simulations, in modeling the conscious behavior of the driver and the action of the automated driver aids. Yet, agent researchers need to thread carefully: we are contributing to a field with 50 years of history, with a collection of finely tuned models, which perform very well as long as their operational assumptions are maintained. There is little to be gained from insisting on a "pure" agent-based model. First, there are low level aspects of the driving which do not conform to the definition of an agent. Second, even if we manage to force our model into an agent straitjacket, we will need to reinvent the significant amount



(a) Rate of exit misses



(b) Average speed

Figure 6: The rate of exit misses and average speed in the function of flow ratio on Highway 408

of fine tuning which went into the virtual physics models. On the other hand, if we successfully integrate the virtual physics and agent models, the benefits are immediate.

This paper described an approach where the model of the conscious driver – representing the strategic thinking about lane preferences and planning for a safe exit – is integrated with and acts through the virtual physics model. We found that the model makes successful predictions on issues which are out of reach of the virtual physics based models. For instance, our model correctly predicts that the highest safety risk for exits appears at the case of moderate congestion, both low traffic cases and high congestion is comparatively more safe. This matches well with the results of studies of predicting crash prone situations for rear-end crashes [8] and lane-change crashes [7, 4, 1].

7. REFERENCES

[1] M. Abdel-Aty and V. Gayah. Real-time crash risk reduction on freeways using coordinated and uncoordinated ramp metering approaches. *ASCE Journal of Transportation Engineering*, 136(5):410–423, 2010.

[2] A. Kesting, M. Treiber, and D. Helbing. General lane-changing model MOBIL for car-following models. *Transportation Research Record: Journal of the Transportation Research Board*, 1999(-1):86–94, 2007.

[3] K. Koskinen, I. Kosonen, T. Luttinen, A. Schirokoff, and J. Luoma. Development of a nanoscopic traffic simulation tool. *Advances in Transportation Studies*, 17, 2009.

[4] C. Lee, M. Abdel-Aty, and L. Hsia. Potential real-time indicators of sideswipe crashes on freeways. *Journal of the Transportation Research Board*, 1953(1):41–49, 2006.

[5] Y. Luo and L. Bölöni. Towards a more accurate agent-based multi-lane highway simulation. In *Proc. of Workshop on Agents in Traffic and Transportation (ATT10)*, pages 13–20, May 2010.

[6] D. Ni. 2DSIM: a prototype of nanoscopic traffic simulation. In *Proc. of Intelligent Vehicles Symposium*, pages 47–52, 2003.

[7] A. Pande and M. Abdel-Aty. Assessment of freeway traffic parameters leading to lane-change related collisions. *Elsevier Journal of Accident Analysis and Prevention*, 38:936–948, 2006.

[8] A. Pande and M. Abdel-Aty. A computing approach using probabilistic neural networks for instantaneous appraisal of rear-end crash risk. *Computer-Aided Civil and Infrastructure Engineering*, 23(7):549–559, 2008.

[9] S. Tisue and U. Wilensky. Netlogo: Design and implementation of a multi-agent modeling environment. In *Proceedings of Agent 2004*, 2004.

[10] M. Treiber, A. Hennecke, and D. Helbing. Congested traffic states in empirical observations and microscopic simulations. *Physical Review E*, 62(2):1805–1824, 2000.

[11] M. Treiber, A. Kesting, and D. Helbing. Delays, inaccuracies and anticipation in microscopic traffic models. *Physica A: Statistical Mechanics and its Applications*, 360(1):71–88, 2006.

Learning Driver's Behavior to Improve Adaptive Cruise Control

Avi Rosenfeld[1], Zevi Bareket[2], Claudia V. Goldman[3],
Sarit Kraus[4], David J. LeBlanc[2] and Omer Tsimoni[3]
[1]Department of Industrial Engineering
Jerusalem College of Technology, Jerusalem, Israel 91160
[2]University of Michigan, Transportation Research Institute, USA
[3]General Motors Advanced Technical Center, Israel
[4]Department of Computer Science, Bar-Ilan University, Israel
rosenfa@jct.ac.il, bareket@umich.edu, claudia.goldman@gm.com,
sarit@cs.biu.ac.il, leblanc@umich.edu, omer.tsimhoni@gm.com

ABSTRACT

Adaptive Cruise Control (ACC) is a technology that allows a vehicle to automatically adjust its speed to maintain a preset distance from the vehicle in front of it based on the driver's preferences. Individual drivers have different driving styles and preferences. Current systems do not distinguish among the users. We introduce a method to combine machine learning algorithms with demographic information and expert advice into existing automated assistive systems. This method can reduce the number of interactions between drivers and automated systems by adjusting parameters relevant to the operation of these systems based on their specific drivers and context of drive. We also learn when users tend to engage and disengage the automated system. This method sheds light on the kinds of dynamics that users develop while interacting with automation and can teach us how to improve these systems for the benefit of their users. While accepted packages such as Weka were successful in learning drivers' behavior, we found that improved learning models could be developed by adding information on drivers' demographics and a previously developed model about different driver types. We present the general methodology of our learning procedure and suggest applications of our approach to other domains as well.

1. INTRODUCTION

Cruise control is a known technology that aids drivers by reducing the burden of controlling the car manually. This technology controls the vehicle speed once the user sets a desired speed. Cruise control is not only convenient, but it has the potential to improve the flow of traffic [15], and can be effective in reducing driver fatigue and fuel consumption [1]. In this paper, we focus on a second generation of cruise controls– adaptive cruise control (ACC). ACC is designed as a comfort-enhancing system, which is an extension of conventional cruise control (CC). The ACC system relieves the driver from some of the longitudinal-control tasks by actually controlling speed and headway keeping, but the driver can choose to engage or disengage the ACC at any time. The major difference between ACC and CC is the use of radar technology to maintain a preset distance between the vehicle with the ACC and other vehicles on the road. This distance is controlled by a "gap" parameter which sets the minimum gap (headway distance) to the vehicle in front of it. Figure 1 shows a picture of a steering wheel with the ACC technology. Note the existence of a "gap" switch on the left side of the figure.

While ACC adds more automation to the driving experience, it typically also requires the driver to set and adjust one more parameter, the gap setting. The current approach is to preset the gap setting to a default value which can be adjusted by the driver manually based on his driving preferences. Another approach taken in previous published attempts was to learn this setting focusing on mechanisms such as fuzzy logic [8, 9]. In these previous approaches, rules were learned manually after having interviewed human drivers. Based on these rules the gap setting value was adjusted automatically to the conditions of the drive without considering the particular driver in the vehicle. Individual drivers, however, differ in their driving styles and preferences. Therefore, a personalized learning approach may be valuable.

In this paper, we primarily focus on a method that learns how to quickly and accurately adjust the gap setting based on the specific driver and context of a drive. To accomplish this task, we created general driver profiles based on an extensive database of driving information that had been collected from 96 drivers [5]. We used post-processing of data from that study. Our general method is that once a new driver is identified we classify this driver as being similar to previously known drivers and set the initial gap setting accordingly.

The challenge of this study was to process real world data so as to obtain the most accurate and practical rules from the learning algorithms. We found that the information gleaned from demographics and the driver's type was crucial for creating more accurate learning models. This work focuses on which attributes will help, and a general methodology for adding them. By following this methodology, we found that a better application could be created in this domain, and are confident that better applications can be created in other domains as well.

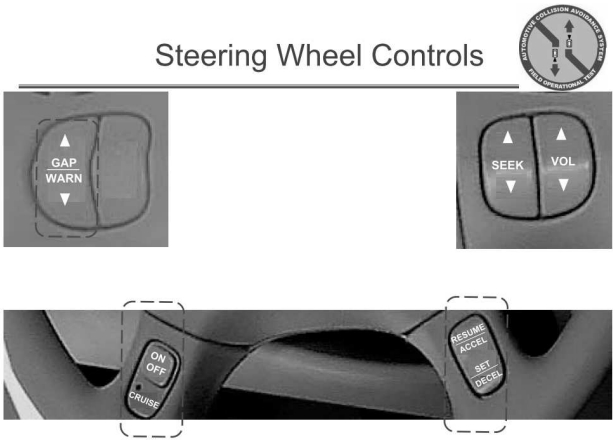


Figure 1: A steering wheel fitted with ACC technology.

This paper is organized as follows. In Section 2 we introduce related work and specify the driver type information that aided in modeling driving behavior. Then, Section 3 describes the theoretical and implementation challenges the ACC learning agent must overcome. Section 4 details the extensive database used to create the ACC agent. Section 5 provides empirical support for the success of effectiveness of this agent. Finally, Section 6 concludes and presents possible directions for future research.

2. RELATED WORK

The concept of using a group of characteristics to learn people's behavior has long been accepted by the user modeling community. Many recommender systems have been built on the premise that a group of similar characteristics, or a stereotype, exists about a certain set of users [12]. Even more similar to our work, Paliouras et. al [10] suggested creating questionnaires, distributing them, and then creating decision trees to automatically define different groups of users. Similarly, our application assumes that some connection exists between users, which can be learned using machine learning techniques. We propose that this approach be applied to customize settings within an application, here ACC, and not within recommender systems.

Previously, Fancher et. al [7], analyzed a group of 36 drivers and their acceptance of adaptive cruise control (ACC). While all drivers enjoyed and accepted the ACC, they found that drivers could be divided into three types with each group demonstrating specific driving tendencies which impact their headway and closing speeds, relative to vehicles ahead. In very general terms, these groups were assumed to be: one that is most aggressive, another that is least aggressive, and a third that is in between. Although it is clear that more detailed grouping may exist, and that a different profiling of the drivers' population can be made, for the purpose of this study the characterization analysis was aimed at identifying the above three grouping types. The three driving styles are: 1. Hunters (aggressive drivers who drive faster than most other traffic and use short headways); 2. Gliders (the least aggressive drivers who drive slower than most traffic or commonly have long headways); and 3. Followers (whose headways are near the median headway and usually match the speed of surrounding traffic). In this scheme of things, Hunters are drivers who tend to drive faster than the surrounding flow and they tend to travel at shorter headway times than those adopted by other drivers. In contrast, at the other end of driver characteristics, Gliders tend to travel slower than the surrounding flow and they tend to travel at longer headway times than those adopted by other drivers. Between the Hunters and Gliders lie the Followers who tend to go with the flow of traffic. They tend to adapt their driving behavior to the situation they are in.

The idea of assisting the driver in the task of longitudinal control has been the focus of research in the last decade [8, 9]. Operation tests have given insight into this task. However, the goal of this project was to attempt to create an intelligent ACC agent that could potentially set this longitudinal value autonomously through adjusting its gap setting per each driver.

In this paper, we use driver characterization into types (hunter, glider or follower) in addition to other demographic information to attempt to build an application that predicts how the ACC should set its gap (headway) given this information and road situation. In general, other research has previously found that we can better predict people's behavior by combining relevant behavior theory, here about people's driving type and demographics, in conjunction with machine learning methods. These studies have included how other behavior theories: Aspiration Adaptation [14] and the Focal Points [18] could be used in conjunction with machine learning algorithms to create an improved classifier. These results also showed some positive correlation between the complexity of the problem domain and the improvement in performance when augmenting the behavior model. Thus, the more complex the learning task, the added gain in the learning model by adding behavior information. This paper explores how the behavior model of a driver's type impacts their gap setting.

3. LEARNING METHOD

Current ACC systems allow the user to choose a value for the gap setting between six possible values (1–6). These values control the distance the ACC autonomously maintains with the vehicle in front of it. Currently, one value is set as default (in our case this value was 6) and the user may change it during his drive as he wishes. In order to study the problem of predicting what gap setting a person would select, we constructed two different types of models. The first type of model was a regression model which attempted to predict the number a given driver would select given the current driving conditions. The second type of model was a decision tree model which treats each number within the system as discrete values representing different categories a driver can choose. Our goal was to use the output of either model to automatically set the gap setting. Towards this goal, the second model is seemingly the better choice as its output directly correlates to a value within the system. In contrast, the regression model outputs a decimal value (e.g. 3.5) that must be first rounded to the closest value within the system to be used. However, the advantage of this model is that a mistake between two close values (e.g. 3.5 being close to 3 and 4) is not as mathematically significant as mistakes between two extreme values (e.g. between 2 to 6). In contrast, the discrete decision tree model weighs all types of errors equally. In practice, the regression model will likely be more useful if the user is willing to accept errors between two similar values.

Additionally, we focus on two secondary goals, when the ACC is first engaged, and when the ACC is disengaged. Here, the goal was not to create an agent to autonomously engage or disengage the ACC. However, by analyzing when people are most comfortable

with the ACC, we hope to understand the user acceptance of such systems.

In both of these learning tasks, we are confronted by the known dataset imbalance problem [2]. In many real-world problems, as is the case here, each class is not equally represented. In fact, in the specific case of the ACC engagement task, over 90% of manual driving cases continue their manual driving, and in only a small percentage of cases do people engage the ACC. From a statistical perspective, a classifier could then naively classify all cases as being in the majority case and still have extremely high accuracy. However, because only the "minority" cases are relevant, novel methods are needed to find them. While several algorithms exist, we specifically focused on the MetaCost algorithm [4] because of its flexibility in controlling the bias size given to the minority case.

A second key implementation challenge lie in the algorithms themselves. While we used the popular Weka learning package [16] to implement all learning algorithms, the content experts often believed that the resulting models were extremely overfitted. Unfortunately, many theoretical learning algorithms are prone for overfitting when applied to real-world datasets [17] and the content experts involved with the project found this to be the case in this domain as well. To overcome this challenge we used simplified decision trees. The idea of using simplified decision trees is not new, and a variety of algorithms have been developed for simplifying decision trees [6]. However, these algorithms were developed for **increased** performance. In this application, we intentionally sacrificed a certain level of performance to reduce overfitting. Thus, we applied these algorithms for a different reason than the one they were developed for, but still achieved the desired result – a non-overfitted decision tree.

Specifically, we used the reduced error pruning method developed by Quinlan [11], named REPTree with in the Weka learning package [16]. According to this approach, a decision tree of maximum height T_{max} is reduced to T_{Depth}, with $T_{Depth} \leq T_{max}$, ostensibly to produce improved performance. In our application we chose the maximum value for T_{Depth} that the content experts deemed was not overfitted, as we found that the best performance was achieved when $T_{Depth} = T_{max}$ and our goal was to achieve the best performance within the model without producing an overfitted model. Empirical results detailing specifics of the models used to create the ACC's agent are explained in the next section.

4. EXPERIMENTAL SETUP

Data for our analysis were taken from the Automotive Collision Avoidance System Field Operational Test (ACAS FOT) [5]. In that study, to understand how different drivers use an ACC, each of 96 drivers was presented with a vehicle fitted with the ACC which they used for a period of 4 weeks. During the first week the ACC system was not available. That is, if the driver engaged the cruise control, it simply maintained speed just like the conventional system (CC). During the next three weeks, if the driver chose to engage the cruise control, it functioned as ACC. In general, three different datasets were considered. The first, and most basic, dataset were objective characteristics that can be studied based on the location of the vehicle itself, e.g., headway distance to the lead vehicle, vehicle speed, longitudinal acceleration, road type (country, city, or highway), weather (including day or night) and road density (is there traffic). A second dataset added driver characteristics. These properties focus on driver demographics such as age, sex, income level

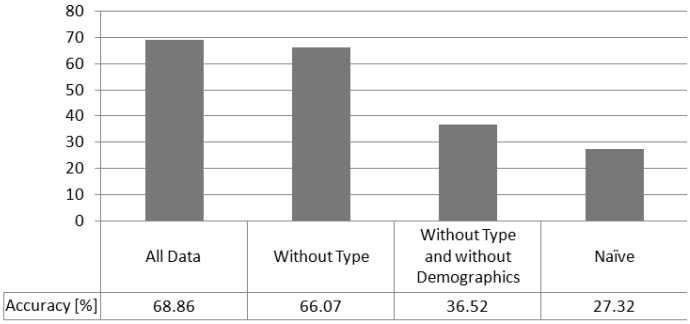


Figure 2: The importance of driver type and demographics in predicting the gap setting within the ACC for a discrete decision tree model.

(high, medium, low), and education level (High School, Undergraduate, and Graduate). The ACAS FOT data consists of a good mixture of these demographics with a 51% male to 49% female split, 31% young (aged 20–30), 31% middle aged (aged 40–50), and 38% older drivers (aged 60–70), and people from a variety of education and socioeconomic levels. The last dataset also logged a previously developed measure used to quantify a driver's behavior [7].

The experimental design of the ACAS FOT was a mixed-factors design in which the between-subjects variables were driver age and gender, and the within-subject variable was the experimental treatment (i.e. ACAS-disabled and ACAS-enabled). The disabled period was treated as a baseline measure, since the research vehicle operated like a conventional passenger vehicle. The drivers operated the vehicles in an unsupervised manner, simply pursuing their normal trip-taking behavior using the ACAS test vehicle as a substitute for their personal vehicle. Use of the test vehicles by anyone other than the selected individuals was prohibited. The primary emphasis on user selection for the field operation test was to roughly mirror the population of registered drivers, with simple stratification for age and gender. No attempt was made to control for vehicle ownership or household income levels. Thus, although the ACAS FOT participants may not be fully representative of drivers who might purchase such a system, they were selected randomly and represent a wide range of demographic factors.

5. RESULTS

In this section we present results for the three previously defined problems: predicting a driver's gap setting within the ACC using both discrete and regression models, predicting when a driver will engage the ACC, and predicting when a driver will disengage the ACC. In all three problems we present how the driver type and other demographic information helped improve the model's accuracy. Additionally, we analyze which attributes were most significant in this application, how we avoiding overfitting, and how we addressed the dataset imbalance problem within this application.

5.1 Setting the ACC's Gap Setting

Figure 2 presents the accuracy of the decision tree model to learn a driver's preferred gap setting in the discrete model. Clearly, adding

the demographic data here is crucial, as the model's accuracy drops from over 66% accuracy with this data to less than 37% accuracy without this. As a baseline, we also include the naive classifier, which is based on the most common gap setting– here the value of 6, which is also the system's default. Note that the naive model had an accuracy of nearly 27%, far less than other models. The user's type did improve accuracy, as adding this information to the type increased accuracy to near 70%. In line with our previous work [13], we hypothesized that adding this behavior model yields less significant increases if it can be learned from other attributes within the data. Here, we believed that adding information about drivers' type is less important, as their type was already evident from information such as the driver's demographics.

To support this hypothesis, we constructed a decision tree (again C4.5) to learn the driver's type. We found that this value could be learned with over 95% accuracy (95.22%) when learned with the full Reptree (T_{max})– which strongly supports our hypothesis. We present a pruned version of this tree (T_{Depth} = 4) within Figure 3. From an application perspective, we were not shocked to find that a driver's age factored heavy in their driving behavior. This characteristic is factored in actuary's insurance tables, and is a known factor in car insurance premiums [3]. Note that this characteristic was the first level below the root of the tree, demonstrating this quality. However, possibly equally interestingly is that we found education, not gender to be the next most important factor as it formed the second level within the decision tree. This factor is typically not considered by insurance companies [3], but may be worth considering. Only in the third level did we find the popular characteristic of gender to factor in, but income also weighed in as an equally important important factor. Overall, we found that young men or women with only a high school degree tended to "hunters" or those with extremely aggressive driving habits, college educated women, and people with higher degrees but lower paying jobs tended to be the less aggressive "gliders". Middle aged men with high school degrees, all middle aged people with college degrees, and people with higher degrees buy lower paying jobs also typically belonged to the middle "gliders" category. But older women with college degrees, people with low or medium paying jobs with only high school degrees, and all older people with higher degrees tend to be of the least aggressive "follower" type. Naturally, exceptions existed, and this simplified tree only is approximately 75% accurate. Nonetheless a general direction is evident from this tree, and was one that the content experts felt was not overfitted.

T_{Depth}	Accuracy [%]
2	47.55%
3	56.41%
4	62.43%
5	65.46%
6	67.51%
7	68.50%

Table 1: Analyzing the tradeoff between the model's accuracy and the height of the tree T_{Depth}.

Similarly, it was important to find a decision tree that models drivers' gap settings that is not overfitted as well. Note that the accuracy of the Figure 2 given all data is nearly 70%. However, while this value is based on the mathematically sound C4.5 algorithm [11], the content experts again felt this decision model was overfitted. We then proceeded to reduce the size of the tree as to generalize the model, thus preventing this phenomenon. However, as Table 5.1, demonstrates reducing the table size does not improve the model's accuracy, as previous theoretical works suggest [6], but did produce trees that were acceptable to the content experts. Note that raising T_{Depth} yields marginal increases in model accuracy with T_{Depth} = 7 being nearly accuracy to the result in Figure 2. In general, we found that the experts were happy with much smaller trees, but those with similar accuracy. For this problem, we display in Figure 4 the resultant tree of T_{Depth} = 4 which is only 6% less accurate the full tree in Figure 2. However, for comparison, the full tree produced with the unpruned C4.5 algorithm has a total size of 1313 leaves and branches, while the pruned tree only has a total size of 50 leaves and branches. Thus, from an application perspective, this tree was strongly favored by the experts, even at the expense of a slightly less accurate model. Note that the rules themselves are still heavily influenced by the driver type and demographic information, with driver type being the first level of the tree and the second and third levels of the tree again being primarily based on demographics such as age, gender, education, and income level.

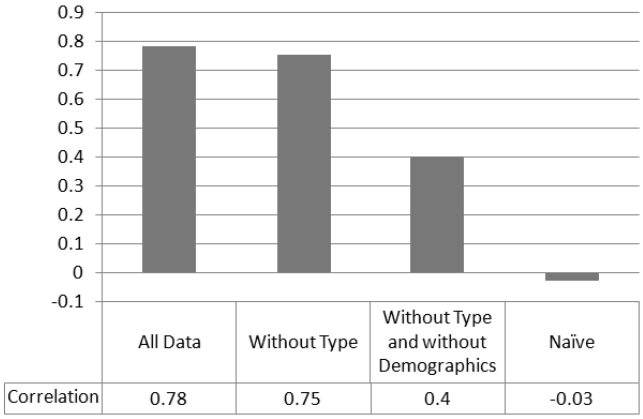


Figure 5: The importance of driver type and demographics in predicting the gap setting within the ACC for a regression model.

Similarly, we were able to create an accurate regression model, the results of which are found in Figure 4. Within these models, correlation values can range from 1.0 (fully positive correlated) to -1.0 (fully negatively correlated) with 0 be with no correlation. We found a model with both demographic and type data yielded a correlation of 0.78, while without this information the accuracy dropped to 0.75. Using only vehicle specific data yielded a model of only 0.4, and the naive model (here using the average gap value of about 3.5) yielded a value of nearly 0. Again, we found that the type only slightly improved the model's accuracy, as much of this information was already subsumed within the drivers' demographics. Here again, the experts opted for a reduced model, despite the sacrifice of slightly less accuracy.

5.2 Predicting when the Driver will Engage and Disengage the ACC

While the focus of the ACC is on the gap setting that differentiates the adaptive cruise control, from the "standard" cruise control, we also considered two additional problems: when people activate the ACC and when they deactivate it. The goal behind the gap value task was to allow an autonomous agent to set, at least initially, this value within the ACC. However, by understanding when people are more likely to use this product we can hopefully increase its

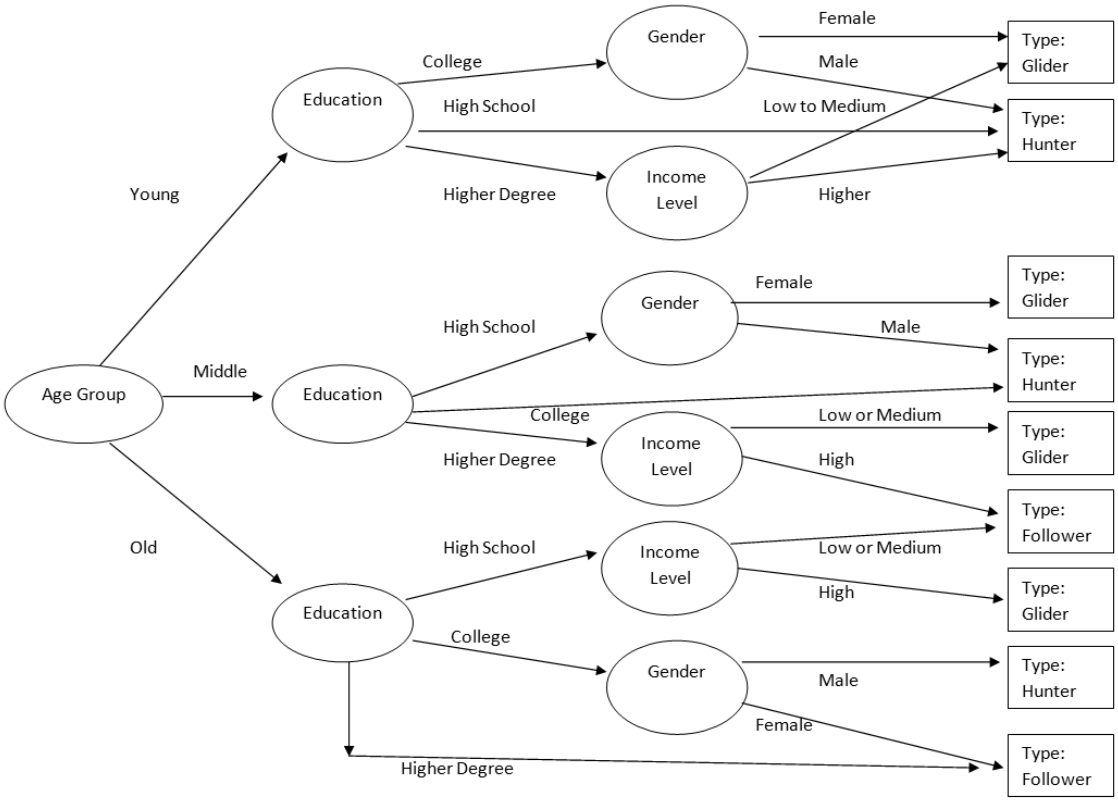


Figure 3: The decision tree for learning a driver's "type".

acceptability and use. Similarly, by understanding when people disengage the ACC we can hopefully create new generations of this technology where people will use it longer and not feel compelled to disengage it.

In both of these learning tasks, we are confronted by the known dataset imbalance problem [2]. In this paper, we constructed two models for these two problems based on the same three types of datasets. The first model is a basic C4.5 without any modification. As was the case in gap setting task, we considered attributes based on the behavior type model, driver demographics and the vehicle's characteristics. In the second model, we again used the same three datasets, but created a learning bias to find the important minority cases. We specifically focused on the MetaCost algorithm [4].

Table 2 displays the complete results demonstrating the tradeoff between a model's accuracy and the success in finding the minority cases in the task of predicting when a driver engages or disengages the ACC. The first four rows represent different models created to predict when a person would activate the ACC. The first model is the standard decision tree algorithm C4.5. In addition, we considered three weight biases within the MetaCost algorithm: 0.5, 0.7 and 0.9. Note that raising these weights allows us to give greater weight to the minority case, thus increasing the recall of cases found, but at a cost to the overall accuracy of the model. For each of these models we trained four different models: one created with all information, one without the type information but with the demographic information, one without the type and without the demographic information, and a naive model that assumes the majority case– that a person continues driving in manual mode. The accuracy of each of these models are found within the first four columns in Table 5.2, and the corresponding recall levels for these models are found in the last four columns of the table. Similarly, we also considered the task of predicting when a person turns off the ACC, and trained models based on the same four algorithms with the same four datasets. The results for the accuracy and the recall of these models are found in the last four rows of Table 2.

Ideally, one would wish for a perfect model: e.g. one with 100% accuracy and recall of all cases. Unfortunately, this is unrealistic, especially in tasks involving people which are prone to variations due to noise. Nonetheless, the overall conclusion is that by adding more information, and specifically about a person's demographics, we were able to achieve higher overall accuracies with better recall.

We would like to present the driver for a recommendation as to when to engage the ACC. Towards this goal, we wish to build a model that is as accurate as possible, but with a minimum threshold. Thus, we wished to set the desired confidence level of the model, as found based on the recall of the minority class, before presenting a recommendation to the user. Figure 6 displays the interplay between the overall model's accuracy and the recall within the minority cases in the task of predicting when a driver engages the ACC. Again, the most desirable result is one in the upper right corner– high accuracy **and** recall. However, as one would expect, and as evident from Table 2, the naive case of continuing without engaging the ACC constitutes over 91% of the cases, but this

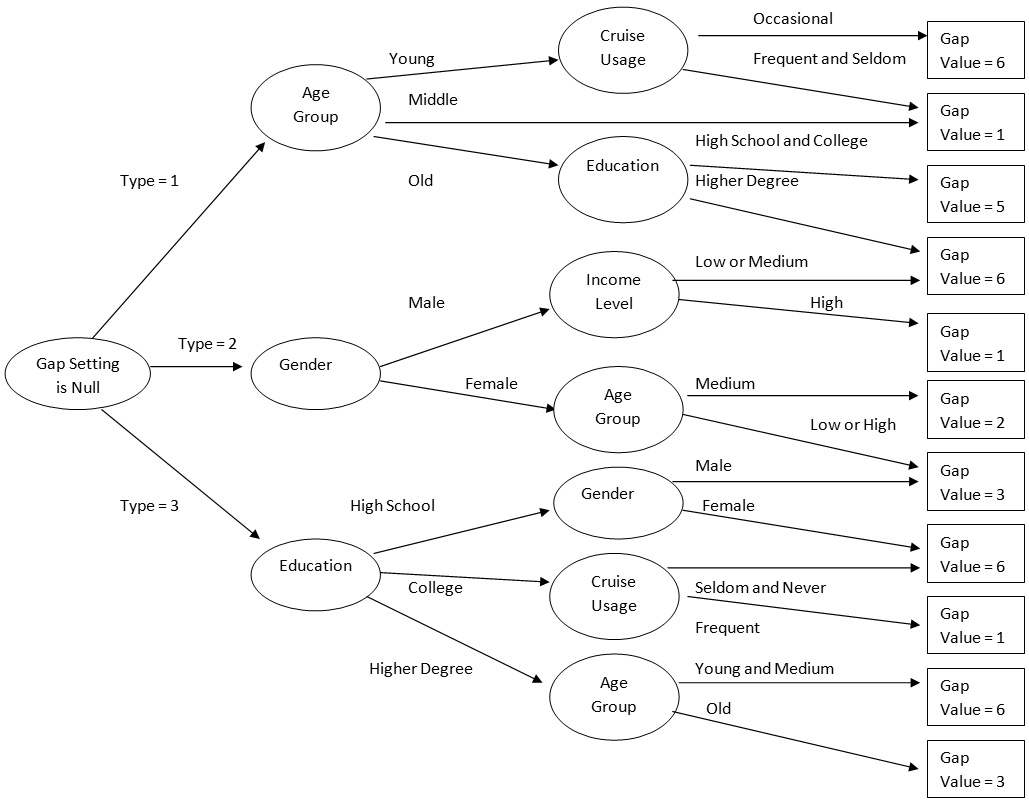


Figure 4: The decision tree for learning the ACC's Gap Setting for T_{Depth} =4.

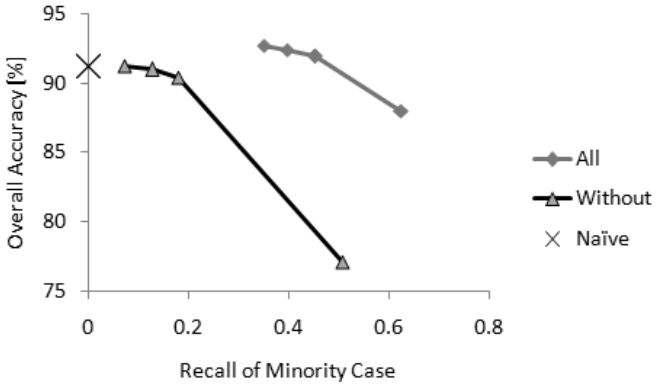


Figure 6: Comparing the overall model accuracy and recall for cases for engaging the ACC

model will have recall of 0 for the minority case. By modifying the weights within the MetaCost algorithm we are able to get progressively higher recall rates over the basic decision tree algorithm. Also note that the model trained with all information achieves significantly better results than one without the type and demographic information.

Similarly, Figure 7 displays the same interplay between the overall model's accuracy and the success in finding the minority cases in the task of predicting when a driver disengages the ACC. In this task, the naive case assumes that the driver will continue with the ACC constitutes over 86% of the cases, but this model will have recall of 0 for the minority case (see the left side of Figure 7). Note that we were again able to raise the recall within the minority case by creating weight biases of (0.5, 0.7 and 0.9), but again at the expense of a lower overall accuracy. However, as opposed to the engage ACC task, we noticed that the gain from the demographic and type information was not very significant. In fact, upon inspection of the output trees, we noticed to our surprise that people's decision to disengage the ACC was more dependent on how quickly the ACC slowed the vehicle down, and not on the overall behavior of the driver. Thus, it should be noticed that simply adding attributes is not a panacea for higher accuracy– it only improves accuracy when relevant to the learning task at hand.

Overall, these results suggest that finding attributes beyond the observed data can be critical for accurately modeling a person's be-

ACC On	All Info	Without Type	Without Demo	Naive	All Info	Without Type	Without Demo	Naive
C4.5	92.67	92.32	91.22	91.27	0.35	0.32	0.07	0
MetaCost 0.5	92.42	91.97	90.97	91.27	0.40	0.36	0.13	0
MetaCost 0.7	91.93	91.38	90.37	91.27	0.45	0.42	0.18	0
MetCost 0.9	87.99	86.60	77.12	91.27	0.63	0.61	0.51	0
ACC Off	All Info	Without Type	Without Demo	Naive	All Info	Without Type	Without Demo	Naive
C4.5	88.71	88.64	88.42	86.37	0.37	0.37	0.35	0
MetaCost 0.5	88.59	88.55	88.14	86.37	0.43	0.42	0.41	0
MetaCost 0.7	87.68	87.49	87.31	86.37	0.49	0.49	0.49	0
MetCost 0.9	82.03	82.23	81.15	86.37	0.66	0.67	0.66	0

Table 2: Analyzing the tradeoff between overall model accuracy (left side of table) and recall of the minority cases (right side) in both the task of when people turn the ACC on (top) and off (bottom).

Figure 7: Comparing the overall model accuracy and recall for cases for disengaging the ACC

havior. Similar to previous studies that found that other behaviorial theories can better predict people's actions [14, 18], this work found that a driver's preferred gap setting could be better predicted by using a model of driving behavior [7]. Even if this measure was not readily available, an accurate estimate of this value could be learned based on a driver's demographic data.

Generally, one of the goals of this paper is to encourage people who build applications to consider incorporating data from external measures, such as psychological or behaviorial models. As was true in other domains as well [14, 18], exclusively using behavior models alone, such as the driver type possible in this domain [7], is not sufficient. By combining the driver type with other data, we achieved a prediction accuracy of nearly 70% within the discrete decision tree model (Figure 2) and a correlation of 0.78 within the regression model (Figure 4). However, when we used only the driver type information and removed the demographic information these models dropped to an accuracy of 46% and 0.55 respectively. This suggests that exclusively using behavior models is not as effective as the approach we present. Thus, we advocate for synthesizing data gleaned from behaviorial models in conjunction with observed domain data, something we believe can be effective in many other domains as well.

Practically, we are studying how either or both of these attributes can be used in the company's ACC. The advantage to using the demographic data alone is that ostensibly it can be provided before the driver begins using the car (e.g. in the showroom) and thus can be used to accurately model the driver from the onset. However, people may be reluctant to provide this information due to privacy concerns. Using driver profiling information is relatively difficult to calculate and is based on observed behavior over a period of time [7]. Thus, this value cannot be used to initially set values within the ACC. However, this data can be collected without privacy concerns and can be used to further improve the system's accuracy over time.

6. CONCLUSIONS

This paper presents an in-depth analysis into how learning approaches could be applied to create an intelligent Adaptive Cruise Control (ACC) agent that learns a driver's profile – both in terms of what gap setting she will chose, and when she is likely to engage or disengage their ACC. To create this agent we used real-world data from the past experience of many drivers from the ACAS field test data [5]. Specifically, we created driver models based on two learning approaches: regression and decision trees. Both were able to learn accurately the gap setting of an individual given his demographics characterization and driving type (hunter, glider or follower) with nearly 70% for the decision tree model and with a correlation of 0.78 for the regression model. These experiments emphasized the need for driver information including a behavior model about the driver's type [7] in addition to the information collected on the trips themselves and their demographic information. These results stress the fact that drivers may be very different from each other and previous approaches that set the gap setting similarly for all drivers [8, 9] are less effective. Therefore, driver characterization is essential for adapting automated systems in the vehicle. These differences among humans are made more salient when trying to learn when users engage or disengage from an automated system such as the ACC. Reactions could be very different teaching us also about the tendencies of users towards automation.

We present solutions for two practical challenges in applying learning algorithms to this challenging domain: preventing overfitted models, and creating effective models in cases where a strong majority category existed but the important events were in the minority category. We address the overfitting issue by created simplified decision trees, and we use the MetaCost algorithm [4] to learn from unbalanced data sets. We present extensive results details specifics of this application and how these algorithms were used within this challenging transportation domain.

More generally, adapting automated processes to better serve humans is a challenging task because humans are characterized by inconsistent behaviors, have difficulties in defining their own preferences, are affected by their emotions, and are affected by the complexity of the problems they face within the context of these

problems. By understanding the current state of acceptance of automated systems and learning about differences among human users, we can improve the next generations of adaptive automated systems adjusted to their particular human users.

7. REFERENCES

[1] R Bishop. Intelligent vehicle applications worldwide. *IEEE Intelligent Systems*, 15(1):78–81, 2000.

[2] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: Synthetic minority over-sampling technique. *J. Artif. Intell. Res. (JAIR)*, 16:321–357, 2002.

[3] Pierre-Andre Chiappori and Bernard Salanie. Testing for asymmetric information in insurance markets. *Journal of Political Economy*, 108(1):56–78, February 2000.

[4] Pedro Domingos. Metacost: a general method for making classifiers cost-sensitive. In *KDD '99: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 155–164, New York, NY, USA, 1999. ACM Press.

[5] R. Ervin, J. Sayer, D. LeBlanc, S. Bogard, M. Mefford, M. Hagan, Z. Bareket, and C. Winkler. Automotive collision avoidance system (acas) field operational test methodology and results, (us dot hs 809 901). washington, dc: Department of transportation, 2005.

[6] Floriana Esposito, Donato Malerba, and Giovanni Semeraro. A comparative analysis of methods for pruning decision trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:476–491, 1997.

[7] Paul Fancher and Zevi Bareket. A comparison of manual versus automatic control of headway as a function of driver characteristics. *3rd Annual World Congress on Intelligent Transport Systems*, 1996.

[8] J. E. Naranjo, C. Gonzalez, R. Garcia, and T. de Pedro. ACC+Stop&go maneuvers with throttle and brake fuzzy control. *Intelligent Transportation Systems, IEEE Transactions on*, 7(2):213–225, 2006.

[9] J. E. Naranjo, C. Gonzalez, J. Reviejo, R. Garcia, and T. de Pedro. Adaptive fuzzy control for inter-vehicle gap keeping. *Intelligent Transportation Systems, IEEE Transactions on*, 4(3):132–142, 2003.

[10] Georgios Paliouras, Vangelis Karkaletsis, Christos Papatheodorou, and Constantine D. Spyropoulos. Exploiting learning techniques for the acquisition of user stereotypes and communities. In *UM99 User Modeling: Proceedings of the Seventh International Conference*, pages 169–178, 1999.

[11] J. Ross Quinlan. *C4.5: Programs for Machine Learning (Morgan Kaufmann Series in Machine Learning)*. Morgan Kaufmann, 1 edition, January 1993.

[12] Elaine Rich. User modeling via stereotypes. *Cognitive Science*, 3:329–354, 1979.

[13] Avi Rosenfeld and Sarit Kraus. Modeling agents through bounded rationality theories. In *IJCAI-09*, pages 264–271, 2009.

[14] Avi Rosenfeld and Sarit Kraus. Modeling agents based on aspiration adaptation theory. *Autonomous Agents and Multi-Agent Systems*, 24(2):221–254, 2012.

[15] Bart van Arem, Cornelie J. G. van Driel, and Ruben Visser. The impact of cooperative adaptive cruise control on traffic-flow characteristics. *IEEE Transactions on Intelligent Transportation Systems*, 7(4):429–436, 2006.

[16] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann, June 2005.

[17] Zijian Zheng, Ron Kohavi, and Llew Mason. Real world performance of association rule algorithms. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '01, pages 401–406, 2001.

[18] Inon Zuckerman, Sarit Kraus, and Jeffrey S. Rosenschein. Using focal point learning to improve human-machine tacit coordination. *Autonomous Agents and Multi-Agent Systems*, 22(2):289–316, 2011.

Recognizing Demand Patterns from Smart Card Data for Agent-Based Micro-simulation of Public Transport

Paul Bouman, Milan Lovric, Ting Li, Evelien van der Hurk, Leo Kroon, Peter Vervest
Rotterdam School of Management, Erasmus University
Burgemeester Oudlaan 50
Rotterdam, The Netherlands
{PBouman,MLovric,TLi,EHurk,LKroon,PVervest}@rsm.nl

ABSTRACT

In public transportation the question of how to achieve a good match between demand and capacity is essential for operators to provide a high quality service level within reasonable costs. Agent-based micro-simulation is a promising method to evaluate the impact of operational decisions and selected tariffs at both the level of the individual passenger and the aggregate level of the operator. During recent years, this technique has been applied successfully to several large scale real life cases. However, the demand of the agent population in these simulations is usually derived from aggregated census data and surveys conducted among a relatively small sample of the travelers. With the advent of smart card ticketing systems new opportunities to generate an agent population have surfaced. We use a unique smart card dataset containing four months of individual mobility data from passengers among three modalities in an urban Dutch public transportation system to generate agent populations. We model the temporal flexibility of agents based on patterns observed in the check-in/check-out behavior of individual travelers. We then run simulations to study how these agent populations react to a discounted tariff in the off-peak hours. Finally, we discuss opportunities to improve our approach in the future.

Categories and Subject Descriptors

H.4.2 [**Information Systems Applications**]: Miscellaneous

General Terms

Experimentation, Algorithms, Management

Keywords

Agent Based Micro-simulation, MATSim, Pattern Based Demand, Public Transport, Revenue Management, Smart Card Data

1. INTRODUCTION

In public transportation systems without seat reservations, the question of how fluctuating demand can be serviced in a cost-efficient way poses a major challenge. Peaks in demand have a high toll on the costs, since they dictate the required amount of staff and the number of vehicles, while vehicles that are almost empty generate a net loss for the operator. Tools that allow the public transport operator to evaluate the effects of operational and strategic decisions on costs and demand are therefore vital to achieve the goal of improving the service quality and financial performance. However, most of the tools used in practice aggregate the passengers to homogeneous flows, either because detailed data is not available, or to reduce the complexity the decision maker has to face. During recent years, smart card systems have been introduced that log all movements of individual passengers through the systems. This gives a lot of detailed data that was previously unavailable. However, given the body of research related to smart card data, we can see that incorporating such data into the tools used for decision making is a non-trivial task [17].

A promising approach is agent-based micro-simulation. In such a simulation, individual passengers and vehicles are modeled through agents that interact with the public transportation system according to their individual goals. In this paper, we will use the MATSim simulation package [1] which has an active user-base and has been applied to a number of large scale scenarios. Within MATSim, all agents try to adapt their plans in such a way that their utility is improved. The simulation runs until there is no significant improvement within the agent population, i.e. until the population reaches an approximate equilibrium.

The major issue in generating an agent population from real life observations is the question how we can prevent agents to divert from this equilibrium in an unrealistic way, without restricting the agents in such a way that their only preference is to replicate the observed state.

We will limit our field of application to the study of *revenue management*. In revenue management[21] we want to control demand by adapting our pricing strategy in such a way that we get a better match between the available capacity and the demand emerging from the population. Our population can try to adapt to our pricing strategy by shifting the time at which they travel. We will study how the population reacts to an off-peak discount, but we believe that our approach is suitable for many other applications. One idea is to include the choice for mode of transport.

When generating our agent population, we run into the problem that the number of observed journeys differs a lot between individual passengers. We solve this problem by combining three types of demand that we can detect in our smart card dataset: *trip-based*, *tour-based* and *pattern-based* demand. Our first goal is to show how we can efficiently generate the agent population from our smart card data using these three demand models. Our next goal is to discuss how we can experiment with different parameters for the demand models to study revenue management. The final goal is to

discuss our results and how we can improve our methods in the future.

The remainder of this paper is organized as follows: in Section 2 we discuss prior literature and related work. In Section 3 we discuss smart card datasets in general and our dataset in particular. Section 4 addresses the modeling of demand, based on the smart card dataset. In Section 5 we discuss the simulation and our experimental setup. We present the results of our experiments in Section 6. Finally, we discuss our results and opportunities for extensions of our approach in Section 7.

2. RELATED WORK

In recent years, smart card ticketing systems have attracted notable attention from the research community. A recent literature review on the use of smart card data in public transportation is given by [17]. They divide the studies into three categories: strategic-level studies, tactical-level studies and operational level studies. Since some of the public transportation systems only work with check-ins, part of the literature focuses on estimating the destination of passengers given their check-in location and time (for example, [23]). Some literature describes how the behavior of passengers can be analyzed. A notable example is [15], where spatial and temporal variations are measured across different types of cards. However, the literature review [17] contains not a single reference to the use of smart card data within a simulation context. Moreover, their conclusion contains the following quote:

> For the mass of data available on individual trips, new modeling methods will be needed, such as the Totally Disaggregate Approach, because classical models cannot be used at a such detailed level of resolution. [...] It will then be possible to calibrate individual base models from these large datasets. [17]

In the simulation of road traffic, microscopic simulation models have been a topic for quite some years. In the 1990's, it was mostly a topic studied as a field of application for super computers [10]. With the increase of computing power, more applications emerged in the 2000's, including [22]. With the introduction of MATSim [1], we saw a rise in literature related to micro-simulation. MATSim has been applied to some very large scale scenarios, including simulations of Berlin [19] and Zürich [13], both including more than a million individual travelers. Recently, MATSim was expanded from the simulation of road traffic, to the simulation of public transportation as well [18]. The website of the project contains a list with the most important publications related to the project and is updated regularly.

The kind of microscopic demand which is fundamental in the design of MATSim, is called *activity-based* demand [9] and was already discussed in the context of micro-simulation by [14] in 1997. This is an approach where travel demand is modeled by means of the activities the individual travelers want to perform over the day. One way to record the activities of individual travelers is by using surveys (for example [5]). In recent studies, census data was used to perform this synthesis of the activity based demand [4]. A survey on this approach to demand generation is given by [16].

Apart from modeling the activity patterns of travelers, a lot of research regarding the behavior of travelers has been performed, resulting in many sophisticated methods. Most notably, we would like to mention the field of discrete choice modeling [7], since it has spawned a lot of research within the domain of transportation. One of the main tools within discrete choice modeling is the stated-choice survey, where respondents have to select their preferred alternative.

A comprehensive textbook on revenue management is [21]. The focus of studies related to revenue management has been on systems where reservations are made in advance. In our setting, however, we do not have a mechanism where we can decide whether we accept new customers. This is different from, for example, long distance trains and the airline industry where tickets are always bought in advance. An example of a study related to revenue management in a comparable railway setting is [12]. This study shows some of the difficulties in applying revenue management within our context. An example of a succesful application for long distance trains with seat reservations is [8].

3. SMART CARD DATA

During recent years, the Dutch smart card, called "*OV-chipkaart*" was introduced as a cross-operator travel product. Starting from 2009, the smart card was made the mandatory product of travel in major Dutch cities, such as Amsterdam and Rotterdam, replacing paper tickets. One of the unique features of the Dutch system is that passengers have to check-in and check-out with the smart card in all modes of travel, including railways.

We use data collected from smart card usage over the course of four months from a major public transport operator in the Netherlands. During this period, the only avaible tickets were different smart card products. The transactions in our dataset denote either a check-in or check-out in a vehicle or on a platform. Moreover the smart card data contains the mode of travel, the unique id of the chip on the smart card (which we will call the media id), the time stamp of the transaction (in seconds) and the location of the transaction. Due to the sensitivity of the data for the operator and privacy concerns for the passengers, we will only show relative numbers and figures in this paper.

We prepared our raw dataset of almost 60 million transactions in such a way that we could process each transaction sequentially. We had to split up the dataset into separate chunks, using a round robin approach to assign media id's we had not seen before to a fixed chunk for that id. Afterwards, we sorted the separate chunks on media id and time stamp in main memory. We combined the results into a single dataset. While processing this set sequentially, we would be sure to encounter all transactions belonging to a certain media id together, with increasing time stamps.

After sorting the dataset, we linked check-ins and check-outs to make trips. Passengers who forget to check-out gives rise to inconsistencies in the dataset. It is relatively easy to filter these inconsistencies out, by assuming that a consecutive check-in and check-out belong together. This is reasonable, since the system has a maximum amount of time after which a check-in becomes invalid. After this linking step we know all the trips made by the passenger. Since the passengers have to check-in and check-out in each vehicle, we have separate trips when the passenger makes a transfer on his journey. Another preprocessing step is to link consecutive trips that are close in time to each other into journeys. This yields our main dataset. Figure 1a shows the numbers

of unique passengers traveling over the course of a typical weekday. Figure 1b showsa histogram describing how many journeys were made with a single smart card. As we can see, most of the smart cards have made only a relatively low number of journeys, but there are plenty of passengers with many journeys.

4. DEMAND MODELING

When it comes to demand modeling for the simulation of public transport, a traditional approach is to use origin-destination matrices estimated from sources such as census data and manual counts of the number of passengers in some sampled vehicles [16]. The main drawback of this approach is that it becomes very difficult and expensive to measure the exact progression of passenger flows over the day. With smart card data, we know the origin, destination and exact time of travel of each individual travel, which allows for new opportunities with respect to measuring these flows.

Regarding flows of passengers in the network, we can take different approaches. The basic approach is to consider a flow through the network as a set of journeys: passengers who travel from a certain origin to a certain destination at a certain time. We will refer to this approach to demand as *trip-based* demand. However, in many cases there will be passengers who travel multiple times within the same day. In many of these cases, their consecutive journeys combine to a tour from origin to origin, with some intermediate destinations. In such cases, events happening at one of the intermediate destinations, will also influence the events in the remainder of the tour. Since our goal is to model individual passengers instead of aggregated flows, these tours contain valuable information. We will refer to this approach to demand as *tour-based* demand.

In activity-based micro-simulation, each individual traveler can be represented by an agent and this approach thus allows for microscopic analysis of a public transport system. The drawback is that we need a lot of information to model these agents. Even if we assume that all activities take place at a station, not all required information is available in the smart card data. The smart card data tells us *where*, *when* and *how* people travel, but it doesn't tell us *why* people travel, which is something that is vital to activity-based demand modeling.

Not all is lost, however: the traditional approach uses various statistical methods and interpolation techniques to fill the gaps of unknown information, in order to be able to simulate a public transport system. We can apply such an approach to the smart card data as well: we use the information which is available, such as location, modality and time of travel as much as possible and fill the gaps of information using estimation methods.

We will refer to the approach that goes beyond the notion of tour-based demand, but does not yet reach the precision of activity-based demand, as *pattern-based* demand. In a broad sense, we define pattern-based demand as demand produced by activities of such a nature that certain patterns will emerge in the travel behavior of passengers who perform the activity routinely. The most typical example of such an activity is working, since people usually work at regular times at a certain location. Other types of activities are education (which is usually bound to a schedule that may or may not change regularly), a periodic visit to family members and visiting sports events. In this paper, we will focus on patterns generated from working activities, since we believe that these will be most easy to recognize. In addition to this, we will consider educational activities with a fixed schedule as working activity, since the implications for the temporal flexibility of a passenger are usually similar. To summarize, we have:

Trip-based demand Demand with only a single journey.

Tour-based demand Demand consisting of a tour of journeys, with consecutive arrivals and departures at the same station. Also, the first and last station are equal.

Pattern-based demand Demand that exhibits a recurring pattern, produced by some regular underlying behavior of the passenger (which is possibly unkown).

4.1 Detecting customer patterns

Commuters usually live and work at the same place. This leaves patterns of frequent *home* → *work* → *home* journeys in the smart card data. We can scan consecutive journeys for these patterns. This way we can derive an activity profile for a customer. For the sake of convenience, we limit ourselves to the class of activity profiles described in the following definition:

Definition 1. Activity Profile
An activity profile is a tuple $(l, b_{pref}, e_{pref}, \delta_b, \delta_e)$ where

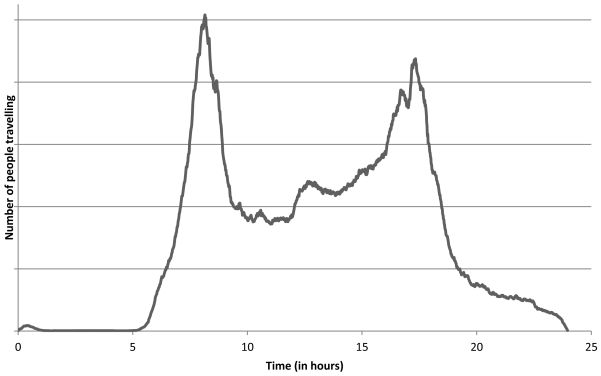
- The activity takes place at location l
- The preferred starting time of the activity is b_{pref}
- The activity will not start before $b_{pref} - \delta_b$ and not after $b_{pref} + \delta_b$
- The preferred ending time of the activity is e_{pref}
- The activity will not end before $e_{pref} - \delta_e$ and not after $e_{pref} + \delta_e$
- The preferred duration of the activity is $e_{pref} - b_{pref}$

Now for each passenger, we will try to decide whether he is commuting and what his home and working stations are. To do this, we have to make a few assumptions.

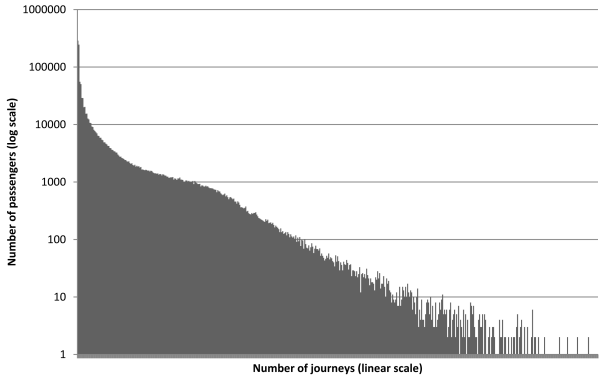
1. We assume that somebody who is commuting travels a lot. Therefore, if the number of times traveled in the considered dataset is not above a certain threshold (which should be chosen according to the length of the time period under consideration), we conclude that the passenger is not a commuter.

2. We assume that a commuter has a fixed home and a fixed location of work and that the stations associated with these locations will be the two most frequently visited stations. To be sure these frequent stations are visited more frequently than other stations, we define thresholds for the number of times they should occur.

3. We assume that, if we include weekends, someone will spend more time at home than at work. Since we can measure the time between a consecutive arrival and a departure from a station, we classify the station where the greatest amount of time is spent as the home station.



(a) Demand histogram of a weekday



(b) Histogram of the number of passengers that made a certain number of journeys within 4 months

Figure 1: Demand as observed in the smart card dataset

4. We assume flexibility in time of travel and the length of the working activity is represented by a certain amount of variation in their travel times between their home and working stations.

We use the first assumption to decide whether we will try to recognize a pattern for a certain passenger at all. The second and third assumptions can be used to recognize a passenger's home station and working station. Finally, we use the fourth assumption to model the flexibility of a passenger based on this variance. These assumptions give us the following efficient algorithm:

Algorithm: Detecting Customer Patterns.

Parameters A minimum sample size θ, thresholds t_0 and t_1 with $0 < t_0, t_1 \leq 1$

Input A set J of n journeys of a single passenger

Output A home station s and a pattern $(t, b_{pref}, e_{pref}, \delta_b, \delta_e)$ that describes a working activity profile as defined in Definition 1

Step 1 **if** $n < \theta$ **then** conclude there is no valid pattern

Step 2 Find stations a, b with maximal frequency as a start or endpoint over the journeys in J

Step 3 Denote n_a, n_b as number of journeys that have a or b as a start or endpoint, n as the total number of journeys in J

Step 4 **if** $\neg(n_a \geq t_0 n \wedge n_b \geq t_1 n)$ **then** conclude there is no valid pattern **else**

Step 4a $\Delta_a :=$ average time difference between consecutive (a, b) and (b, a) journeys

Step 4b $\Delta_b :=$ average time difference between consecutive (b, a) and (a, b) journeys

Step 4c **if** $\Delta_a \geq \Delta_b$ **then** $s := a; t := b$ **else** $s := b; t := a$

Step 5 Take the average arrival time of (s, t) journeys as preferred starting time b_{pref}

Step 6 Take the average departure time of (t, s) journeys as preferred ending time e_{pref}

Step 7 Take the standard deviation of (s, t) arrival times as the start time flexibility δ_b

Step 8 Take the standard deviation of (t, s) departure times as the ending time flexibility δ_e

Step 9 **return** $s, (t, b_{pref}, e_{pref}, \delta_b, \delta_e)$

It is not difficult to see that each of the steps can be performed in time linear with respect to the set of journeys J, except for Step 2, where we have to calculate frequency statistics. To take the first and second most frequent station, we can sort the stations based on their frequencies. Since at most $O(n)$ station occur in J, this gives a $O(n \log n)$ time bound. In [20], it is discussed that this selection problem takes $O(n \log n)$ time in general. Since there are no loops in the algorithm, we may conclude that it runs in $O(n \log n)$ time for a single passenger with n journeys.

4.2 Deriving the Agent population

We will now discuss how to derive an agent population from our dataset. In the beginning of Section 4, we discussed the difference between *trip-based*, *tour-based* and *pattern-based* demand. Since there are smart cards that are used only once and passengers who have highly irregular travel patterns (because they don't use public transport to commute), we will not be able to derive a pattern for each customer and we may not even be able to find a tour in the data for each customer. Therefore, we will take a step-wise approach, where we first try to calculate a pattern for a passenger. If this succeeds, we will generate demand for this passenger based on the pattern we found. If we fail to find a pattern, we search for a tour and generate tour-based demand by introducing dummy activities at the intermediate stations of the tour. If we even fail to find a tour, we will generate trip-based demand by generating agents for each trip the customer made.

We will choose a single day (preferably not during the weekend) to model. We first filter our dataset such that we only retain customers that have traveled on that day. After filtering, we decompose our dataset into three parts: one group contains customers of which we know a lot, one

group contains customers of which we have a tour and lastly, one group of customers with a single or unpredictable travel pattern. For each customer, we will have to generate an activity plan for the day. We will take a different approach to the generation of plans for each group of customers.

A plan for the day is a list of activity profiles with planned ending times for all activities. There is one exception: the last activity of the agent should be a home activity, which has no ending time. The ending time in the plan of an agent may differ from the ending times in the activity profile: an agent may try to deviate from his preferred time if this gives him an improvement in utility. The planned ending time is exactly what allows the agent to do this. When we start generating plans for our agent population, we will initially stick with the preferred ending times from the activity profiles as the planned ending times. For the group of customers for which we have derived a pattern, we can generate a *home* → *work* → *home* activity plan. For the group of customers for which we only have a tour, we only have a set of locations. For the activity profiles, we can easily derive a starting and ending time, using the check-out and check-in time at each intermediate station. The flexibility is a problem, however. For the time being, we decide to select a global value for the δ_b and δ_e of tour-based agents. We take a similar approach with the trip-based customers, where we generate a single agent for each trip. For each journey we observe from u to v, we generate an agent with a *home*→ *dummy*→ *home* pattern, where the first home activity should be performed at location u and the dummy and last home activity should be performed at location v. This gives us the following efficient algorithm for demand generation:

Algorithm: Generation of Demand.

Input A day d and a set of customers C with for each $c \in C$ their respective set of journeys J_c

Output An agent population for day d

Step 1 $P := \{p : p \in C, J_c$ contains a journey during day $d\}$

Step 2 $P_{pat} := \{p : p \in P, J_p$ has a pattern $\}$

Step 3 $P_{tour} := \{p : p \in P \backslash P_{pat}, J_p$ makes a tour at day $d\}$

Step 4 $P_{trip} := P \setminus (P_{pat} \cup P_{tour})$

Step 5 Initialize agent set $A := \emptyset$

Step 6 for each $p \in P_{pat}$

Step 6a Generate an agent with a "*home* → *work* → *home*" plan

Step 6b Add the agent to A

Step 7 for each $p \in P_{tour}$

Step 7a Generate an agent with a plan containing the tour locations and ending times of p's tour at day d

Step 7b Add the agent to A

Step 8 for each $p \in P_{trip}$, **for each** (u, v) journey traveled by p on d

Step 8a Generate an agent with a "*home* (at u) → *dummy* (at v) → *home* (at v)" plan of which the *dummy* activity should start at the check-out time of the journey

Step 8b Add the agent to A

Step 9 return A

The running time of this algorithm is proportional to the size of the J_c sets. Let us define $n = \sum_{c \in C} |J_c|$. If we define $k = |C|$ as the number of customers and m as the maximum number of journeys for a single customer, we can easily see that $n \leq mk$. Steps 1-3 are regular filtering steps, that can be performed by examining each set J_c or by applying the earlier algorithm and can therefore all run in $O(mk \log m) = O(n \log m)$ time. The loops in steps 6-8 each iterate at most over k customers and generating the plan for each customer can be done in $O(m)$ time. Therefore, steps 6-8 run in $O(mk) = O(n)$ time as well. Therefore, the whole algorithm runs in $O(n \log m)$ time.

5. SIMULATION

5.1 MATSim

For our agent-based simulation, we used the MATSim 0.3.0 software package. To run a MATSim based simulation, we need three ingredients: the agent population, a network describing how vehicles can travel between nodes and a public transportation schedule. When we start the simulation, all agents calculate an initial plan. The main loop consists of a simulation and a replanning phase. During the replanning phase, each agent can adapt his activity plan. They do so by using certain modules available in MATSim, called mutators. During the simulation phase, all plans are executed and all events related to movements and activities of agents and vehicles are generated. The mutators used by the agents to adapt their plans, can be given individual probabilities. An example of such mutators are the rerouting mutator, that recalculates the fastest route between activities based on the network congestion of the previous day. Another example is the time mutator, that shifts the planned starting and ending times of the activities randomly, while retaining their sequential order.

Recently, the mobility simulation of MATSim has been extended with support for public transport [18]. This mobility simulation is an extension of the road-traffic simulation. In MATSim, model public transport vehicles as cars with a driver and a lot of space for additional passengers moving over a network that is given as input.

To generate the required network, we used a list of stations with their geographical locations and the available schedule information for all three modalities. We add the stations as nodes in the network. If there was a vehicle that visited two stations consequently in the schedule, we added a link between the two stations, with the distance of the link based on Euclidean distance between the two stops. We enforce the vehicles to wait at each stop until their scheduled time of departure. The mobility simulation itself is a discrete event simulation through a queuing network generated from the input network.

MATSim allows us to transfer money from or to an agent, but this mechanism is not triggered automatically. We added a module that imposes fares on the agents. It keeps track

θ	pattern	tour	trip
80	26%	32%	42%
120	14%	40%	46%
160	4%	48%	48%
200	1%	50%	49%
∞	0%	50%	50%

Table 1: Population distributions for different θ values

of the moments agents enter and exit the vehicles and the distances traveled by the vehicles. The fare of a journey consists of a *base tariff* that is the same for all journeys and a *distance tariff* with a certain fixed amount per meter traveled. An additional aspect is the transfer time: if the check-out time and check-in time of two consecutive journeys is small enough, the agent doesn't have to pay the *base tariff* a second time. At the end of the simulation of a single day, the accumulated fares are billed to the agent and transformed into disutilities during the evaluation of the executed plan. The utility function itself is described in [11]. The main idea is that traveling gives a disutility, while performing an activity gives utility. We did not yet implement an extension of this scoring function that assigns a personal price sensitivity to each agent, so this is currently a common parameter for all agents. We used 6 and -6 as the (global) coefficients for the performing and traveling utilities and -18 as the coefficient for late arrival.

5.2 Experimental Setup

We ran our experiments on a desktop PC with a quad-core Intel Core 2 Quad Q6600 processor and 8 GB of RAM running Windows 7 Professional SP1, 64-bit. Since we want our passengers to have their working station in at least half of their journeys and we want their home station to be at least as frequent as their working station, we chose $t_0 = 0.6$ and $t_1 = 0.5$. Prior to our experiments, we generated populations for different values of θ. We examined a few possible values, of which the distributions are presented in Table 1. Since $\theta = 80$, $\theta = 120$ and $\theta = \infty$ give us the greatest variations, we chose these for our experiments. For the tour-based and trip-based demand, we wanted our agents to keep as close as possible to their observed travel time, so we fixed δ_b and δ_b for their activity patterns to 5 minutes. This gives us a total of three different agent populations.

For our pricing strategy, we took figures inspired by the real world pricing policies. We set the base tariff to 0.75 and the distance based tariff to 0.115. The allowed transfer time is set to 30 minutes. For our experimentation with revenue management policies, we ran each of our populations through the network two times: once with a single tariff over the full day and once with a discount of 1% outside the peak hours (the peak hours are between 7:00–9:00 and 16:00–19:00). We chose 1% because our agents will always try to optimize their utility, even if the increase is very small. The check-in time determines whether the discount is given. To allow agents to shift their times, we enabled MATSim's time mutator module. Running each population against both pricing policies, we get a total of six experiments.

After some preliminary experiments, we saw that the increase of agent-utilities slowed down significantly between the 60th and 100th iteration. To be sure our simulation reached a state that is close to an equilibrium, we ran each simulation up until the 180th iteration.

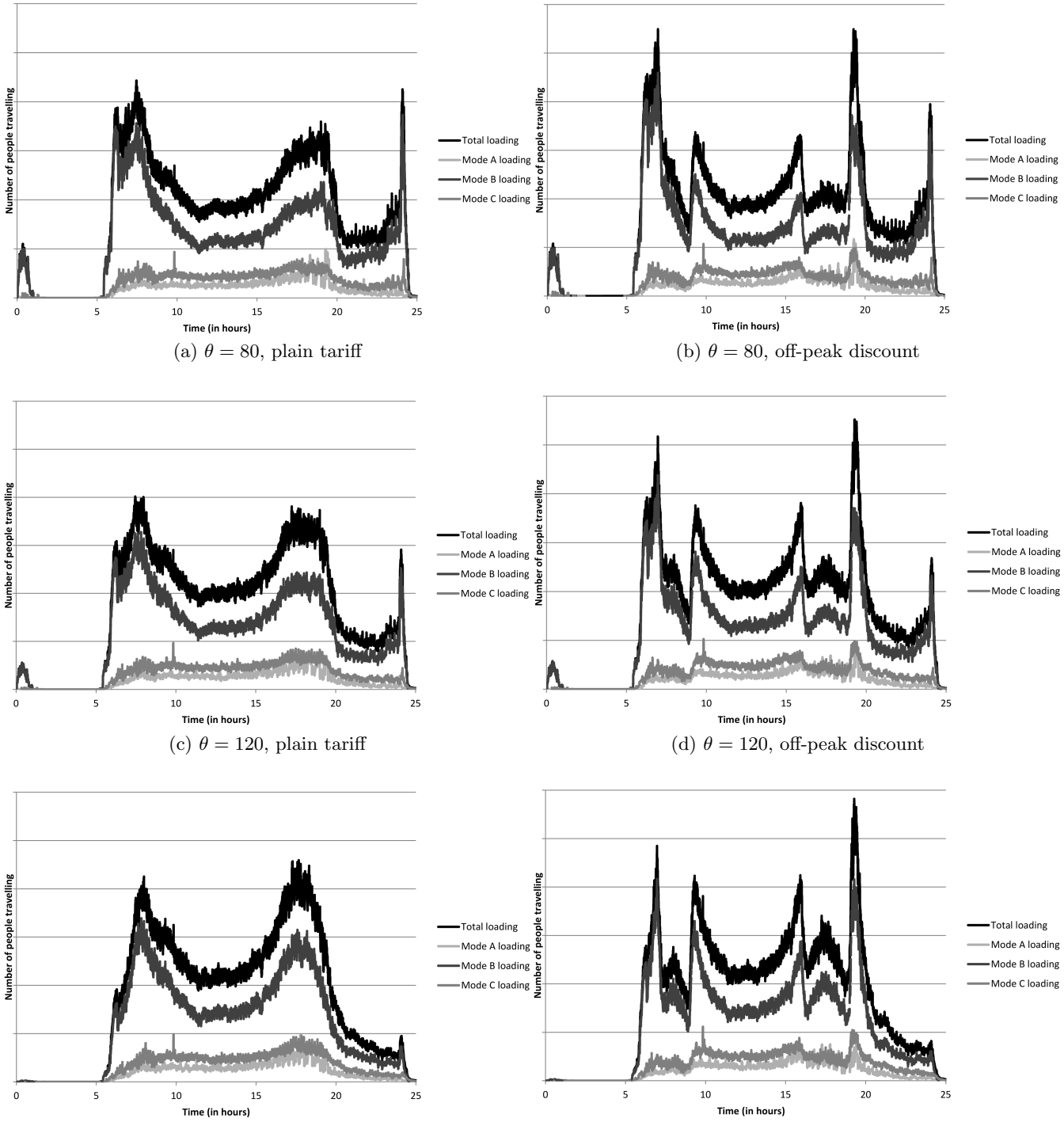
6. RESULTS

Generating our agent population could be done very efficiently from our sorted dataset of journeys that we derived in Section 3. The average time required to process this full set of 27 million journeys and write the agent population to MATSim input files was on average 107 seconds. Our simulation could roughly execute a complete iteration of the mobility simulation in a little less than two minutes. Some additional time was needed for finding all the shortest routes through the transit network and dumping all the plans of the agents after each 10th iteration. A complete run of a single scenario took roughly five hours. The vehicle loadings observed after the first iteration were in all of our six scenarios relatively similar to Figure 2e. But at the 180th iteration, we saw notable differences.

Let us first consider the case where we have a homogeneous pricing strategy over the whole day. When we move from $\theta = 80$ (Figure 2a) to $\theta = 120$ (Figure 2c), we see that the peak during the morning peak becomes a bit smaller, while the evening peak becomes a plateau that is a bit wider. This implies that, as soon as we treat some passengers who where pattern based in the $\theta = 80$ case as tour or trip-based during the $\theta = 120$ case, they tend to move away from the morning peak, but towards the evening peak. When we increase θ to ∞ (Figure 2e), we see that the morning peak increases a bit and the evening peak increases a lot. This suggests that some of the pattern-based agents in the $\theta = 120$ case actually traveled during the morning peak in the $\theta = \infty$ case, where they were less flexible.

One thing that should be noted is the high peak close to the end of the day in both Figure 2a and Figure 2b. This is a clipping artifact and implies that a certain group of agents prefers to travel at the end of the day and suggests there is a problem with the calibration of these agents. Although the problem decreases when we increase θ, the problem does not disappear entirely, even when we have $\theta = \infty$. We ignore this problem for the time being, but it suggests that we should be careful in drawing conclusions based on these results, and it is an issue that should be addressed in the future.

Now let us consider the scenarios where we discounted the off-peak hours. The most obvious result is the fact that this generates new peaks outside the peak-hour windows that are even higher than the rush hour peaks. This implies that even with a relatively small 1% discount, most of the agents have an incentive to divert from their initial plans. There can be two reasons for this behavior: either the agent is flexible enough to divert without losing utility, or the disutility of being early or late is smaller than the utility gained from the discount. We can study the result of decreasing the flexibility by comparing the results for $\theta = 80$ (Figure 2b) with the results for $\theta = 120$ (Figure 2d). A noticeable difference can be observed in the patterns that emerge within the peak-hour time windows. The evening peak in Figure 2d has a triangular structure, compared to the $\theta = 80$ case. When we increase θ to ∞, we get this triangular pattern in the morning peak as well and the effect is even stronger in the evening peak. Since all agents will travel by public transport and many agents diverted to the off-peak hours, the discount resulted in a drop in revenue.



(a) $\theta = 80$, plain tariff

(b) $\theta = 80$, off-peak discount

(c) $\theta = 120$, plain tariff

(d) $\theta = 120$, off-peak discount

(e) $\theta = \infty$, plain tariff

(f) $\theta = \infty$, off-peak discount

Figure 2: Vehicle loadings after 180 iterations for different sample size thresholds θ. On the horizontal axis, the time of day is displayed. On the vertical axis, the number of people currently travelling is displayed.

7. DISCUSSION

Our results show that our proposed method of generating an agent population from a smart card dataset and performing a microscopic simulation where each customer is presented by an agent is achievable within a reasonable amount of time. Generating the agent population and performing a single run of the simulation (given that all routes are calculated) both take under two minutes of time.

The results themselves show that the agents in our population react heavily to our discounted pricing policy, even if we have very inflexible agents in the $\theta = \infty$ case. However, we see that a certain number of agents still prefers to travel within the more expensive time window and in case of the $\theta = 120$ and $\theta = \infty$ cases, we see a triangular peak emerging within the peak hours. This suggests there is a population of agents for which the utility of arriving late is worse than the fare reduction. This should typically hold for agents who have to make a short distance trip, because for these agents the fare reduction is relatively low, compared to agents who have to travel a longer distance. This holds for real life passengers as well: a reduction on a small fare is of course much smaller than a reduction on a large fare. However, we can argue that the response of the agents is still too radical. We think the simulation will benefit greatly from calibration and utility functions that are not entirely linear with regard to the fare (especially when comparing prices, humans tend to disregard small price differences to some extent). Adding individual price sensitivities to our population of agents will be another way to improve in this regard.

When we compare the differences between our populations for different values of θ, we see that all populations maintain the property that during the typical peak hours demand is greatest. The value of θ seems to have the biggest impact on the evening peak. For lower values of θ, this part of the demand spreads outs to a much larger extent than the morning peak. This corresponds to the observation that usually the evening peak is longer in time and not as sharp as the morning peak. This is something which we can observe to some extent in Figure 1a as well. The main issue with the lower values of θ seems to be that we get greater clipping artifacts around 5:00 and 24:00. We hope this can be addressed by calibrating the utility functions, or by limiting the flexibility of agents when we come across individuals with extreme cases.

In the remainder of this section, we will discuss different topics for future research. In Section 7.1 we discuss how to improve the demand generation itself. We discuss the possibilities with regard to calibration of the parameters used by the simulation in Section 7.2. Section 7.3 addresses the question how we can incorporate additional datasets in order to distinguish different types of activities. Finally, Section 7.4 addresses the issue of validation.

7.1 Demand Generation

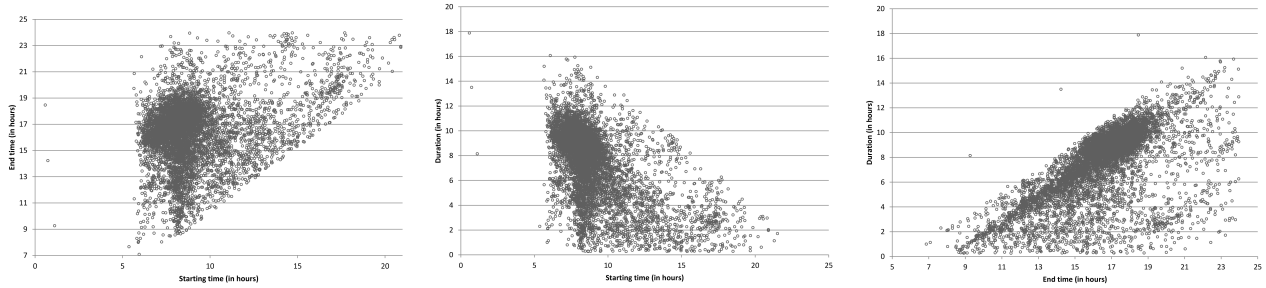
First of all, we must consider our demand generation algorithm. In our algorithm we make a couple of assumptions. The assumption that people who commute travel a lot, is very fundamental and probably realistic. The assumption that commuters have a fixed home and fixed working location probably often holds, but may be relaxed a bit: it can be broken by people who have more than one place to spend the night, or who have a job that has different locations that get visited in regular patterns. With enough observations, it may be possible to detect such patterns as well. The assumption that people spend more time at home probably holds often as well, but we must be careful with regard to outliers: it may be possible that somebody switches mode while at work (either by taking a bike or a car). In such an event, it would be possible that our approach reveals that somebody stayed for days at his working station, while this was not true in reality. The assumption that the variation in travel behavior of a passenger reflects his flexibility with regard to travel time is the most doubtful. Studies with more information regarding this assumption would be extremely valuable in improving our demand generation process.

While we have shown that our approach can efficiently generate an agent population from a real life smart card dataset, the fact that we have taken an approach that is very efficient and straightforward to implement has the disadvantage of being relatively crude. One may argue that we can introduce sophisticated pattern recognition and data-mining techniques in this process, in order to generate an agent population that is closer to reality. One area for future improvement is that we use the average starting time and ending time of working activities, but ignore their possible correlations. In Figure 3 we can examine the scatter plot of the durations, starting and ending times of working activities performed by pattern-based individuals within the four months of our dataset. While a more thorough analysis is necessary, it seems probable that some correlations can be exploited. Improving our method in this regard is a priority, since we believe that this is useful information to make the behavior of the agent population more realistic.

7.2 Calibration

In order to reflect real life behavior more closely, calibration of our simulation is a required to use it in a decision support setting. The single global price elasticity for all agents is something that should be implemented on an individual level. We can do this in two ways. We can change the program to specify a utility function of each individual agent. Alternatively, we can adapt our fare-module to mimic price sensitivity. We can use a personal transformation function for each agent that scales the fares down for insensitive agents and scales the fares up for sensitive agents. Another kind of sensitivity that is valuable to model, is the sensitivity to the crowdedness of vehicles. If vehicles become too crowded, additional delay can induce delays in the public transportation system. This aspect was mostly ignored in our current simulation.

The right values for the price elasticities will be very difficult to estimate from only check-ins and check-outs. The main problem in this regard is the fact that we do not know what possible alternatives were available and have been considered by the passenger, before he made his journey. In the field of discrete choice modeling, this kind of data is referred to as *revealed choice* data. In situations where surveys are conducted and the subjects are exposed to multiple alternatives from which they must select a single option, we get *stated choice* data. Within the field of discrete choice modeling, most of the research effort has been performed on analyzing stated choice data. This allows us to accurately and efficiently estimate properties such as price elasticities within a population. In our case, it would be necessary to combine information obtained from stated choice experiments to calibrate the simulation obtained from revealed



(a) Starting time vs ending time (b) Starting time vs duration (c) End time vs duration

Figure 3: Correlations of starting times, ending times and duration ($\theta = 120$)

choice data. Some literature on how to combine revealed and stated choice data has been published during the 1990's ([6] and [3] are two examples). However, [17] reveals that there is little research in this area concerning smart card data.

7.3 Extensions based on additional datasets

One possible way to move our pattern based demand closer to real activity based demand models is by combining the smart card dataset with other datasets. A promising approach might be to look at regional information of stations. We could use such datasets to construct profiles of stations, which would allow us to make better guesses with regard to the activities that can be performed around the stations. If a station is close to a large industrial plant or office buildings, it is very probable that passengers traveling there do so because they have to work. A station close to a shopping mall will not only attract the employees of the shops, but customers as well. Local stations that coincide with a railway station or an airport are likely to attract passengers that want to travel further, or want to travel home. Stations in residential areas will likely serve as home stations, or as stations that get visited by passengers who want to visit friends or family. We propose to use data provided by the OpenStreetMap project [2], since this contains tags with information on available activities at certain locations.

After we generate profiles for all our stations with such information, we can take this information into account while recognizing patterns. This would allow us to make better guesses of the temporal flexibility of passengers for which we don't have a large enough set of journeys. Suppose we observe a passenger who starts his day with a journey from a residential area to an area with a lot of office buildings and stays there for 6 hours, then travels to an area with a shopping mall and stays there for 1 hour, after which he travels home. Even if we never observed any other journeys by this passenger, we can still make an educated guess about what he was doing and thus to what extend he could have been flexible. However, this calls for much more sophisticated statistical models than the one we are currently using. Depending on the kind of questions we want to study, it may or may not be worth the effort to go this far.

7.4 Validation

Validating a simulation like this is not a trivial task. One aspect that we can validate is the question whether the simulation can be used as a predictive tool for the movement of passengers through a public transportation network. The straightforward way to do this is by splitting the dataset at a certain moment in time. We can then use the first part of the dataset to generate agent populations and compare the outcomes to what is observed in the second part of the dataset. At first, we should choose a moment within a period where no policy and scheduling changes have occurred. If we can pass this test, we can raise the bar by choosing the moments at which a policy change has occurred, such as the introduction of a new schedule or new pricing schemes.

Another aspect that we may want to validate, is the question whether the emerging activity patterns of the agents represent the real-life activity patterns of the passengers represented by the agents. Validating this aspect requires much greater effort than validating the movements of passengers. One approach could be to use survey data containing activity logs registered in diaries and compare the diaries to the activity plans in the simulation. There may be some privacy issues with this approach, since it would require that we link the smart card id's to the participants, in order to match a diary to an agent. A possible workaround is to generate faux check-in/check-out data from the diaries by generating a check-in and a check-out for the journeys documented in the diaries. We could then use this dataset as if it were a smart card dataset and investigate to what extend the generated activity patterns of the agents reproduce the original activity plans.

In a similar way, we can consider the study of other location tracking datasets, such as triangulation logs from mobile phone operators or the location logs from the mobile phones themselves. The main advantage is that such a dataset contains more details on the whereabouts of individuals, which gives more opportunity to estimate what they are doing. For example, using smart card data we may observe that a person checks out at a station near a shopping mall and checks in four hours later. However, we have no data to decide whether it is probable that this person has been shopping or that this person has been working as an employee at one of the stores. If we have a mobile phone log, we may observe that the person has visited a great number of stores during these four hours. This would be evidence that he was not working as an employee.

8. CONCLUSIONS

We have shown how we can use smart card data to generate different types of demand. We developed an agent-based simulation that allows us to analyze the movements of the agents through our multimodal public transportation network. We experimented with different settings for the

number of trip-based agents and with a 1% discount in the off-peak hours. Finally, we discussed several opportunities for future research.

As soon as we sorted our dataset in such a way that we could process all journeys customer by customer in chronological order, demand generation could be done very efficiently. We used simple rules to determine whether a customer should be modeled using *trip* based, *tour* based or *pattern* based demand. We have evaluated the impact of different thresholds for the *pattern* based customers on the resulting approximate equilibrium. We have also seen that an off-peak discount can be used to let a part of the agent population shift their travel times. In our case, this lead to a lower revenue. However, the effect on the required capacity must be taken into account when making a tradeoff between costs and revenue.

There are many opportunities for future research. First, our simulation can greatly benefit from proper calibration. Additionally, our method for demand generation can be improved upon, both by taking a closer look at the smart card data itself using more advanced techniques and by combining the smart card data with additional datasets. Including heterogeneity in the price sensitivity of the agents would be another improvement over the current situation. Finally, the simulation should be validated. We believe that an improved version of our simulation can be helpful in both the design of revenue management systems, including location based and modality based tariff schemes and other fields of study within a public transport context.

9. ACKNOWLEDGMENTS

We would like to thank Kai Nagel and his colleagues for their comments and support with the simulation. This research is sponsored by the Dutch National Science Foundation (NWO) Complexity Grant (#600.645.000.09). We also wish to thank Netherlands Railways for their support.

10. REFERENCES

[1] Multi-agent transport simulation toolkit, 2012. http://www.matsim.org.

[2] Openstreetmap, 2012. http://www.openstreetmap.org.

[3] W. Adamowicz, J. Louviere, and M. Williams. Combining revealed and stated preference methods for valuing environmental amenities. *Journal of environmental economics and management*, 26(3):271–292, 1994.

[4] K. Axhausen, M. Balmer, and F. Ciari. A new mode choice model for a multi-agent transport simulation. In *8th Swiss Transport Research Conference. ETH, Eidgenössische Technische Hochschule*, 2008.

[5] K. Axhausen, A. Zimmermann, S. Schönfelder, G. Rindsfüser, and T. Haupt. Observing the rhythms of daily life: A six-week travel diary. *Transportation*, 29(2):95–124, 2002.

[6] M. Ben-Akiva, M. Bradley, T. Morikawa, J. Benjamin, T. Novak, H. Oppewal, and V. Rao. Combining revealed and stated preferences data. *Marketing Letters*, 5(4):335–349, 1994.

[7] M. Ben-Akiva and S. Lerman. *Discrete choice analysis: theory and application to travel demand*, volume 9. The MIT press, 1985.

[8] N. Ben-Khedher, J. Kintanar, C. Queille, and W. Stripling. Schedule optimization at SNCF: From conception to day of departure. *Interfaces*, pages 6–23, 1998.

[9] J. Bowman and M. Ben-Akiva. Activity-based disaggregate travel demand model system with activity schedules. *Transportation Research Part A: Policy and Practice*, 35(1):1–28, 2001.

[10] G. D. B. Cameron and G. I. D. Duncan. Paramics - parallel microscopic simulation of road traffic. *The Journal of Supercomputing*, 10:25–53, 1996. 10.1007/BF00128098.

[11] D. Charypar and K. Nagel. Generating complete all-day activity plans with genetic algorithms. *Transportation*, 32:369–397, 2005. 10.1007/s11116-004-8287-y.

[12] H. Link. Pep–a yield-management scheme for rail passenger fares in Germany. *Japan Railway & Transport Review (March)*, (38):50–55, 2004.

[13] K. Meister, M. Balmer, F. Ciari, A. Horni, M. Rieser, R. Waraich, and K. Axhausen. Large-scale agent-based travel demand optimization applied to Switzerland, including mode choice. In *12th World Conference on Transportation Research, Lisbon, July 2010*, 2010.

[14] E. Miller. Microsimulation and activity-based forecasting. In *Activity-Based Travel Forecasting Conference*, 1997.

[15] C. Morency, M. Trépanier, and B. Agard. Measuring transit use variability with smart-card data. *Transport Policy*, 14(3):193–203, 2007.

[16] K. Müller and K. Axhausen. *Population synthesis for microsimulation: State of the art*. ETH Zürich, Institut für Verkehrsplanung, Transporttechnik, Strassen-und Eisenbahnbau (IVT), 2010.

[17] M. P. Pelletier, M. Trépanier, and C. Morency. Smart card data use in public transit: A literature review. *Transportation Research Part C: Emerging Technologies*, 19(4):557 – 568, 2011.

[18] M. Rieser. *Adding transit to an agent-based transportation simulation*. PhD thesis, Technical University Berlin, Berlin, 2010.

[19] C. Rommel. Automatic feedback control applied to microscopically simulated traffic the potential of route guidance in the berlin traffic network. Technical report, VSP Working Paper, 2007.

[20] S. S. Skiena. *The Algorithm Design Manual*. Springer, 2nd edition, Aug. 2008.

[21] K. Talluri and G. Van Ryzin. *The theory and practice of revenue management*, volume 68. Springer Verlag, 2005.

[22] M. Treiber, A. Hennecke, and D. Helbing. Congested traffic states in empirical observations and microscopic simulations. *Physical Review E*, 62(2):1805, 2000.

[23] M. Trépanier, N. Tranchant, and R. Chapleau. Individual trip destination estimation in a transit smart card automated fare collection system. *Journal of Intelligent Transportation Systems*, 11(1):1–14, 2007.

Decentralized Data Fusion and Active Sensing with Mobile Sensors for Modeling and Predicting Spatiotemporal Traffic Phenomena

Jie Chen[†], Kian Hsiang Low[†], Colin Keng-Yan Tan[†], Ali Oran[§], and Patrick Jaillet[‡]
Department of Computer Science, National University of Singapore, Republic of Singapore[†]
Singapore-MIT Alliance for Research and Technology, Republic of Singapore[§]
Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, USA[‡]
{chenjie, lowkh, ctank}@comp.nus.edu.sg[†]
aoran@smart.mit.edu[§], jaillet@mit.edu[‡]

ABSTRACT

The problem of modeling and predicting spatiotemporal traffic phenomena over an urban road network is important to many traffic applications such as detecting and forecasting congestion hotspots. This paper presents a decentralized data fusion and active sensing (D^2FAS) algorithm for mobile sensors to actively explore the road network to gather and assimilate the most informative data for predicting the traffic phenomenon. We analyze the time and communication complexity of D^2FAS and demonstrate that it can scale well with increasing number of observations when the number of sensors is large. We provide a theoretical guarantee on its predictive performance to be equivalent to a sophisticated centralized approximate Gaussian process prediction model. This result implies that the computational load of the centralized model can be distributed among the mobile sensors, thereby achieving efficient and scalable prediction. Empirical evaluation on a real-world traffic phenomenon dataset over an urban road network shows that our D^2FAS algorithm is significantly more time-efficient and scalable (i.e., in the number of observations and sensors) than existing state-of-the-art algorithms while achieving comparable predictive performance.

1. INTRODUCTION

Knowing and understanding the traffic conditions and phenomena over road networks has become increasingly important to the goal of achieving smooth-flowing, congestion-free traffic, especially in densely-populated urban cities. According to a 2011 urban mobility report [28], the traffic congestions in the USA have caused 1.9 billion gallons of extra fuel, 4.8 billion hours of travel delay, and $101 billion of delay and fuel cost. Such huge resource wastage can be potentially mitigated if the spatiotemporally varying traffic phenomena (e.g., speeds and travel times along road segments) are predicted accurately enough in real time to detect and forecast the congestion hotspots; network-level (e.g., ramp metering, road pricing) and user-level (e.g., route replanning) measures can then be taken to relieve these congestions, so as to improve the overall efficiency of road networks.

In practice, it is non-trivial to achieve real-time, accurate prediction of a spatiotemporally varying traffic phenomenon because the quantity of sensors that can be deployed to observe an entire road network is cost-constrained. Traditionally, static sensors such as loop detectors [9, 34] are placed at designated locations in a road network to collect data for predicting the traffic phenomenon. However, they provide sparse coverage (i.e., many road segments are not observed, thus leading to data sparsity), incur high installation and maintenance costs, and cannot reposition by themselves in response to changes in the traffic phenomenon. Low-cost GPS technology allows the collection of traffic data using passive mobile probes [35] (e.g., taxis/cabs). Unlike static sensors, they can directly measure the travel times along road segments. But, they provide fairly sparse coverage due to low GPS sampling frequency (i.e., often imposed by taxi/cab companies) and no control over their routes, incur high initial implementation cost, pose privacy issues, and produce highly-varying speeds and travel times while traversing the same road segment due to inconsistent driving behaviors. A critical mass of probes is needed on each road segment to ease the severity of the last drawback [30] but is often hard to achieve on non-highway segments due to sparse coverage. In contrast, we propose the use of active mobile probes [33] to overcome the limitations of static and passive mobile probes. In particular, they can be directed to explore any segments of a road network to gather traffic data at a desired GPS sampling rate while enforcing consistent driving behavior.

How then do the mobile probes/sensors actively explore a road network to gather and assimilate the most informative observations for predicting the traffic phenomenon? There are three key issues surrounding this problem, which will be discussed together with the related works:

Models for predicting spatiotemporal traffic phenomena. The spatiotemporal correlation structure of a traffic phenomenon can be exploited to predict the traffic condi-

tions of any unobserved road segment at any time using the observations taken along the mobile sensors' paths. To achieve this, existing Bayesian filtering frameworks [2, 34, 35] utilize various handcrafted parametric models predicting traffic flow along a highway stretch that only correlate adjacent segments of the highway. Hence, their predictive performance will be compromised when the current observations are sparse and/or the actual spatial correlation spans multiple segments. Their strong Markov assumption further exacerbates this problem. It is also not demonstrated how these models can be generalized to work for arbitrary road network topologies and more complex correlation structure. Existing multivariate parametric traffic prediction models [8, 18] do not quantify uncertainty estimates of the predictions and impose rigid spatial locality assumptions that do not adapt to the true underlying correlation structure.

In contrast, we assume the traffic phenomenon over an urban road network (i.e., comprising full range of road types like highways, arterials, slip roads, etc.) to be realized from a rich class of Bayesian non-parametric models called the *Gaussian process* (GP) (Section 2) that can formally characterize its spatiotemporal correlation structure and refine it with growing number of observations [21]. More importantly, GP can provide formal measures of predictive uncertainty (e.g., based on variance or entropy criterion) for directing the mobile sensors to explore highly uncertain areas of the road network. The work of [9] used GP to represent the traffic phenomenon over a network of only highways and defined the correlation of speeds between highway segments to depend only on the geodesic (i.e., shortest path) distance of these segments with respect to the network topology. Different from the work of [9], we further improve the correlation structure of GP by enabling it to exploit road segment features (e.g., length, number of lanes, direction, speed limit) for differentiating road types, which is not found in the works described above.

Data fusion. The observations are gathered distributedly by each mobile sensor along its path in the road network and have to be assimilated in order to predict the traffic phenomenon. Since a large number of observations are expected to be collected, a centralized approach to GP prediction cannot be performed in real time due to its cubic time complexity.

To resolve this, we propose a decentralized data fusion approach to efficient and scalable approximate GP prediction (Section 3). Existing decentralized and distributed Bayesian filtering frameworks for addressing non-traffic related problems [3, 4, 20, 26, 32] will face the same difficulties as their centralized counterparts described above if applied to predicting traffic phenomena, thus resulting in loss of predictive performance. Distributed regression algorithms [7, 22] for static sensor networks gain efficiency from spatial locality assumptions, which cannot be exploited by mobile sensors whose paths are not constrained by locality. The work of [5] proposed a distributed data fusion approach to approximate GP prediction based on an iterative Jacobi overrelaxation algorithm, which incurs some critical limitations: (a) the past observations taken along the mobile sensors' paths are assumed to be uncorrelated, which greatly undermines its predictive performance when they are in fact correlated and/or the current observations are sparse; (b) when the number of robots grows large, it converges very slowly; (c) it assumes that the range of positive correlation has to be bounded by some factor of the communication range. Our proposed decentralized algorithm does not suffer from these limitations and can be computed exactly with efficient time bounds.

Active sensing. The mobile sensors have to actively gather the most informative observations for minimizing the uncertainty of modeling and predicting the traffic phenomenon. Existing centralized [13, 14, 15] and decentralized [12, 31] active sensing algorithms scale poorly with increasing number of observations and/or mobile sensors. We propose a decentralized active sensing algorithm that overcomes these issues of scalability (Section 4).

This paper presents a novel *Decentralized Data Fusion and Active Sensing* (D^2FAS) algorithm (Sections 3 and 4) for sampling spatiotemporally varying environmental phenomena with mobile sensors. Note that the decentralized data fusion component of D^2FAS can also be used for static and passive mobile sensors. The practical applicability of D^2FAS is not restricted to traffic monitoring; it can be used in other environmental sensing applications such as precision agriculture, mineral prospecting [16], monitoring of ocean and freshwater phenomena [6, 23, 17] (e.g., plankton bloom, anoxic zones), forest ecosystems, pollution (e.g., oil spill), or contamination (e.g., radiation leak). The specific contributions of this paper include:

- Analyzing the time and communication overheads of D^2FAS (Section 5): we prove that D^2FAS can scale better than existing state-of-the-art algorithms with increasing number of observations when the number of sensors is large;
- Theoretically guaranteeing the predictive performance of the decentralized data fusion component of D^2FAS to be equivalent to that of a sophisticated centralized approximate GP prediction model (Section 3). This result implies that the computational load of the centralized model can be distributed among the mobile sensors, thereby achieving efficient and scalable prediction;
- Improving the correlation structure of GP model by enabling it to exploit road segment features (e.g., length, number of lanes, direction, and speed limit) and the road network topology (Section 2.1);
- Empirically evaluating the predictive performance, time efficiency, and scalability of D^2FAS algorithm on a real-world traffic phenomenon (i.e., speeds of road segments) dataset over an urban road network (Section 6): D^2FAS is more time-efficient and scales significantly better with increasing number of observations and sensors while achieving predictive performance close to that of existing state-of-the-art algorithms.

2. GAUSSIAN PROCESS REGRESSION OVER GRAPH

The *Gaussian process* (GP) can be used to model a spatiotemporal traffic phenomenon over a road network as follows: The traffic phenomenon is defined to vary as a realization of a GP. Let V be a set of road segments representing the domain of the road network such that each road segment $s \in V$ is specified by a p-dimensional vector of features and

is associated with a realized (random) measurement z_s (Z_s) of the traffic condition such as speed if s is observed (unobserved). Let $\{Z_s\}_{s\in V}$ denote a GP, that is, every finite subset of $\{Z_s\}_{s\in V}$ follows a multivariate Gaussian distribution [25]. Then, the GP is fully specified by its *prior* mean $\mu_s \triangleq \mathbb{E}[Z_s]$ and covariance $\sigma_{ss'} \triangleq \mathrm{cov}[Z_s, Z_{s'}]$ for all $s, s' \in V$. In particular, we will describe in Section 2.1 how the covariance $\sigma_{ss'}$ for modeling the correlation of measurements between all pairs of segments $s, s' \in V$ can be designed to exploit the road segment features and the road network topology.

A chief capability of the GP model is that of performing probabilistic regression: Given a set $D \subset V$ of observed road segments and a column vector z_D of corresponding measurements, the joint distribution of the measurements at any set $Y \subseteq V \setminus D$ of unobserved road segments remains Gaussian with the following *posterior* mean vector and covariance matrix

$$\mu_{Y|D} \triangleq \mu_Y + \Sigma_{YD}\Sigma_{DD}^{-1}(z_D - \mu_D) \quad (1)$$

$$\Sigma_{YY|D} \triangleq \Sigma_{YY} - \Sigma_{YD}\Sigma_{DD}^{-1}\Sigma_{DY} \quad (2)$$

where μ_Y (μ_D) is a column vector with mean components μ_s for all $s \in Y$ ($s \in D$), Σ_{YD} (Σ_{DD}) is a covariance matrix with covariance components $\sigma_{ss'}$ for all $s \in Y, s' \in D$ ($s, s' \in D$), and Σ_{DY} is the transpose of Σ_{YD}. The posterior mean vector $\mu_{Y|D}$ (1) is used to predict the measurements at any set Y of unobserved road segments. The posterior covariance matrix $\Sigma_{YY|D}$ (2), which is independent of the measurements z_D, can be processed in two ways to quantify the uncertainty of these predictions: (a) the trace of $\Sigma_{YY|D}$ yields the sum of posterior variances $\Sigma_{ss|D}$ over all $s \in Y$; (b) the determinant of $\Sigma_{YY|D}$ is used in calculating the Gaussian posterior joint entropy

$$\mathbb{H}[Z_Y|Z_D] \triangleq \frac{1}{2}\log(2\pi e)^{|Y|}|\Sigma_{YY|D}| \ . \quad (3)$$

In contrast to the first measure of uncertainty that assumes conditional independence between measurements in the set Y of unobserved road segments, the entropy-based measure (3) accounts for their correlation, thereby not overestimating their uncertainty. Hence, we will focus on using the entropy-based measure of uncertainty in this paper.

2.1 Graph-Based Kernel

If the observations are noisy (i.e., by assuming additive independent identically distributed Gaussian noise with variance σ_n^2), then their prior covariance $\sigma_{ss'}$ can be expressed as

$$\sigma_{ss'} = k(s, s') + \sigma_n^2\delta_{ss'}$$

where $\delta_{ss'}$ is a Kronecker delta that is 1 if $s = s'$ and 0 otherwise, and k is a kernel function measuring the pairwise "similarity" of road segments. For a traffic phenomenon (e.g., road speeds), the correlation of measurements between pairs of road segments depends not only on their features (e.g., length, number of lanes, speed limit, direction) but also the road network topology. Therefore, the kernel function should be defined to exploit both the features and topology information, which will be described next.

Let the road network be represented by a weighted directed graph $G \triangleq (V, E, w)$ comprising a set V of vertices that denotes the domain of all possible road segments, a set $E \subseteq V \times V$ of directed edges such that there is a directed edge (s, s') from $s \in V$ to $s' \in V$ iff the end of segment s connects to the start of segment s' in the road network, and a weight function $w : E \to \mathbb{R}^+$ measuring the standardized Manhattan distance [1] of each directed edge:

$$w((s, s')) \triangleq \sum_{i=1}^{p} \frac{|[s]_i - [s']_i|}{r_i}$$

where $[s]_i$ ($[s']_i$) is the i-th component of the feature vector specifying road segment s (s'), and r_i is the range of the i-th feature. The weight function w serves as a dissimilarity measure between adjacent road segments.

The next step is to compute the shortest path distance $d(s, s')$ between all pairs of road segments $s, s' \in V$ (i.e., using Floyd-Warshall or Johnson's algorithm) with respect to the topology of the weighted directed graph G. Such a distance function is again a measure of dissimilarity, rather than one of similarity, as required by a kernel function. Furthermore, a valid GP kernel needs to be positive semidefinite and symmetric [27], which are clearly violated by d.

To construct a valid GP kernel from d, multi-dimensional scaling [1] is applied to embed the domain of road segments into the p'-dimensional Euclidean space $\mathbb{R}^{p'}$. Specifically, a mapping $g : V \to \mathbb{R}^{p'}$ is determined by minimizing the squared loss

$$g^* = \arg\min_g \sum_{s,s'\in V} \left(d(s, s') - \|g(s) - g(s')\|\right)^2 \ .$$

With a small squared loss, the Euclidean distance $\|g^*(s) - g^*(s')\|$ between $g^*(s)$ and $g^*(s')$ is expected to closely approximate the shortest path distance $d(s, s')$ between any pair of road segments s and s'. After embedding into the Euclidean space, a conventional kernel function such as the squared exponential one [25] can then be used:

$$k(s, s') = \sigma_s^2 \exp\left(-\frac{1}{2}\sum_{i=1}^{p'}\left(\frac{[g^*(s)]_i - [g^*(s')]_i}{\ell_i}\right)^2\right)$$

where $[g^*(s)]_i$ ($[g^*(s')]_i$) is the i-th component of the p'-dimensional vector $g^*(s)$ ($g^*(s')$), and the hyperparameters σ_s, $\ell_1, \ldots, \ell_{p'}$ are, respectively, signal variance and length-scales that can be learned using maximum likelihood estimation [25]. The resulting kernel function k[1] is guaranteed to be valid.

2.2 Sparse Approximation

Although the GP is an effective predictive model, it faces a practical limitation of cubic time complexity in the number $|D|$ of observations; this can be observed from computing the posterior distribution (i.e., (1) and (2)), which requires inverting the covariance matrix Σ_{DD} that incurs $\mathcal{O}(|D|^3)$ time. If $|D|$ is expected to be large, then GP prediction cannot be performed in real time. For practical usage, we have to resort to computationally cheaper approximate GP prediction.

[1]For spatiotemporal traffic modeling, the kernel function k can be extended to account for the temporal dimension.

A simple method of approximation is to select only a subset U of the entire set D of observed road segments (i.e., $U \subset D$) to compute the posterior distribution of the measurements at any set $Y \subseteq V \setminus D$ of unobserved road segments. Such a sparse *subset of data* (SoD) approximation method produces the following predictive Gaussian distribution, which closely resembles that of the full GP model (i.e., by simply replacing D in (1) and (2) with U):

$$\mu_{Y|U} = \mu_Y + \Sigma_{YU}\Sigma_{UU}^{-1}(z_U - \mu_U) \quad (4)$$

$$\Sigma_{YY|U} = \Sigma_{YY} - \Sigma_{YU}\Sigma_{UU}^{-1}\Sigma_{UY} \ . \quad (5)$$

Notice that the covariance matrix Σ_{UU} to be inverted only incurs $\mathcal{O}(|U|^3)$ time, which is independent of $|D|$.

The predictive performance of SoD approximation is sensitive to the selection of subset U. In practice, random subset selection often yields poor performance. This issue can be resolved by actively selecting an informative subset U in an iterative greedy manner: Firstly, U is initialized to be an empty set. Then, all road segments in $D \setminus U$ are scored based on a criterion that can be chosen from, for example, the works of [10, 11, 29]. The highest-scored segment is selected for inclusion into U and removed from D. This greedy selection procedure is iterated until U reaches a pre-defined size. Among the various criteria introduced earlier, the differential entropy score [11] is reported to perform well [19]; it is a monotonic function of the posterior variance $\Sigma_{ss|U}$ (5), thus resulting in the greedy selection of a segment $s \in D \setminus U$ with the largest variance in each iteration.

3. DECENTRALIZED DATA FUSION

In the previous section, two centralized data fusion approaches to exact (i.e., (1) and (2)) and approximate (i.e., (4) and (5)) GP prediction are introduced. In this section, we will discuss the decentralized data fusion component of our D^{2}FAS algorithm, which distributes the computational load among the mobile sensors to achieve efficient and scalable approximate GP prediction.

The intuition to our decentralized data fusion algorithm is as follows: each of the K mobile sensors constructs a local summary of the observations taken along its own path in the road network and communicates its local summary to every other sensor. Then, it assimilates the local summaries received from the other sensors into a globally consistent summary, which is exploited for predicting the traffic phenomenon as well as active sensing. This intuition will be formally realized and described in the paragraphs below.

While exploring the road network, each mobile sensor summarizes its local observations taken along its path based on a common support set $U \subset V$ known to all the other sensors. Its local summary is defined as follows:

Definition 1 (Local Summary). *Given a common support set $U \subset V$ known to all K mobile sensors, a set $D_k \subset V$ of observed road segments and a column vector z_{D_k} of corresponding measurements local to mobile sensor k, its local summary is defined as a tuple $(\dot{z}_U^k, \dot{\Sigma}_{UU}^k)$ where*

$$\dot{z}_U^k \triangleq \Sigma_{UD_k}\Sigma_{D_kD_k|U}^{-1}(z_{D_k} - \mu_{D_k}) \quad (6)$$

$$\dot{\Sigma}_{UU}^k \triangleq \Sigma_{UD_k}\Sigma_{D_kD_k|U}^{-1}\Sigma_{D_kU} \quad (7)$$

such that $\Sigma_{D_kD_k|U}$ is defined in a similar manner to (5).

Remark. Unlike SoD (Section 2.2), the support set U of road segments does not have to be observed since the local summary (i.e., (6) and (7)) is independent of the corresponding measurements z_U. So, U does not need to be a subset of $D = \bigcup_{k=1}^K D_k$. To select an informative support set U from the set V of all possible segments in the road network, an offline active selection procedure similar to that in the last paragraph of Section 2.2 can be performed just once prior to observing data to determine U. In contrast, SoD has to perform online active selection every time new road segments are being observed.

By communicating its local summary to every other sensor, each mobile sensor can then construct a globally consistent summary from the received local summaries:

Definition 2 (Global Summary). *Given a common support set $U \subset V$ known to all K mobile sensors and the local summary $(\dot{z}_U^k, \dot{\Sigma}_{UU}^k)$ of every mobile sensor $k = 1, \ldots, K$, the global summary is defined as a tuple $(\overline{z}_U, \overline{\Sigma}_{UU})$ where*

$$\overline{z}_U \triangleq \sum_{k=1}^K \dot{z}_U^k \quad (8)$$

$$\overline{\Sigma}_{UU} \triangleq \Sigma_{UU} + \sum_{k=1}^K \dot{\Sigma}_{UU}^k \ . \quad (9)$$

Remark. In this paper, we assume all-to-all communication between the K mobile sensors. Supposing this is not possible and each sensor can only communicate locally with its neighbors, the summation structure of the global summary (specifically, (8) and (9)) makes it amenable to be constructed using distributed consensus filters [20]. We omit these details since they are beyond the scope of this paper.

Finally, the global summary is exploited by each mobile sensor to compute a globally consistent predictive Gaussian distribution, as detailed in Theorem 1A below, as well as to perform decentralized active sensing (Section 4):

Theorem 1. *Let a common support set $U \subset V$ be known to all K mobile sensors.*

A. *Given the global summary $(\overline{z}_U, \overline{\Sigma}_{UU})$, each mobile sensor computes a globally consistent predictive Gaussian distribution $\mathcal{N}(\mu_Y^{\mathrm{D^2FAS}}, \Sigma_{YY}^{\mathrm{D^2FAS}})$ of the measurements at any set Y of unobserved road segments where*

$$\mu_Y^{\mathrm{D^2FAS}} \triangleq \mu_Y + \Sigma_{YU}\overline{\Sigma}_{UU}^{-1}\overline{z}_U \quad (10)$$

$$\Sigma_{YY}^{\mathrm{D^2FAS}} \triangleq \Sigma_{YY} - \Sigma_{YU}(\Sigma_{UU}^{-1} - \overline{\Sigma}_{UU}^{-1})\Sigma_{UY} \ . \quad (11)$$

B. *Let $\mathcal{N}(\mu_{Y|D}^{\mathrm{PITC}}, \Sigma_{YY|D}^{\mathrm{PITC}})$ be the predictive Gaussian distribution computed by the centralized partially independent training conditional (PITC) approximation of GP*

model [24] where

$$\mu^{\mathrm{PITC}}_{Y|D} \triangleq \mu_Y + \Gamma_{YD}\left(\Gamma_{DD}+\Lambda\right)^{-1}\left(z_D-\mu_D\right) \quad (12)$$

$$\Sigma^{\mathrm{PITC}}_{YY|D} \triangleq \Sigma_{YY} - \Gamma_{YD}\left(\Gamma_{DD}+\Lambda\right)^{-1}\Gamma_{DY} \quad (13)$$

such that

$$\Gamma_{AB} \triangleq \Sigma_{AU}\Sigma^{-1}_{UU}\Sigma_{UB} \quad (14)$$

and Λ is a block-diagonal matrix constructed from the K diagonal blocks of $\Sigma_{DD|U}$, each of which is a matrix $\Sigma_{D_kD_k|U}$ for $k=1,\ldots,K$ where $D=\bigcup_{k=1}^K D_k$. Then, $\mu^{\mathrm{D^2FAS}}_Y = \mu^{\mathrm{PITC}}_{Y|D}$ and $\Sigma^{\mathrm{D^2FAS}}_{YY} = \Sigma^{\mathrm{PITC}}_{YY|D}$.

The proof of Theorem 1B is given in Appendix A. The equivalence result of Theorem 1B bears two implications:

Remark 1. The computational load of the centralized PITC approximation of GP model can be distributed among K mobile sensors, thereby improving the time efficiency of prediction. Specifically, supposing $|Y| \leq |U|$ for simplicity, the $\mathcal{O}\big(|D|((|D|/K)^2+|U|^2)\big)$ time incurred by PITC can be reduced to $\mathcal{O}\big((|D|/K)^3+|U|^3+|U|^2K\big)$ time of running our decentralized algorithm on each of the K sensors, the latter of which scales better with increasing number $|D|$ of observations.

Remark 2. We can draw insights from PITC to elucidate an underlying property of our decentralized algorithm: It is assumed that $Z_{D_1},\ldots,Z_{D_K},Z_Y$ are conditionally independent given the measurements at the support set U of road segments. To potentially reduce the degree of violation of this assumption, an informative support set U is actively selected, as described earlier in this section. Furthermore, the experimental results on a real-world traffic phenomenon dataset[2] over an urban road network (Section 6) show that D^{2}FAS can achieve predictive performance comparable to that of the full GP model while enjoying computational gain over it, thus demonstrating the practicality of such an assumption for predicting traffic phenomena. The predictive performance of D^{2}FAS can be improved by increasing the size of U at the expense of greater time and communication overhead.

4. DECENTRALIZED ACTIVE SENSING

The problem of active sensing with K mobile sensors is formulated as follows: Given the set $D_k \subset V$ of observed road segments and the currently traversed road segment $s_k \in V$ of every mobile sensor $k=1,\ldots,K$, the mobile sensors have to select the most informative walks $w^*_1,\ldots,w^*_K$ of length L each and with respective origins $s_1,\ldots,s_K$ in the road network G:

$$(w^*_1,\ldots,w^*_K) = \arg\max_{(w_1,\ldots,w_K)} \mathbb{H}\Big[Z_{\bigcup_{k=1}^K Y_{w_k}} \Big| Z_{\bigcup_{k=1}^K D_k}\Big] \quad (15)$$

where Y_{w_k} denotes the set of unobserved road segments induced by the walk w_k. Interestingly, it can be shown using the chain rule for entropy that these maximum-entropy walks $w^*_1,\ldots,w^*_K$ minimize the posterior joint entropy (i.e., $\mathbb{H}[Z_{V\setminus\bigcup_{k=1}^K (D_k\bigcup Y_{w^*_k})}|Z_{\bigcup_{k=1}^K (D_k\bigcup Y_{w^*_k})}]$) of the measurements at the remaining unobserved segments (i.e., $V\setminus\bigcup_{k=1}^K(D_k\bigcup Y_{w^*_k})$) in the road network. After executing the walk w^*_k, each mobile sensor k observes the set $Y_{w^*_k}$ of road segments and updates its local information:

$$D_k \leftarrow D_k\bigcup Y_{w^*_k}\ , z_{D_k} \leftarrow z_{D_k\bigcup Y_{w^*_k}}, s_k \leftarrow \text{terminus of } w^*_k\ . \quad (16)$$

Without imposing any structural assumption, solving the active sensing problem (15) will be prohibitively expensive due to the space of possible joint walks $(w_1,\ldots,w_K)$ that grows exponentially in the number K of mobile sensors. To overcome this scalability issue, $Z_{Y_{w_1}},\ldots,Z_{Y_{w_K}}$ are assumed to be conditionally independent given the measurements at the set $D=\bigcup_{k=1}^K D_k$ of observed road segments. Such an assumption is not uncommon: it is often made in order to calculate the widely-used sum of posterior variances (i.e., mean-squared error) criterion (Section 2). In practice, this assumption usually becomes less restrictive when the number $|\mathcal{D}|$ of observed road segments increases to potentially reduce the degree of violation of conditional independence, the correlation of measurements between road segments decreases, and/or the mobile sensors are sufficiently far apart. Using the chain rule for entropy and subsequently the conditional independence assumption, the active sensing problem (15) reduces to

$$\begin{aligned}
&\max_{(w_1,\ldots,w_K)} \mathbb{H}\Big[Z_{\bigcup_{k=1}^K Y_{w_k}}\Big|Z_D\Big]\\
&= \max_{(w_1,\ldots,w_K)} \sum_{k=1}^K \mathbb{H}\Big[Z_{Y_{w_k}}\Big|Z_{\bigcup_{i=1}^{k-1} Y_{w_i}\bigcup D}\Big]\\
&= \max_{(w_1,\ldots,w_K)} \sum_{k=1}^K \mathbb{H}\Big[Z_{Y_{w_k}}\Big|Z_D\Big] = \sum_{k=1}^K \max_{w_k} \mathbb{H}\Big[Z_{Y_{w_k}}\Big|Z_D\Big]\ ,
\end{aligned}$$

which can be solved in a decentralized manner by each mobile sensor k:

$$w^*_k = \arg\max_{w_k} \mathbb{H}\Big[Z_{Y_{w_k}}\Big|Z_D\Big] = \arg\max_{w_k} \Big|\Sigma_{Y_{w_k}Y_{w_k}|D}\Big| \quad (17)$$

such that the second equality follows from (3) and the posterior covariance matrix $\Sigma_{Y_{w_k}Y_{w_k}|D}$ can be obtained using one of the data fusion methods described earlier, specifically, using (2) of full GP model (Section 2), (5) of SoD (Section 2.2), or (11) of D^{2}FAS (Section 3). If full GP or SoD is to be performed separately on each of the K mobile sensors rather than centrally, then the observations that are gathered distributedly by the sensors have to be fully communicated to every sensor. In contrast, D^{2}FAS only requires exchanging local summaries (Definition 1) between sensors.

Algorithm 1 below outlines the key operations of our D^{2}FAS algorithm to be run on each mobile sensor k, as detailed previously in Sections 3 and 4.

5. TIME AND COMMUNICATION OVERHEADS

In this section, the time and communication overheads of our D^{2}FAS algorithm are analyzed and compared to that of decentralized active sensing coupled with full GP (FGP) or SoD data fusion method to be run on each of the K sensors.

[2]The work of [24] only illustrated the predictive performance of PITC on a simulated toy example.

Algorithm 1: D^{2}FAS$(U, K, L, k, D_k, z_{D_k}, s_k)$

while *true* **do**
- `/* Data fusion (Section 3) */`
- Construct local summary by (6) and (7)
- Exchange local summary with every sensor $i \neq k$
- Construct global summary by (8) and (9)
- Predict measurements at unobserved road segments by (10) and (11)
- `/* Active Sensing (Section 4) */`
- Compute maximum-entropy walk w_k^* by (11) and (17)
- Execute walk w_k^* and observe its road segments $Y_{w_k^*}$
- Update local information D_k, z_{D_k}, and s_k by (16)

5.1 Time Complexity

Our D^{2}FAS algorithm comprises the data fusion and active sensing components. The data fusion component involves computing the local and global summaries and the predictive Gaussian distribution, as shown in Algorithm 1. To construct the local summary using (6) and (7), each sensor has to evaluate $\Sigma_{D_k D_k|U}$ in $\mathcal{O}(|U|^3 + |U|(|D|/K)^2)$ time and invert it in $\mathcal{O}((|D|/K)^3)$ time, after which the local summary is obtained in $\mathcal{O}(|U|^2|D|/K + |U|(|D|/K)^2)$ time. The global summary is computed in $\mathcal{O}(|U|^2 K)$ by (8) and (9). Finally, the predictive Gaussian distribution is derived in $\mathcal{O}(|U|^3 + |U||Y|^2)$ time using (10) and (11). Supposing $|Y| \leq |U|$ for simplicity, the time complexity of data fusion is then $\mathcal{O}((|D|/K)^3 + |U|^3 + |U|^2 K)$.

The active sensing component involves computing the maximum-entropy walk by (11) and (17). Let the maximum outdegree of G be denoted by Δ. Then, each mobile sensor k has to consider Δ^L possible walks. For each walk w_k, evaluating the determinant of $\Sigma^{\text{D}^2\text{FAS}}_{Y_{w_k} Y_{w_k}}$ incurs $\mathcal{O}(L|U|^2 + L^3)$ time. The time complexity of active sensing is therefore $\mathcal{O}(\Delta^L L(|U|^2 + L^2))$.

Hence, the time complexity of D^{2}FAS is $\mathcal{O}((|D|/K)^3 + |U|^3 + |U|^2 K + \Delta^L L(|U|^2 + L^2))$. In contrast, the time incurred by decentralized active sensing coupled with FGP and SoD are, respectively, $\mathcal{O}(|D|^3 + \Delta^L L(|D|^2 + L^2))$ and $\mathcal{O}(|U|^3|D| + \Delta^L L(|U|^2 + L^2))$. It can be observed that D^{2}FAS can potentially scale better with increasing number $|D|$ of observations when the number K of sensors is large. The scalability of D^{2}FAS vs. FGP and SoD will be further evaluated empirically in Section 6.

5.2 Communication Complexity

Let the communication overhead be defined as the size of each broadcast message. Recall from Algorithm 1 (i.e., D^{2}FAS) that, in each iteration, each sensor broadcasts a $\mathcal{O}(|U|^2)$-sized summary encapsulating its local observations, which is robust against communication failure. In contrast, FGP and SoD require each sensor to broadcast, in each iteration, a $\mathcal{O}(|D|/K)$-sized message comprising exactly its local observations to handle communication failure. If the number of local observations grows to be larger in size than a local summary of predefined size, then our D^{2}FAS algorithm is more scalable than FGP and SoD in terms of communication overhead.

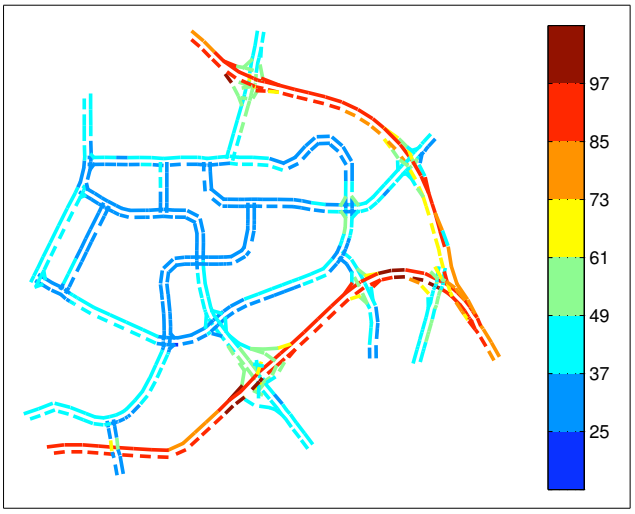


Figure 1: Traffic phenomenon (i.e., speeds (km/h) of road segments) over an urban road network in Tampines area, Singapore during evening peak hours on April 20, 2011. It comprises 775 road segments including highways, arterials, slip roads, etc. The mean speed is 48.8 km/h and the population standard deviation is 20.5 km/h.

6. EXPERIMENTS AND DISCUSSION

This section evaluates the predictive performance, time efficiency, and scalability of our D^{2}FAS algorithm on a real-world traffic phenomenon (i.e., speeds of road segments) dataset over an urban road network, as shown and detailed in Fig. 1. The performance of D^{2}FAS is compared to that of decentralized active sensing coupled with two state-of-art data fusion methods: full GP (FGP) and SoD (Section 2). A network of K mobile sensors is tasked to explore the road network to gather a total of up to 960 observations. To reduce computational time, each sensor repeatedly computes and executes maximum-entropy walks of length $L = 2$ (instead of computing a very long walk), unless otherwise stated. The size of the support set U is set to be 64. The experiments are run on a Linux PC platform with Intel® Core™2 Quad CPU Q9550 at 2.83 GHz.

6.1 Performance Metrics

The first metric evaluates the predictive performance of a tested algorithm: it measures the *root mean squared error* (RMSE)

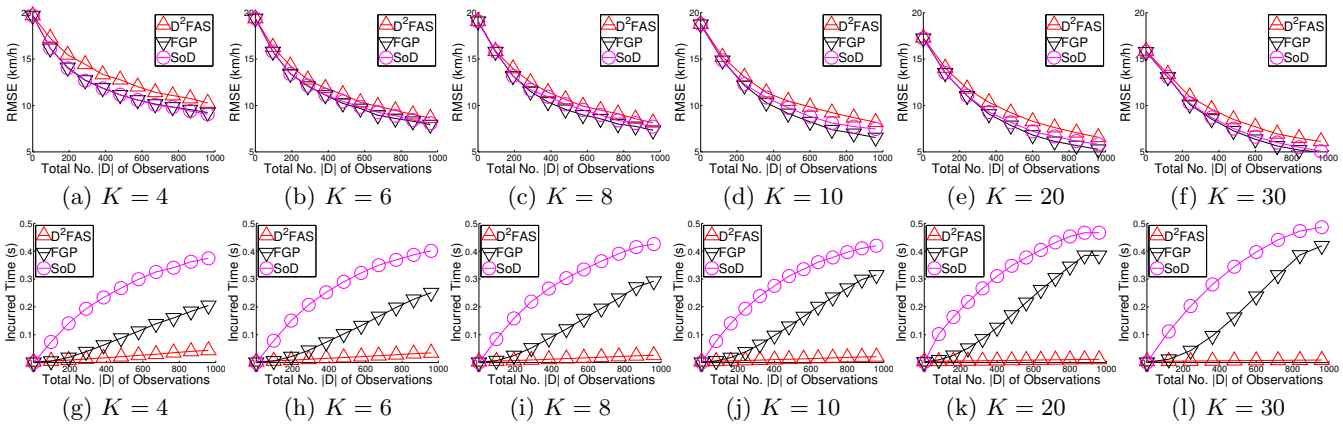
$$\sqrt{\frac{1}{|V|}\sum_{s \in V}(z_s - \widehat{\mu}_s)^2}$$

over the entire domain V of the road network that is incurred by the predictive mean $\widehat{\mu}_s$ of the tested algorithm, specifically, using (1) of FGP, (4) of SoD, or (10) of D^{2}FAS.

The second performance metric evaluates the time efficiency and scalability of a tested algorithm by measuring its incurred time.

6.2 Results and Analysis

Fig. 2 shows the results of the performance of the tested algorithms averaged over 40 randomly generated starting sensor



(a) $K = 4$ (b) $K = 6$ (c) $K = 8$ (d) $K = 10$ (e) $K = 20$ (f) $K = 30$

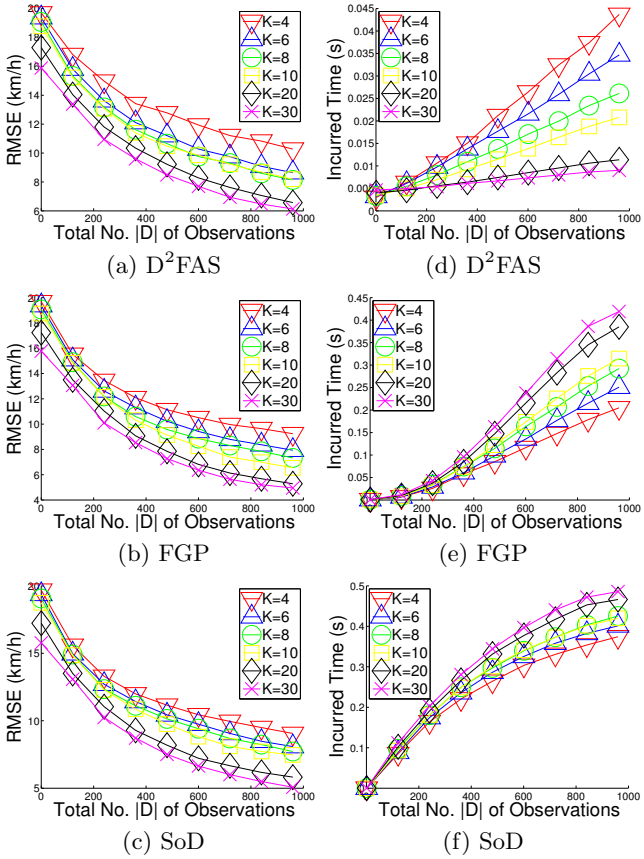
(g) $K = 4$ (h) $K = 6$ (i) $K = 8$ (j) $K = 10$ (k) $K = 20$ (l) $K = 30$

Figure 2: Graphs of (a-f) predictive performance and (g-l) time efficiency vs. total no. $|D|$ of observations gathered by varying number K of mobile sensors.

locations with varying number $K = 4, 6, 8, 10, 20, 30$ of sensors. It can be observed that D^{2}FAS is more time-efficient and scales significantly better with increasing number $|D|$ of observations (Figs. 2g to 2l) while achieving predictive performance close to that of FGP and SoD (Figs. 2a to 2f). Hence, the real-time performance and scalability (i.e., in the number of observations) of our D^{2}FAS algorithm enable it to be used for persistent large-scale traffic modeling and prediction where a large number of observations are expected to be available. The slightly better predictive performance of FGP and SoD are expected since they are able to exploit all collected observations for data fusion. In contrast, D^{2}FAS can only exploit local summaries over the small support set U. As mentioned earlier in Section 3, the predictive performance of D^{2}FAS can be improved by increasing the size of U at the expense of greater time and communication overhead.

Using the same results as that in Fig. 2, Fig. 3 plots them differently to reveal the scalability of the tested algorithms with increasing number K of mobile sensors. It can be observed from Figs. 3a to 3c that the predictive performance of all tested algorithms improve with a larger number of sensors because each sensor needs to execute fewer number of walks and its performance is therefore less adversely affected by its myopic selection (i.e., $L = 2$) of maximum-entropy walks. As a result, more informative unobserved road segments are explored. As shown in Fig. 3d, the time incurred by D^{2}FAS decreases due to its decentralized data fusion component that can distribute the computational load among a greater number of sensors. In contrast, it can be seen from Figs. 3e and 3f that the time incurred by FGP and SoD increase: as discussed above, a larger number of sensors result in a greater quantity of more informative unique observations to be gathered (i.e., fewer repeated observations), which increase the time needed for data fusion. When $K \geq 10$, D^{2}FAS is at least 1 order of magnitude faster than FGP and SoD. Hence, the scalability (i.e., in the number of sensors) of our D^{2}FAS algorithm allows the deployment of a large-scale mobile sensor network to achieve more accurate traffic modeling and prediction.

Fig. 4 shows the results of the performance of our D^{2}FAS algorithm with varying length $L = 2, 4, 6, 8, 10$ of maximum-

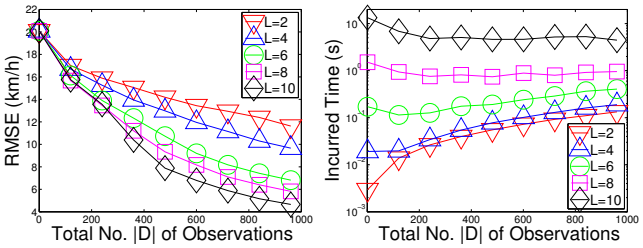


(a) D^{2}FAS (d) D^{2}FAS

(b) FGP (e) FGP

(c) SoD (f) SoD

Figure 3: Graphs of (a-c) predictive performance and (d-f) time efficiency vs. total no. $|D|$ of observations gathered by varying number K of sensors.



(a) (b)

Figure 4: Graphs of (a) predictive performance and (b) time efficiency vs. total no. $|D|$ of observations gathered by 2 mobile sensors running D^{2}FAS with varying length L of maximum-entropy walks.

entropy walks; we choose to experiment with just 2 sensors since Fig. 3d reveals that a smaller number of sensors produce poorer predictive performance and higher incurred time. It can be observed that the predictive performance improves with increasing walk length L because the selection of maximum-entropy walks is less myopic. When L increases to 10, the incurred time increases to about 10 seconds, which is reasonable in practice. By deploying a larger number of sensors, the incurred time is expected to decrease while improving the predictive performance.

7. CONCLUSION

This paper describes a decentralized data fusion and active sensing algorithm for modeling and predicting spatiotemporal traffic phenomena with mobile sensors. Analytical and empirical results have shown that our D^{2}FAS algorithm is extremely time-efficient and scales significantly better with increasing number of observations and sensors while achieving predictive performance close to that of state-of-the-art FGP and SoD. Hence, D^{2}FAS is practical for deployment in a large-scale mobile sensor network to achieve persistent and accurate traffic modeling and prediction. For our future work, we will assume that each sensor can only communicate locally with its neighbors (instead of assuming all-to-all communication between sensors) and develop a *distributed* data fusion approach to efficient and scalable approximate GP prediction based on our D^{2}FAS algorithm and consensus filters [20].

8. ACKNOWLEDGMENTS

This work was supported by Singapore-MIT Alliance Research and Technology (SMART) Subaward Agreement 14 R-252-000-466-592.

9. REFERENCES

[1] I. Borg and P. J. F. Groenen. *Modern Multidimensional Scaling: Theory and Applications*. Springer, NY, 2005.

[2] H. Chen, H. A. Rakha, and S. Sadek. Real-time freeway traffic state prediction: A particle filter approach. In *Proc. IEEE ITSC*, pages 626–631, 2011.

[3] T. H. Chung, V. Gupta, J. W. Burdick, and R. M. Murray. On a decentralized active sensing strategy using mobile sensor platforms in a network. In *Proc. CDC*, pages 1914–1919, 2004.

[4] M. Coates. Distributed particle filters for sensor networks. In *Proc. IPSN*, pages 99–107, 2004.

[5] J. Cortes. Distributed kriged Kalman filter for spatial estimation. *IEEE Trans. Automat. Contr.*, 54(12):2816–2827, 2009.

[6] J. M. Dolan, G. Podnar, S. Stancliff, K. H. Low, A. Elfes, J. Higinbotham, J. C. Hosler, T. A. Moisan, and J. Moisan. Cooperative aquatic sensing using the telesupervised adaptive ocean sensor fleet. In *Proc. SPIE Conference on Remote Sensing of the Ocean, Sea Ice, and Large Water Regions*, volume 7473, 2009.

[7] C. Guestrin, P. Bodik, R. Thibaus, M. Paskin, and S. Madden. Distributed regression: an efficient framework for modeling sensor network data. In *Proc. IPSN*, pages 1–10, 2004.

[8] Y. Kamarianakis and P. Prastacos. Forecasting traffic flow conditions in an urban network: Comparison of multivariate and univariate approaches. *Transport. Res. Rec.*, 1857:74–84, 2003.

[9] A. Krause, E. Horvitz, A. Kansal, and F. Zhao. Toward community sensing. In *Proc. IPSN*, pages 481–492, 2008.

[10] A. Krause, A. Singh, and C. Guestrin. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. *JMLR*, 9:235–284, 2008.

[11] N. Lawrence, M. Seeger, and R. Herbrich. Fast sparse Gaussian process methods: The informative vector machine. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 609–616, Cambridge, MA, 2003. MIT Press.

[12] K. H. Low, J. Chen, J. M. Dolan, S. Chien, and D. R. Thompson. Decentralized active robotic exploration and mapping for probabilistic field classification in environmental sensing. In *Proc. AAMAS*, 2012.

[13] K. H. Low, J. M. Dolan, and P. Khosla. Adaptive multi-robot wide-area exploration and mapping. In *Proc. AAMAS*, pages 23–30, 2008.

[14] K. H. Low, J. M. Dolan, and P. Khosla. Information-theoretic approach to efficient adaptive path planning for mobile robotic environmental sensing. In *Proc. ICAPS*, pages 233–240, 2009.

[15] K. H. Low, J. M. Dolan, and P. Khosla. Active Markov information-theoretic path planning for robotic environmental sensing. In *Proc. AAMAS*, pages 753–760, 2011.

[16] K. H. Low, G. J. Gordon, J. M. Dolan, and P. Khosla. Adaptive sampling for multi-robot wide-area exploration. In *Proc. IEEE ICRA*, pages 755–760, 2007.

[17] K. H. Low, G. Podnar, S. Stancliff, J. M. Dolan, and A. Elfes. Robot boats as a mobile aquatic sensor network. In *Proc. IPSN-09 Workshop on Sensor Networks for Earth and Space Science Applications*, 2009.

[18] W. Min and L. Wynter. Real-time road traffic prediction with spatio-temporal correlations. *Transport. Res. C-Emer.*, 19(4):606–616, 2011.

[19] S. Oh, Y. Xu, and J. Choi. Explorative navigation of mobile sensor networks using sparse Gaussian processes. In *Proc. CDC*, pages 3851–3856, dec. 2010.

[20] R. Olfati-Saber. Distributed Kalman filter with embedded consensus filters. In *Proc. CDC*, pages 8179–8184, 2005.

[21] P. Orbanz and Y. W. Teh. Bayesian nonparametric models. In C. Sammut and G. I. Webb, editors, *Encyclopedia of Machine Learning*, pages 81–89. Springer, NY, 2010.

[22] M. A. Paskin and C. Guestrin. Robust probabilistic inference in distributed systems. In *Proc. UAI*, pages 436–445, 2004.

[23] G. Podnar, J. M. Dolan, K. H. Low, and A. Elfes. Telesupervised remote surface water quality sensing. In *Proc. IEEE Aerospace Conference*, 2010.

[24] J. Quiñonero-Candela and C. E. Rasmussen. A unifying view of sparse approximate Gaussian process regression. *JMLR*, 6:1939–1959, 2005.

[25] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, 2006.

[26] M. Rosencrantz, G. Gordon, and S. Thrun. Decentralized sensor fusion with distributed particle filters. In *Proc. UAI*, pages 493–500, 2003.

[27] B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, 1 edition, 2002.

[28] D. Schrank, T. Lomax, and B. Eisele. *TTI's 2011 Urban Mobility Report*. Texas Transportation Institute, Texas A&M University, 2011.

[29] M. Seeger and C. Williams. Fast forward selection to speed up sparse Gaussian process regression. In C. M. Bishop and B. J. Frey, editors, *Proc. AISTATS*, 2003.

[30] K. K. Srinivasan and P. P. Jovanis. Determination of number of probe vehicle required for reliable travel time measurement in urban network. *Transport. Res. Rec.*, 1537:15–22, 1996.

[31] R. Stranders, A. Farinelli, A. Rogers, and N. R. Jennings. Decentralised coordination of mobile sensors using the max-sum algorithm. In *Proc. IJCAI*, pages 299–304, 2009.

[32] S. Sukkarieh, E. Nettleton, J. Kim, M. Ridley, A. Goktogan, and H. Durrant-Whyte. The ANSER project: Data fusion across multiple uninhabited air vehicles. *IJRR*, 22(7-8):505–539, 2003.

[33] S. M. Turner, W. L. Eisele, R. J. Benz, and D. J. Holdener. Travel time data collection handbook. Technical Report FHWA-PL-98-035, Federal Highway Administration, Office of Highway Information Management, Washington, DC, 1998.

[34] Y. Wang and M. Papageorgiou. Real-time freeway traffic state estimation based on extended Kalman filter: a general approach. *Transport. Res. B-Meth.*, 39(2):141–167, 2005.

[35] D. B. Work, S. Blandin, O. Tossavainen, B. Piccoli, and A. Bayen. A traffic model for velocity data assimilation. *AMRX*, 2010(1):1–35, 2010.

APPENDIX

A. PROOF OF THEOREM 1B

We need to first simplify the $\Gamma_{YD}\left(\Gamma_{DD}+\Lambda\right)^{-1}$ term in the expressions of $\mu_{Y|D}^{\text{PITC}}$ (12) and $\Sigma_{YY|D}^{\text{PITC}}$ (13).

$$\begin{aligned}
&\left(\Gamma_{DD}+\Lambda\right)^{-1}\\
&=\left(\Sigma_{DU}\Sigma_{UU}^{-1}\Sigma_{UD}+\Lambda\right)^{-1}\\
&=\Lambda^{-1}-\Lambda^{-1}\Sigma_{DU}\left(\Sigma_{UU}+\Sigma_{UD}\Lambda^{-1}\Sigma_{DU}\right)^{-1}\Sigma_{UD}\Lambda^{-1}\\
&=\Lambda^{-1}-\Lambda^{-1}\Sigma_{DU}\overline{\Sigma}_{UU}^{-1}\Sigma_{UD}\Lambda^{-1}\ .
\end{aligned}\tag{18}$$

The second equality follows from matrix inversion lemma. The last equality is due to

$$\begin{aligned}
&\Sigma_{UU}+\Sigma_{UD}\Lambda^{-1}\Sigma_{DU}\\
&=\Sigma_{UU}+\sum_{k=1}^{K}\Sigma_{UD_k}\Sigma_{D_kD_k|U}^{-1}\Sigma_{D_kU}\\
&=\Sigma_{UU}+\sum_{k=1}^{K}\dot{\Sigma}_{UU}^{k}=\overline{\Sigma}_{UU}\ .
\end{aligned}\tag{19}$$

Using (14) and (18),

$$\begin{aligned}
&\Gamma_{YD}\left(\Gamma_{DD}+\Lambda\right)^{-1}\\
&=\Sigma_{YU}\Sigma_{UU}^{-1}\Sigma_{UD}\left(\Lambda^{-1}-\Lambda^{-1}\Sigma_{DU}\overline{\Sigma}_{UU}^{-1}\Sigma_{UD}\Lambda^{-1}\right)\\
&=\Sigma_{YU}\Sigma_{UU}^{-1}\left(\overline{\Sigma}_{UU}-\Sigma_{UD}\Lambda^{-1}\Sigma_{DU}\right)\overline{\Sigma}_{UU}^{-1}\Sigma_{UD}\Lambda^{-1}\\
&=\Sigma_{YU}\overline{\Sigma}_{UU}^{-1}\Sigma_{UD}\Lambda^{-1}
\end{aligned}\tag{20}$$

The third equality is due to (19).

From (12),

$$\begin{aligned}
\mu_{Y|D}^{\text{PITC}} &= \mu_Y+\Gamma_{YD}\left(\Gamma_{DD}+\Lambda\right)^{-1}\left(z_D-\mu_D\right)\\
&= \mu_Y+\Sigma_{YU}\overline{\Sigma}_{UU}^{-1}\Sigma_{UD}\Lambda^{-1}\left(z_D-\mu_D\right)\\
&= \mu_Y+\Sigma_{YU}\overline{\Sigma}_{UU}^{-1}\overline{z}_U\\
&= \mu_Y^{\text{D}^2\text{FAS}}\ .
\end{aligned}$$

The second equality is due to (20). The third equality follows from

$$\begin{aligned}
\Sigma_{UD}\Lambda^{-1}\left(z_D-\mu_D\right) &= \sum_{k=1}^{K}\Sigma_{UD_k}\Sigma_{D_kD_k|U}^{-1}\left(z_{D_k}-\mu_{D_k}\right)\\
&= \sum_{k=1}^{K}\dot{z}_U^k=\overline{z}_U\ .
\end{aligned}$$

From (13),

$$\begin{aligned}
&\Sigma_{YY|D}^{\text{PITC}}\\
&=\Sigma_{YY}-\Gamma_{YD}\left(\Gamma_{DD}+\Lambda\right)^{-1}\Gamma_{DY}\\
&=\Sigma_{YY}-\Sigma_{YU}\overline{\Sigma}_{UU}^{-1}\Sigma_{UD}\Lambda^{-1}\Sigma_{DU}\Sigma_{UU}^{-1}\Sigma_{UY}\\
&=\Sigma_{YY}-\Big(\Sigma_{YU}\overline{\Sigma}_{UU}^{-1}\Sigma_{UD}\Lambda^{-1}\Sigma_{DU}\Sigma_{UU}^{-1}\Sigma_{UY}\\
&\quad-\Sigma_{YU}\Sigma_{UU}^{-1}\Sigma_{UY}\Big)-\Sigma_{YU}\Sigma_{UU}^{-1}\Sigma_{UY}\\
&=\Sigma_{YY}-\Sigma_{YU}\overline{\Sigma}_{UU}^{-1}\left(\Sigma_{UD}\Lambda^{-1}\Sigma_{DU}-\overline{\Sigma}_{UU}\right)\Sigma_{UU}^{-1}\Sigma_{UY}\\
&\quad-\Sigma_{YU}\Sigma_{UU}^{-1}\Sigma_{UY}\\
&=\Sigma_{YY}-\Big(\Sigma_{YU}\Sigma_{UU}^{-1}\Sigma_{UY}-\Sigma_{YU}\overline{\Sigma}_{UU}^{-1}\Sigma_{UY}\Big)\\
&=\Sigma_{YY}-\Sigma_{YU}\left(\Sigma_{UU}^{-1}-\overline{\Sigma}_{UU}^{-1}\right)\Sigma_{UY}\\
&=\Sigma_{YY}^{\text{D}^2\text{FAS}}\ .
\end{aligned}$$

The second equality follows from (14) and (20). The fifth equality is due to (19).

An Agent-based model to assess the impacts of introducing a shared-taxi system in Lisbon (Portugal)

Luis M. Martinez
Instituto Superior Técnico
Phone: (+351) 21 841 84 25
E-mail: martinez@civil.ist.utl.pt

Gonçalo Correia
University of Coimbra
Phone: (+351) 239 79 71 05
E-mail: gcorreia@dec.uc.pt

José M. Viegas
Instituto Superior Técnico
Phone: (+351) 21 841 84 13
E-mail: viegas@civil.ist.utl.pt

ABSTRACT

This paper presents a simulation procedure to assess the market potential for the implementation of a new shared taxi service in Lisbon (Portugal). The proposed shared taxi service has a new organisational design and pricing scheme which aims to use the capacity in traditional taxi services in a more efficient way. In this system a taxi acting in "sharing" mode offers lower prices to its clients, in exchange for them to accept sharing the vehicle with other persons who have compatible trips, (time and space) while also increasing the revenue for the operator.

The paper proposes and tests an agent based simulation model in which a set of rules for space and time matching between the shared taxis and passengers is identified considering a maximum deviation from the original route and then presents an algorithm that considers the following different objectives: minimum cost per passenger.km, maximum revenue per vehicle.km, minimum passenger in-vehicle time, minimum vehicle idle time.

An experiment for the city of Lisbon is presented with the objective of testing the proposed simulation conceptual model and to show the potential of sharing taxis for improving mobility management in urban areas.

General Terms: Algorithms; Design; Performance

Keywords: Agent-based models; shared taxi systems; ride matching

1. INTRODUCTION

The rising of automobile usage deriving from urban sprawl and car ownership growth is making traffic congestion more frequent and harder in urban areas. Moreover the majority of the trips are single occupant vehicle trips (SOV) resulting in more automobiles for the same persons. In 1990 approximately 90% of the work trips and 58% of the other trips in the United States were done in SOV [1]. Numbers of 1997 show that the occupation rate of the automobiles in commuting trips for the 15 Countries of the European Union was, at that time, in the interval between 1.1 and 1.2 persons per vehicle [2]. This results in air pollution, energy waste and unproductive and inefficient consumption of the time that persons have, and this does not show a tendency to slow down. In fact, traffic congestion and the corresponding environmental damage present a tendency to be aggravated.

This brings direct disadvantages for the users but also for the general economy and society at large. In 2001, the White Book on Transport Policy in the European Union stated that "if nothing is done, the cost of congestion will, on its own, account for 1 % of the EU's gross domestic product in 2010" [3], with a significant part of these costs respecting to urban transportation: traffic congestion associated to the automobile commuter trips. This is happening even in countries with high fuel prices, good Public Transport (PT) systems and dense land occupation [1].

PT cannot be the only alternative because providing transport capacity for peak periods would result in too many vehicles staying idle in non-peak periods, and too many people would be served with two or more transfers. Thus, there is the need to consider other alternatives, outside the classical transport modes. This is actually not a new idea. In the seventies, with the Arab Oil Crises, scientific interest arose for new transport alternatives, mainly in the United States. In fact it was in this decade that the first extensive research on this subject was published. In 1974 Ron Kirby and Kisten Bhat of the Urban Institute in Washington, U.S., released their report named: "Para-transit: Neglected Options for Urban Mobility" [4], this term, "Para-transit" was used as a general term to describe the various forms of flexible transportation that do not follow fixed routes or schedules such as shared taxis or carpooling.

Each one of these new modes has been studied and developed in the last decades, with several research projects and experiments being run and tested all over the world but the most advanced mainly in the USA and in Europe. They have been generally studied as isolated measures for controlling traffic congestion or for improving mobility options and, in some cases, they were able to have some (albeit rather limited) impact in reaching these objectives.

The shared taxi alternative denotes the use of common taxi-cabs by more than one person (or small party) serving multiple trips in the same taxi route [5]. This allows increasing the taxi operator's profit because costs should not vary significantly while there is the possibility of collecting a price from each passenger, even implementing a lower fare which should attract more passengers to this mode. Being a PT option but at the same time a low capacity mode, it is ideal for serving as a feeder system for other heavy transportation modes such as suburban trains [6].

However, there are not only advantages in using this system. In order for it to work there has to be people willing to share the vehicle with unknown passengers. In this case this should be softened by the presence of the taxi driver when compared to carpooling. Regarding trip time there may also be some discomfort for the extra riding time resulting from detours, this may or may not be compensated by lower transport costs and shorter waiting time for an available taxi.

All these questions make this an interesting mode for policy consideration, and for being modelled through simulation,

studying the effect of different operational parameters on the its market potential, mobility enhancement and transferring SOV trips to more efficient transport options.

In this paper we present such a simulation model developed under the principles of agent-based techniques. In the next section we review the existing shared taxi experiments followed by the system that we propose. The conceptual model is developed next presenting its main components, relationships and necessary input data and possible output indicators. In the following section an experiment is conducted using the conceptual model implemented in the simulation software AnyLogic (Xj Technologies). In this experiment we aim at proving the usefulness of the model by trying to answer the question as to how many less taxicab vehicles would be needed to attend current taxi demand if they were all functioning in shared mode. In the final section of the paper we end taking conclusions about the shared taxi mode and the simulation method.

2. THE SHARED TAXI EXPERIENCE

The idea of sharing taxis is not entirely new, both for economic reasons and for convenience there have been experiences in different countries of the world. However, the concept may vary greatly and is sometimes confused with other transportation alternatives as, for instance, vanpooling or mini-bus services. These are usually classified as paratransit transport services [7], a term which initially covered only unregulated services and is now extending to several offers being integrated in city transport networks.

These paratransit services usually operate under fixed routes, picking up passengers in pre-determined stops or at any point and leaving them in any place along a fixed route, charging a lower fare when compared to the regulated transport services. They found their share in places where supply was weaker. Not surprisingly it was in third world countries that these alternatives flourished, nourished by poor quality PT services and a great latent demand for travelling. For instance, while illegal, it is still normal in Korea to share a taxi with people having similar destinations [8].

Nevertheless these transport alternatives have also found their space in developed countries. One of the examples is carpooling, which has taken a very significant share of US commuters, is present even in Europe where PT systems are traditionally of a superior quality in service and comfort [9].

The most similar transport mode with the shared taxi systems has actually appeared very early in the 20th century in the USA and it had the curious designation of jitney. “During the economic downturn of 1914, some Los Angeles motorists down on their luck began giving rides at a nickel, or ‘jitney’, per trip and tended to shadow streetcar routes” [10].

The concept of collective taxi has been used for many years in Istanbul, Turkey, where it is a popular transportation alternative. There, it is called the dolmus which means to fill in Turkish. These cabs run a pre-determined route, with each passenger paying only a portion of the normal fare, making it a win-win situation where passengers pay less and drivers earn more money for the same distance. Passengers can get out anywhere along the route for a single set fare that is the same for all passengers no matter what their destination. Although their use is declining, dolmushes still operate within cities, and between cities and nearby towns and villages.

Despite the unregulated transport experiments with shared taxis or mini-bus and their progressively being included in the regulated services which always demands a certain level of standardisation of the operation, these systems objectives have not reduced their value along the years. There is still demand for intermediate modes between private transportation and high capacity PT vehicles such as buses and subway systems. That is why the shared taxi is being recovered as a modern transport option.

Advanced initiatives have been tested in order to take advantage of modern communication technology, namely cell phones, to help make viable the concept of shared taxi. A seed-stage company in the UK has developed a system that collates requests for point-to-point travel from a dispersed set of clients via SMS (they text-message by cell phone their destination postcode to the system), and then packages clients going in the same direction into one vehicle at a discounted fare. This is active now in four cities: London, Liverpool, Bournemouth and Isle of White. Passengers are instructed to go to pre-determined pickup points to meet the driver who will have received a text confirming each passenger’s booking reference [11].

In Brussels, Belgium, taxis are a regulated private sector undertaking, legally defined as door-to-door transport (strictly distinguished from limousine and car rental), with a proportional distance and time-based fare. Local authorities grant licenses, set price levels, supervise compliance with social legislation and define policy objectives. There is no possible confusion with the PT operators, whose core business is regular collective transport, based on fixed routes and timetables, integrated into bus, tramway and subway networks. Both are struggling against their polarised public images: whereas PT is upgrading to decrease its reputation as overcrowded, unreliable transit for the captive masses, taxis are striving to be seen as more than elite luxury transport [12]. In this city, authorities decided to implement a new night taxi service operated by a dispatcher and call-centre as a public service contract. The operator has been equipped with an optimisation system technology, and has upgraded the necessary number of cabs of affiliated taxi operators. No extra vehicles or drivers were put into circulation: ordinary taxis alternate between traditional taxi trips or shared taxi trips, as dispatched by the central. The operator provides a monthly listing of trips, their real cost (as registered by the taximeters) and fare revenue. The authorities then compensate for the difference, and the central distributes this sum among the taxi companies involved. For the taxi companies, each shared taxi trip is simply an extra trip, yielding full revenue.

From a modelling perspective, some incipient models have been recently developed to explore the shared taxi concept. Most of them use an optimisation [1316], simulation [17] or simulation-optimisation approach that support the ride matching algorithms, as well as a network operation model in dynamic models[1820]. Some of the simulation models have explored an Agent-based formulation [1720].

Overall it is noteworthy to verify that the previously referred studies have used simulation models and have pointed them as a good method to test the proposed dispatching strategies given the highly dynamic characteristics of the taxi services. Moreover, it is obviously impractical to deploy new taxi

directives immediately in the real world without carefully studying them, which can be done through a realistic computer simulated environment.

3. A NEW SHARED TAXI SYSTEM

As we have seen, when taxi shared services are successful they are so for two main reasons: short supply of traditional taxi services and other PT modes and/or allowing saving money in travel expenses. It is not surprising that the night period has come up as the best period for operating such transportation option: supply of PT is rather low during this period of the day, moreover there are many young people going out who often do not have a driving license, or want to drink beyond the legal limit for driving and whose only option is the taxi, an option which is usually expensive and that could be reduced through sharing the vehicle.

The system that we propose should be more comprehensive and not just an alternative for a night out, it should be a real option for any kind of trip at any period of the day within the boundaries of an urban area. Nevertheless the price must also play a strong role for sharing the taxi in such a way.

One should be reminded that the taxi is one of the best transport options that a person can have when convenience, comfort and safety are considered. A person is driven in a private vehicle which picks him up at the origin's door and drops him off at a precise destination point, without worries about parking the vehicle, and carrying a load whenever needed. Travel time maybe affected by traffic congestion during peak periods of the day but in many cities (as in Lisbon) less so than for a private car, as taxis are allowed to use Bus lanes. Moreover, as they are making a point to point trip, they can take detours recommended by GPS-based navigation systems, whereas when using traditional PT options the route is fixed.

The only problem remains to be the price of riding a cab. This varies from country to country, however it is never as low as other PT modes, hence it makes it a transportation option for higher income people or for those who do not own a private vehicle [21]. Sharing the taxi allows dividing the cost of the ride as already mentioned. However, the key question is: how is it possible to maintain the advantages of the taxi while sharing the vehicles? We have seen that most taxi sharing schemes are supported by pre-defined routes and/or have pre-located stops where people have to go, thus in practice the door to door advantage is lost.

The system which we propose makes use of current communication technology and GPS in order to bring flexibility to the system, managing virtually any possible origin and destination in an urban area. Trip requests are sent through a cell phone stating current position (or wished boarding point) and asking for a ride for a specific destination point. A central dispatcher collects this request and must then find a taxi match (this process is explained in the next section).

Central dispatching is already used as part of regular taxi services in order to improve customer demand compliance by computing in real time the closest taxi available [22]. However, the task of matching passengers and vehicles is obviously not straightforward as some of the cabs will already be transporting one or more passengers who have to be adequately served and reach their destination in acceptable time. The detours for picking and dropping-off other passengers may hinder many matches to be formed. This is not the case with the majority of the examples of current shared taxi practice where taxis stay practically in pre determined routes constrained by the existing stops.

4. THE SIMULATION FRAMEWORK

Every simulation experiment should start by a conceptual model which determines the relationship between its main elements and aims capturing the way the real system will function once it is implemented.

Because this is a simulation model of a system which will work in real-time, the simulation is based in a typical working day. The environment where the simulation takes place is a Road Network of the city where shared-taxi vehicles circulate and trips should be created according to census data or trip generation indicators. A Dispatcher will manage a centralised operation assigning taxis to clients using as his main information sources: the location of shared taxi vehicles, their current occupancy rate and the location of clients (assuming for simplification purposes that all passengers will want to be picked up at their current coordinates).

The simulation model for shared taxi services which we present is developed through agent-based simulation which is a class of computational models for simulating the actions and interactions of autonomous agents (either individual or collective entities such as organisations or groups) with the objective of assessing their effects on the system as a whole.

The models simulate the simultaneous operations and interactions of multiple agents, in an attempt to re-create and predict the appearance of complex phenomena. The process is one of emergence from the lower (micro) level of systems to a higher (macro) level. As such, a key notion is that simple behavioural rules generate complex behaviour.

This structure makes it clear how to program each element of the Agent-Based model for the shared taxi system and understand its possibilities. Using this classical structure one may begin by defining these elements for the two types of agent in the model: Taxis and Clients.

4.1 Client Agent

When a client decides to take a taxi, he first decides which type of service he will take: hail a taxi near their origin (where he may decide to go to a specific point of the network with greater probability of finding an available taxi); walk to a close taxi rank; or call a dispatching company. The selection of the action is randomly generated but with different probability profiles according to the city area and time of the day, trying to reproduce the knowledge that clients have. The possible states of this agent are then: searching for a taxi, waiting for an assigned taxi, or riding a taxi.

The general flowchart of the client agent is presented in Figure 1, where the different states, transitions and interaction are detailed. The decision process will be dependent of the type of taxi market selected by the client (e.g. hailing, taxi rank or phone request for a shared taxi service).

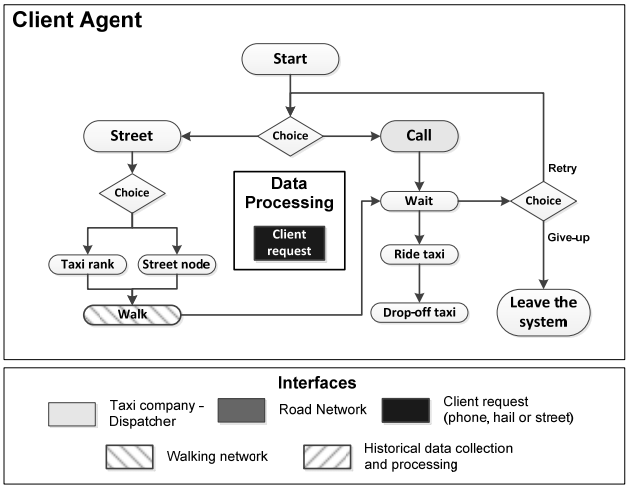


Figure 1. Simulation flowchart of the clients' agent

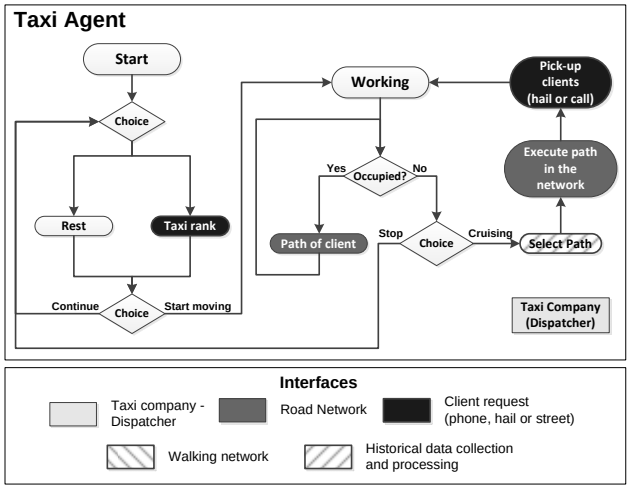


Figure 2. Simulation flowchart of the taxi agent

The rules for his behaviour are:

- Hail a taxi in the initial node or walk to a better hailing location;
- Walk to the "best" taxi rank within a walking threshold of his current location (using a trade-off function between the probability of finding a taxi and the willingness to walk);
- Dial to a dispatching service (randomly selected among the existing available options) to ask for a share taxi service;
- When the client goes to a road node or taxi rank, he waits for a taxi using a FIFO serving procedure;
  - If the client does not get a taxi after a threshold waiting time, he may re-evaluate (using a probabilistic approach) the decision of waiting or calling a dispatcher company to get a taxi;
  - After waiting up to a maximum of $\text{waiting}_{\max}$, the client leaves the system.
- When the client calls for a taxi and one is assigned to him, he automatically accepts that assignment;
- If a taxi is not assigned to him immediately, he waits for a given period (e.g. 1 minute) and places another taxi order, being the waiting time accounted since the first call for a taxi. After a maximum of three trials, the client considers selecting another dispatcher;
- After waiting more than the limit threshold ($\text{waiting}_{\max}$) without a taxi being assigned, he gives up from the service and goes out of the system.

4.2 Taxi Agent

A taxi can be connected to different taxi dispatching companies, being operated by a single driver (owner of the car) or belong to a taxi firm where several drivers work in shifts. The organisational model of supply is an input of the model. The possible states of this agent are then: being on route to pick-up a specific passenger (allocated by the Dispatcher); being on route; in service with passengers on board; being on route to a taxi rank; browsing the area for passengers; waiting at a taxi rank for an assignment; or being idle (taxi driver resting).

The general flowchart of the taxi agent is presented in Figure 2, where the different states, transitions and interaction are detailed.

The main rules of behaviour are:

- The taxi is normally heading to a taxi rank to wait for the next service. This next service may be picking up a client at that taxi rank or, if in the meanwhile the central dispatch assigns him a passenger, he will deviate from the current destination. When the taxi decides to stop at a taxi rank, it uses a route passing through areas where the probability of finding a customer is higher;
- The taxi not connected to a central dispatch system also routes through the network covering mainly the areas which historically have had a higher demand for taxi trips;
- The taxis located at a taxi rank give up waiting for a passenger if the service time in the taxi queue leads to a waiting time greater than a threshold. In this situation, the taxis either search for another taxi rank or route through the network (Monte Carlo generated);
- Taxis have shifts thus they are not always active. These are city and country specific and must be set because it determines the percentage of active taxis. If the taxi is connected to a central dispatch system, the company office's location will be selected as stop location; otherwise, the taxi will select randomly a node of the network to become inactive.

4.3 Simulation Environment

The environment where the agent based simulation takes place is a road network where taxi vehicles circulate and trips are created according to mobility survey data of the city. The road network contains link attributes, resulting in different travel times for different periods of the day. In each period, the network should accurately translate the impedance of travelling from point to point in the simulated urban area, reproducing the measured average congestion of road sections for the different periods of the day. Yet, the model presents a static non-equilibrium based traffic assignment procedure for the taxis, in a fixed traffic state, depending on the hour of the day. This simplification reduces considerably the computational burden of the model because it avoids the inclusion of other modes using the same road infrastructure (i.e. private cars and PT vehicles).

The model assumes that taxi drivers are experienced and that they are able to choose the shortest path for their destination, thus we use the Dijkstra's Algorithm, which computes in real

time the shortest (quickest) path between any given pair of nodes on the road network for a given time period during the day. We assume that the variation of the number of taxis in service in our simulation does not affect the predefined travelling speeds on the links of the network.

This changing environment is then used as interface for the different agents of the system, which interact through this platform and generate new data that changes its state variables. The different information linkages among agents and between agents and the environment can also be seen in Figure 1 and Figure 2.

The model presents five main types of interactions. A key element of interaction of the model is the taxi request, which can activate the three different types of taxi operational modes (rank, hail and call). Depending on the selected option by the user other types of interfaces are activated. If the clients chose to dial to a taxi company, a dispatcher service is activated to match the user and the active taxis. Otherwise, the client will connect to the taxi through the walking network: either by hailing a taxi or by walking to the most adequate taxi rank nearby. This demand data is then collected by the system to provide information to the taxi driver about the historical distribution, time and space dimensions, of the clients. This information is then used by taxi drivers to choose the most adequate taxi ranks to stop at different hours of the day. Furthermore, this information is also used to choose the most attractive routes for finding clients in the street. The last element of interaction between the agents and the environment is used in the hailing market, where the "vision" of the clients of a taxi that is approaching and of a taxi driver of a client request is modelled. This component considers the geometry of the road network (length of the road links and angles at intersections) assessing the maximum range of visibility at a certain location. Moreover, the probability of a taxi being able to stop and get the client is also a function of the estimated traffic flow of the street where the client is located. If arc is congested is more difficult for a taxi driver to switch to the right lane and stop for boarding. All these processes are parameterised in the model, considering fixed parameters for all clients and taxi instead of a statistical distribution with a specific value generated for each individual agent.

4.4 The Dispatcher

The Dispatcher was not conceived in the model as an Agent, but as an entity that defines a set of rules for matching together taxis and passengers, concentrating all real-time information required to produce and monitor these trips.

The choice of which taxis to match with each client follows a linear programming optimisation model. The problem was formulated with an objective function that aims to combine the minimisation of passenger travel time (the one(s) riding and the one requesting a taxi), while also considering the revenues of each individual taxi and the equity among them (always a strong concern in the real world).

There are several ride-matching optimisation algorithms formulated in the literature [23]. Yet, most of these models formulate a simultaneous matching between several drivers, going to their destinations, and several ride requests, aiming to achieve a system optimum considering all demand and supply gathered in a time interval. This is especially important when there is great density of clients and drivers, but in our case we believe that the simplicity of only considering one client at a time makes the simulation much faster and the solution will not be significantly far from the optimum due to the low density of requests in the city (distance between callers is too high for there to be any true competition for the same taxi).

We should note however that it would be easy to integrate in the simulation model any kind of algorithm to match passengers and clients, which is one of the advantages of simulation techniques.

The current formulation reduces significantly the complexity of this problem by evaluating the best ride matching alternative for a single client, following a request order. The smaller size of this problem allows searching exhaustively for an optimal solution within a small to medium size domain of solutions in a very short simulation time.

The strategy for selecting taxis is shown in Figure 3, in it we may see multiple taxis available within a coverage area centred in the client's coordinates. We also describe the coefficients used to build the constraints of the problem.

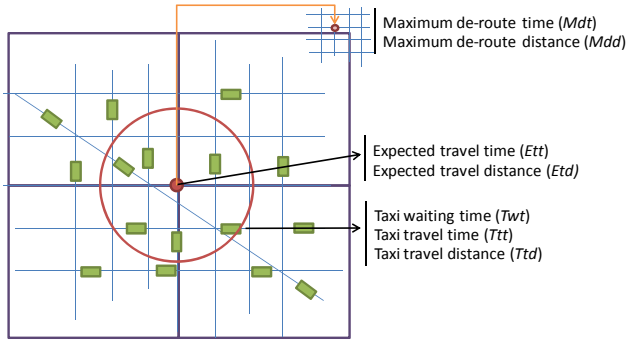


Figure 3. The taxi-client matching problem

In order to solve this combinatorial problem, we started by defining the maximum de-route time (*Mdt*) and de-route distance (*Mdd*) that the passenger is willing to accept for the current trip. These parameters of the simulation were initially set by the authors as percentage of *Ett* and *Etd* values respectively, function of travelled time and distance.

Then, for each client *i*, the dispatcher's computer specifies: The expected travel time (*Ett*) and travel distance (*Etd*) for the given origin and destination of the passenger, computed by the shortest path algorithm for the current time period of the day (Dijskstra's Algorithm included in the agent-based model).

It also computes for each taxi *j* and each client *i* at time instant *t*, the following variables: the waiting time for the taxi (*Twt*), the taxi travel time (*Ttt*) and travel distance (*Ttd*). This travel time takes into consideration the minimum sum of disturbance time for each passenger on board that would be introduced to the current riders and to the new client. This time is also computed using a combinatorial problem which can be expressed by:

$$Ttt_{Taxi\,P} = tt_{Taxi\,P} + min\left\{ tt_{Pi} + \sum_{j=i,k=j+1}^{Clients} tt_{jk} \right\}$$

Where $Ttt_{Taxi\,P}$ is the travelling time between the current position of the taxi and pick-up point of client *P*, tt_{Pi} the travelling time between the pick-up point of client *P* and the drop-off point of client *i*, and tt_{jk} the travelling time between the drop-off point of client *j* and the drop-off point of client *k*.

The estimation of the taxi travel time (*Ttt*) is done using the procedure presented in Figure 4, where we may see the different approaches depending on the number of passengers already on-board of the taxi.

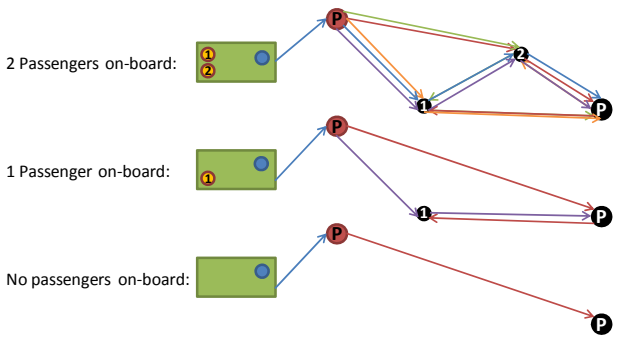


Figure 4. Example of the Taxi travel time (*Ttt*) estimation for different number of passengers on-board

The model contains the information on which road network arc the taxi and the passenger are currently positioned. It also collects the code of the zone in which the passenger is contained as well as the codes of neighbouring zones (vector *Nz*).

Then the problem is to select the taxis which are within a certain distance (e.g. 2 km) of the client's position scanning also their neighbouring zones (*Nz*) which comply with the client's constraints to travel time and distance acceptance (*Mdt* and *Mdd*). The mathematical formulation of the problem is the following:

$$\min_{i \in Taxis \subseteq Nz \subseteq R \subseteq 2km} \{Twt_{ij} + Ttt_{ij} + 1000 \cdot Empty_i - 2500 \cdot EB_i - 3000 \cdot 1Pass_i + 1500 \cdot 2Pass_i\}$$

Subject to:

$$\forall j \epsilon \, \mathrm{i} \therefore \mathrm{Ttt_j} \leq \mathrm{Ett_i} \cdot (1 + \mathrm{Mdt_j})$$

$$\forall j \epsilon \, \mathrm{i} \therefore \mathrm{Ttd_j} \leq \mathrm{Etd_i} \cdot (1 + \mathrm{Mdd_j})$$

Where Twt_{ij} is the waiting time of client *j* to be picked-up by taxi *i*; $Empty_i$ is a binary variable which takes the value 1 if taxi *i* is empty; EB_i is a binary variable which takes the value 1 if taxi *i* has been without passengers for the last 5 minutes; $1Pass_i$ is a binary variable that takes the value 1 if taxi *i* has one client already on-board; finally $2Pass_i$ is also a binary variable that takes value 1 if the taxi *i* has already two clients on-board.

The objective function, while minimising the client travel time, also assigns preferentially clients to taxis which have been empty during the last five minutes and also to taxis with two clients already on-board, presenting the same premium as the previous (weights in the objective function), and especially to taxis that have one client already on-board, which lead to greater taxi revenues and maximum discounts to the clients.

This optimisation procedure, while not corresponding to a NP-Complete problem, also presents increasing computing times with the problem dimension, which has been addressed in the simulation by reducing the subset of candidate taxis in each optimisation procedure. The considered subset includes 50% of the total taxi fleet contained by the relevant zones (vector *Nz*) or a minimum number of candidate 50 taxis.

This method allows a considerable reduction of computational time in large scale simulations (typically all the trips in an urban area), by reducing the number of times the shortest path algorithm has to be applied for the estimation of the objective function, especially during the peak hours when the average frequency of requests is considerably increased.

Another important algorithm which is used dynamically during the simulation is the estimation of taxi densities along the road network for the different zones, and the estimation of this indicator deviation relative to historical data. This information is used to determine the most suitable destinations in the network for each taxi that is going to route for passengers at a given simulation period *t*.

The Dispatcher gathers information about passenger requests from previous days at the same hour of the day and joins this information to the historical data in order to estimate the predicted concentration of taxi passengers during the next hours. At each time period, the Dispatcher measures the deviation of taxis available for clients calls (empty or with available capacity) in each zone of the city relative to its estimation of what would be required and distributes recommendations for direction of browsing based in the utilities of the different zones. Zone *i* utility function for time period *t* is given by:

$$ZU_{ti} = \frac{ED_{ti}}{\sum_{j=1}^{N} ED_{tj}}$$

$$ED_{ti} = 0.75 \cdot \sum_{k=1}^{N} D_{kti} + 0.25 \cdot SH \cdot \sum_{d=1}^{N} OD_{tid}$$

Where ED_{ti} is the estimated taxi demand of zone *i* for the time period *t* (an hour), D_{kti} are the collected taxi calls for zone *i* for the time period *t* in the *k* day, SH is the estimated taxi share for the study area, and OD_{tid} is the total number of trips in the study area for the period *t* that were originated in zone *i*.

The obtained utilities are then converted to probabilities of selecting each zone, and for each taxi order, the model generates a random number and assigns a destination zone. The final destination road network node is obtained by a random generation procedure among the nodes contained by the selected zone.

5. LISBON CASE-STUDY

The initial test bed of this new simulation procedure was the municipality of Lisbon, Portugal. Lisbon is the Capital city of Portugal and is the largest city of the country with approximately 565,000 inhabitants in an area of 84.6 km^2. Lisbon is situated on the Atlantic Ocean coast on the Tagus estuary, being the most western capital in mainland Europe. Lisbon is the centre of the Lisbon Metropolitan Area (LMA), which has approximately 2.8 million inhabitants, representing roughly 25% of Portugal population, with an area of 2,962.6 km^2, formed by other 18 municipalities.

The taxi market in Lisbon is formed by approximately 3,500 taxis, which have to apply and pay a municipal license. The number of available licenses is capped, and has not increased in recent years, which led to a significant enhancement of its (unofficial) value. Taxis have to apply and pay a municipal license [24]. The number of available licenses is capped, and has not increased in recent years, which led to a significant increase of its (unofficial) value. These licenses cannot be traded directly on the market, still companies are the owners of the licenses and

companies are tradable which indirectly leads to a license market.

This license allows taxis to operate simultaneously in three market types regarding the way clients access the service: rank market, hail market and pre-booked market:

- Rank places are designated places where a taxi can wait for passengers and vice versa. Taxis and customers form queues regulated by a FIFO system. Disadvantages are that due to the FIFO policy established, price has no effect on customer choice of which taxi to take.
- In the hail market, clients hail a cruising taxi on the street. There is uncertainty about the waiting time and the quality of the service customers will find. The advantage here is that the customer does not have to walk to a taxi rank.
- In the pre-booked market, consumers telephone a dispatching centre asking for an immediate taxi service or for a later taxi service. Only in this kind of market consumers can choose between different service providers or companies. At the same time, companies can get clients' loyalty providing a good door to door service.

The three markets described are active in Lisbon and taxis may operate in them at the same time. Some taxi drivers are associated to a taxi phone dispatching company paying a fee to have access to that pool of clients. The client also has to pay the phone call when he wants to access that service. A recent study performed by Mobility and Transport Institute (IMT) showed that only approximately 48% of the taxis are associated with a dispatching company, being the other 52% restricted to the hailing and taxi rank market [24].

These three service configurations have different market expressions across the world, although they are almost all the times present in the taxi market at the same time. In New York, for instance, most of the passengers hail the taxi on the street (90%) while in Stockholm: 55% call the taxi by phone, 20% by going to a taxi rank and only 25% hail the taxi on the street [21].

The fact that in Lisbon taxis may operate in the three markets simplifies significantly the regulation of the market. In parallel, the taxi drivers' profession is also regulated by the national transport regulator (Mobility and Transport Institute – IMT). The taxis can be driven by licensed drivers, which have do take a course and pay a levy. IMT has surveyed recently taxis, and inspected the shifts of taxi drivers. The results showed that from the 3,500 taxis registered in Lisbon, only 3,100 taxis, in average, are active daily. The survey did also identify five main types of taxis drivers' shifts, which mainly depend on the ownership of the taxi (owned by the driver or by a taxi company). The resulting types of shifts of the taxi drivers in the city can be seen in Table 1.

Taxi fares are also strictly regulated by specific legislation, which set the price of the trip by three different components: a fixed starting fee, a distance related fee and a time related fee, linked to the delay time produced by congestion, set for time that is travelled for speeds under 30km/h.

In order to simulate the behaviour of the taxi market within the city of Lisbon, we gathered a large set of data required for the simulation. This data encompasses the estimation of the taxi travel demand in the city, including:

- the origin and destination of the taxi trips as well as their starting time;
- the road network;
- a calibrated traffic assignment model to obtain travel times in the road network;
- the taxi ranks location; and
- a zoning system, which was used to compute taxi concentrations along the city and help taxi drivers to decide where to go at any time during the day.

Table 1.Taxi driver shift considered in the simulation

Shift	1st driver shift	2nd driver shift
Type 1	6 am to 7 pm (idle from 12 pm to 1 pm)	
Type 2	8 am until 9 pm (idle from 2 pm to 3 pm)	
Type 3	1pm to 2 am (idle from 7 pm to 8 pm)	
Type 4	7 am until 6:30 pm (idle from 1 pm to 2 pm)	6:40 pm to 5:40 am (idle from 0:40 am to 1:10 am)
Type 5	9 am until 8:30 pm (idle from 3 pm to 4 pm)	20:40 pm to 9 am (idle from 2:40 am to 3:10 pm)

The simulation procedure uses as input the results of a synthetic travel simulation model, which was developed under the SCUSSE research project [25]. This model is based on a mobility survey of the LMA performed in 1994, with approximately 60,000 trips and 23,000 persons surveyed, and an activity database of 2009 that was used to update the travel patterns observed in the initial survey. This is a rule based model, which uses the reported travels by respondents and their connections along the day, to disaggregate a total population of trips of the LMA based on the 2009 activity generation (trip generation coefficients for different activities along the day) and transport network, generating specific origin and destination points, transport mode used and starting time of each trip.

The synthetic travel model generated 21,075 taxi trips during a week day inside the city of Lisbon. The distribution of these taxi trips along the day is presented in Figure 5, where we may observe a higher concentration of trips during the morning peak and some periods during the lunch break and the afternoon.

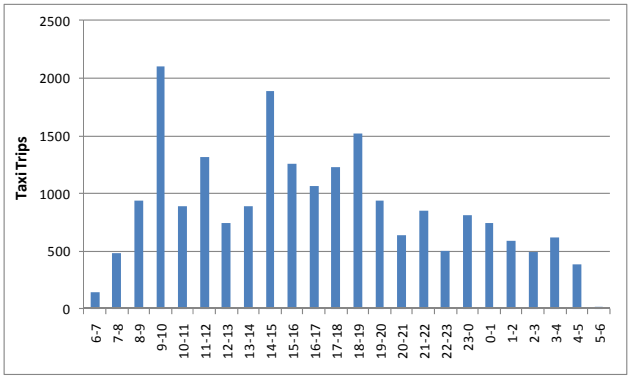


Figure 5. Distribution of taxi trips throughout a working day

We have to acknowledge, that the number of estimated taxi trips is considerably lower than the real demand, which should include trips from Lisbon to other municipalities (additional 3,435 according to the model), and non residents of the LMA as tourists and other visitors (e.g. professionals from other parts of the country), not represented in the survey sample. Furthermore, normally transport modes with lower shares tend to be misrepresented in a survey due to random sampling procedures.

All these facts may affect considerably the real representation of taxi trips in the municipality of Lisbon. Yet, the purpose of the paper is not to fully represent reality, but to show the proof of concept in using this simulation procedure to model an intermediate alternative transport mode. A full representation of demand is going to be used in the next stages of the research, namely through a survey of the taxi drivers and their businesses.

The simulated trips are randomly assigned to one of the network nodes within 200 meters away of the origin or destination points. The shared taxi passengers are then picked up and dropped off, in these nodes, for simplicity purposes.

The model was implemented in a road network model of the Lisbon municipality formed by the first four levels of the road hierarchy, comprising urban motorways, ring-roads, major arterials and the main local distribution network. This network contains 11,242 links and 7,106 nodes.

For determining the travel times of all links and intersections of the road network along the day, we used a calibrated micro-simulation traffic assignment model (AIMSUN - TSS) for the morning peak hour (8 to 9 o´clock). This model was calibrated using a Mobility Survey from 2004 used to develop the Lisbon Mobility Plan, and a zoning system of 66 TAZs.

The travel times for each link and intersection during the different hours of the day were estimated using the existing percentages of trips generated during the day. In Figure 6 we may see the percentage of private car trips which affect the travel time in the network.

The travel time of each time interval is then computed using the following equation:

$$Load\ Factor_i = \frac{Percentage\ trips_i}{Percentage\ trips_{8-9}}$$

Where the load factor of time interval *i* results from dividing the estimated percentage of trips in time interval *i* and the percentage of trips between 8 and 9 am. Thus the travel time (*TT*) of each link is given by:

$$TT_{ji} = TT_{j0} \cdot \left[1 + 2 \cdot \frac{Load\ Factor_i \cdot load_j}{capacity_j}\right]$$

Where TT_{ji} is the travel time of link *j* in the travel time interval *i*; TT_{j0} the free flow travel time of link *j*; $load_j$ the traffic load of link *j*; and $capacity_j$ the capacity of link *j*. This value delay function is available in the Highway Capacity Manual [26], being used with the parameter $\alpha = 2$ and $\beta = 3$.

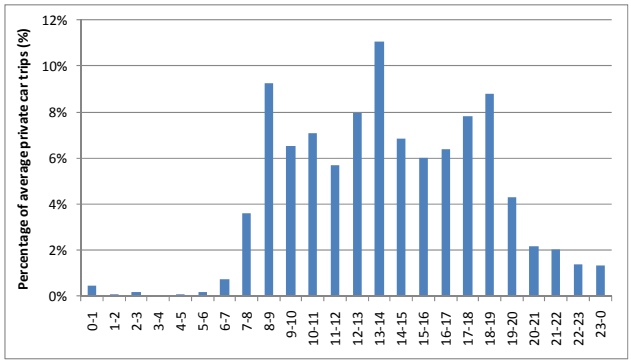


Figure 6. Distribution of the percentage of average private car trips throughout a working day

The travel time lost in each intersection of the road network was computed using a similar approach. The $Load\ Factor_i$ is once again used as a correction factor from a base value of the reference interval between 8 and 9 am. The value for node *j* and time interval *i* is given by the equation:

$$NT_{ji} = NT_{j0} \cdot \left[\frac{1}{1 + e^{2.4795 - 6.7378 \cdot Load\ Factor_i}}\right]$$

Without an available source of a generic delay function in an intersection related with the traffic volume, this equation was obtained by the calibration of an inverse logistic curve that was initially used to measure accessibility [27]. The general equation is given by:

$$y = 11 + e^{a - b \cdot x}$$

Where *a* and *b* are parameters that require calibration for the specific application. A calibration of this equation was done taken into account that values of the Load Factor do not present significant reductions on the intersection impedance (0.70 load factor leads to a corrections factor of 0.90), and that low congestion situations lead to a significant reduction of the time lost in an urban intersection (0.05 load factor leads to a corrections factor of 0.1).

The model also includes the location of all the taxi ranks in the city of Lisbon (82 taxi ranks), where the taxis can be idle or wait for a passenger call. All the agents and objects of the simulation were aggregated into a zoning system formed by 115 different zones. This zoning system was obtained using a zoning optimisation procedure for the city of Lisbon, using the 2004 Mobility Survey data [28]. This spatial discretisation considerably reduces the complexity of the model by collecting information of all taxis available and occupied within each area of the city, and simultaneously, retrieving information to the taxis about the most willing spots to find passenger.

6. TESTING A SHARED TAXI SYSTEM

The simulation model experiment developed for this paper consists on a performance comparison of the current regular taxi system in the city of Lisbon, and the new shared taxi system discussed in this paper. The experiment considers a static taxi demand to the new market configuration that might occur from introducing shared taxis allowing measuring the expected reduction of waiting time and fare paid by the customers. In the present paper we do not consider demand elasticity to price or waiting time.

This static formulation represents a first step on the assessment of the potential impact of the implementation of the service, focusing on a users' perspective which may lead to future induced demand. Thus, this assessment compares output operation indicators for the current taxi fleet with a mixed fleet of conventional and shared taxis.

Furthermore, the willingness of passenger to dial for a taxi service was not altered from the reference scenario, which in reality could be altered if the passengers expect a better service from the shared system when compared to the fee paid to dial for a taxi service.

For this simulation experiment the total taxi fleet of Lisbon was considered to be 2,000 taxis instead of the real 3,100 taxis that operate daily in Lisbon, this is due to the demand underestimation on the available data discussed above, which would considerably bias the performed analysis. The use of

approximately 2/3 of the fleet derives from an experienced guess from the authors based on the knowledge obtained from the mobility survey, however this lacks from empirical verification. We will consider in this test that all the current taxi fleet connected to a central taxi dispatcher company would switch automatically to the shared taxi market (approx. 48%).

Different total taxi fleet sizes were tested in order to ensure the consistency of the results, and assess the impact of the shared system configuration under a more saturated taxi market.

The taxi discount scheme tested in the experiment was the following:

- Riding a shared taxi alone has a 15% discount;
- Sharing a taxi with another client has a 40% discount to each client;
- Sharing a taxi with two other clients has a 55% discount to each client;

The fare paid by each client results from the sum of the different stretches of the trip with different occupancy rates of the taxi, which present different discounts. The simulation will measure the discount obtained for each client of a shared taxi relative to the reference price of riding alone, thus allowing estimating the client's savings introduced by the new system.

Figure 7 presents a screenshot of an area of the city during the simulation, where we may observe the taxis (represented as larger circles) and the clients' states (represented as small circles).

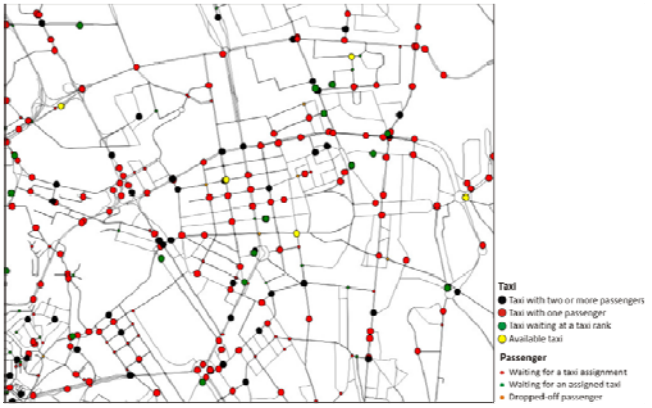


Figure 7. Screenshot of the Agent-Based simulation method working in the Anylogic simulation environment of Lisbon

In order to compare the resulting outputs of the model, we developed a set of indicators to measure the performance of the system compared to the base scenario of a fleet without shared taxis.

The obtained indicators for the base scenario, with a fleet of 2,000 taxis, were able to reproduce considerably well the main aggregate indicators of the system performance from the supply side as the average taxi revenue (79.76 euros per day against the measured 79.57 euros per day), and the average number of travelled km (14.24 against the 15.98 services obtained from real data).

The obtained results are presented in Table 2, showing that the shared system may lead to a significant reduction in the average passengers' waiting time and also the average taxi system fare, which present savings for shared riders close to 20 percent. Furthermore, the taxis may also benefit from an increase in operational efficiency, measured by the average revenue per travelled km, showing that the taxis enhance the estimated value for this indicator, although not observing a monotonous trend as in the other indicators.

Table 2.Performance indicators of the shared taxi system for different fleet sizes

Fleet size (48% shared)	Av. pass. waiting time (min)	Av. taxi fare [€]	Av. savings of shared riders (%)	Av. total travel time [min]	Av. revenue per taxi km [€/km]
1400	12.94 (-28.68%)	7.43 (-5.47%)	-19.02%	26.6 (-15.93%)	0.46 (5.78%)
1600	12.17 (-19.57%)	7.45 (-4.82%)	-17.92%	25.55 (-8.25%)	0.44 (4.82%)
1800	11.95 (-15.42%)	7.37 (-5.78%)	-15.56%	24.95 (-6.15%)	0.42 (7.38%)
2000	11.72 (-10.00%)	7.45 (-4.60%)	-15.01%	24.66 (-2.76%)	0.38 (1.94%)

The results show that the shared configuration may lead to considerable changes in the system performance from a users' perspective. This change, considering a static demand to the fare and waiting time reduction, leads to a reduction of the taxi system revenue derived from the offered discount. The average reduction in revenues for taxi drivers is approximately 10% for the chosen taxi fleets, which has to be compensated by a similar demand increase if the shared system is to produce a win-win situation for the clients and the taxi drivers.

7. CONCLUSIONS AND FUTURE WORK

This paper sets an innovative simulation procedure to assess the market potential of an advanced dynamic shared taxi service. This model was developed using agent-based simulation taking the advantage of modelling taxis and clients as agents who take decisions which are specific to their interests. At the same time an entity that manages the assignment between these two types of agents was identified and programmed to act in both the interest of the passenger and taxi in order to improve the level of service offered by taxis while still improving this business overall profit.

This new procedure was implemented in a large scale example: the municipality of Lisbon that counts about 3,500 taxi vehicles, from which 3,100 operate daily. This example allowed comparing different taxi fleet compositions, varying from the current fleet, where all taxis serve just one trip, to new scenarios where different taxi percentages acting in a sharing mode are introduced replacing the traditional ones.

Further developments of this research will include a thorough characterisation of the taxi market behaviour and also the assessment of the impact on the demand for taxi travel and operator revenue introduced by offering the shared taxi system.

8. ACKNOWLEDGMENTS

This research is being supported by the Portuguese National Science Foundation (FCT) under the SCUSSE Project - MIT Portugal Program.

9. REFERENCES

[1] Shaheen, S. A., Sperling, D. and Wagner, C. 1999. Carsharing and Partnership Management An International

Perspective. Carsharing and Partnership Management An International Perspective, 1666, 118-124.
[2] 1997. Indicators of energy use and efficiency : understanding the link between energy and human activity. ParisParis: OECD/IEA;
[3] 2001. WHITE PAPER - European transport policy for 2010: time to decide, European Commission.
[4] Kirby, R. and Bhat, K. 1974. Para-transit: Neglected Options for Urban Mobility. Washington D.C.Washington D.C.: Urban Institute;
[5] Teal, R. 1980. Shared ride taxi services as community public transit: final report. Washington, Springfield, Va.Washington, Springfield, Va.: The Office, for sale by National Technical Information Service;
[6] Lee, K. T., Lin, D. J. and Wu, P. J. 2005. Planning and design of a taxipooling dispatching system. Planning and design of a taxipooling dispatching system, 1903, 86-95.
[7] Vuchic, V. 2007. Urban Transit Systems and TechnologyJohn Wiley & Sons;
[8] Jeon, C. M., Amekudzi, A. A. and Vanegas, J. 2006. Transportation system Sustainability issues in high-, middle-, and low-income economies: Case studies from Georgia (US), South Korea, Colombia, and Ghana. Transportation system Sustainability issues in high-, middle-, and low-income economies: Case studies from Georgia (US), South Korea, Colombia, and Ghana, 132, 3, 172-186.
[9] Correia, G. and Viegas, J. M. 2011. Carpooling and carpool clubs: Clarifying concepts and assessing value enhancement possibilities through a Stated Preference web survey in Lisbon, Portugal. Carpooling and carpool clubs: Clarifying concepts and assessing value enhancement possibilities through a Stated Preference web survey in Lisbon, Portugal, 45, 2, 81-90.
[10] Hodges, A. 2006. 'Roping the Wild Jitney': the jitney bus craze and the rise of urban autobus systems 'Roping the Wild Jitney': the jitney bus craze and the rise of urban autobus systems 21, 3, 253-276.
[11] Cooper, J., Mundy, R. and Nelson, J. 2010. Taxi! : urban economies and the social and transport impacts of the taxicab. FarnhamFarnham: Ashgate;
[12] Dufour, D. "Shared taxis in Brussels : the missing link in urban transport?" Urbanicity Retrieved 10 of March, 2010.
[13] Lee, D. H., Wang, H., Cheu, R. L. and Teo, S. H. 2004. Taxi dispatch system based on current demands and real-time traffic conditions. Taxi dispatch system based on current demands and real-time traffic conditions, 1882, 193-200.
[14] Shrivastava, M., Chande, P. K., Monga, A. S. and Kashiwagi, H. 1997. Taxi dispatch: A fuzzy rule approach. Taxi dispatch: A fuzzy rule approach, 978-982.
[15] Wang, H., Lee, D. H. and Cheu, R. 2011. Microscopic Traffic Simulation based Dispatch Modeling for Taxi Booking Service. Microscopic Traffic Simulation based Dispatch Modeling for Taxi Booking Service, 187, 677-682.
[16] Von Massow, M. and Canbolat, M. S. 2010. Fareplay: An examination of taxicab drivers' response to dispatch policy. Fareplay: An examination of taxicab drivers' response to dispatch policy, 37, 3, 2451-2458.
[17] Seow, K. T., Dang, N. H. and Lee, D.-H. 2007. Towards an automated multiagent taxi-dispatch system. IEEE Conference on Automation Science and Engineering. Scottsdale, AZ, USA.
[18] Kim, H., Yang, I. and Choi, K. 2011. An agent-based simulation model for analyzing the impact of asymmetric passenger demand on taxi service. An agent-based simulation model for analyzing the impact of asymmetric passenger demand on taxi service, 15, 1, 187-195.
[19] Kim, H., Oh, J. S. and Jayakrishnan, R. 2005. Effect of taxi information system on efficiency and quality of taxi services. Effect of taxi information system on efficiency and quality of taxi services, 1903, 96-104.
[20] Alshamsi, A., Abdallah, S. and Rahwan, I. 2009 Multiagent Self-Organization for a Taxi Dispatch System. In: *Proceedings of the 8th International Joint Conference on Autonomous Agents and Multiagent Systems*, Budapest, Hungary.
[21] Darbera, R. 2010. Taxicab regulation and urban residents' use and perception of taxi services: a survey in eight cities. 12 th WCTR - World Conference in Transport Research. Lisbon, Portugal.
[22] Lee, D.-H., Wang, H., Cheu, R. and Teo, S. 2004. Taxi Dispatch System Based on Current Demands and Real-Time Traffic Conditions. Taxi Dispatch System Based on Current Demands and Real-Time Traffic Conditions, 1882, 193-200.
[23] Agatz, N., Erera, A. L., Savelsbergh, M. W. P. and Wang, X. 2011 Dynamic Ride-Sharing: a Simulation Study in Metro Atlanta. In: *Proceedings of the 19th International Symposium on Transportation and Traffic Theory*.
[24] IMT. 2006. Estudo Sobre as Condições de Exploração de Transportes em Táxi na Cidade de Lisboa, Instituto da Mobilidade e dos Transportes, I.P. Lisbon.
[25] Viegas, J. M. and Martinez, L. M. 2010 Generating the universe of urban trips from a mobility survey sample with minimum recourse to behavioural assumptions. In: *Proceedings of the 12th World Conference on Transport Research*, Lisbon.
[26] National Research Council (U.S.). Transportation Research Board. 2000. Highway capacity manual. Washington, D.C.Washington, D.C.: Transportation Research Board, National Research Council;
[27] Martinez, L. M., Viegas, J. M. and Eiró, T. 2011 A new approach to modelling distance-decay functions for accessibility and transport studies. In: *Proceedings of the World Symposium on Transport and Land Use Research*, Whistler, British Columbia.
[28] Martinez, L. M., Viegas, J. M. and Silva, E. A. 2009. A traffic analysis zone definition: a new methodology and algorithm. A traffic analysis zone definition: a new methodology and algorithm, 36, 5, 581-599.