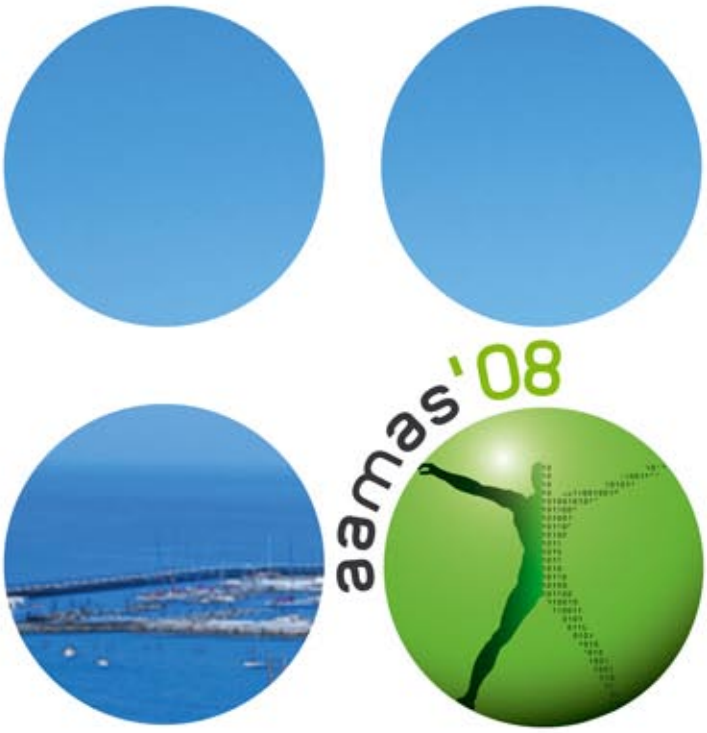


The Seventh International Conference on Autonomous Agents and Multiagent Systems

Estoril, Portugal
May 12-16, 2008

Workshop 14: Agents in Traffic and Transportation

Ana L. C. Bazzan
Franziska Klügl
Sascha Ossowski
(Editors)

ATT 2008

5th International Workshop on
Agents in Traffic and Transportation

held at AAMAS 2008
May, 13th, Estoril, Portugal

Ana L. C. Bazzan
Franziska Klügl
Sascha Ossowski
(editors)

Preface

Since its beginnings back in the year 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 – in particular from the area of Autonomous Agents and Multiagent Systems – and Transportation Engineering. The series aims at promoting cross-fertilization among these disciplines, focussing on how large-scale complex transportation systems can be modelled, simulated, and managed – both at micro and at macro level – employing techniques of agent-based simulation, decentralised coordination, and adaptive regulation.

This fifth edition of ATT was held together with the International Conference on Autonomous Agents and Multiagent Systems (AAMAS), in Estoril (Portugal) on May 13, 2008. Previous editions were: Barcelona, together with Autonomous Agents in 2000; Sydney, together with ITS 2001; New York, together with AAMAS 2004; and Hakodate, together with AAMAS 2006.

This edition of the workshop was the most successful in the history of ATT. In response to the call for papers, it attracted the submission of 26 high-quality papers from 16 different countries. All papers were thoroughly reviewed by 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 9 of these submissions as full papers, an acceptance ratio of 35%. In addition, 6 submissions were accepted as short 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, reinforcement learning, coalitions and collectives, to name just a few. We hope you will enjoy it! Finally, we owe a big "Thank you" to all people who dedicated their time and energy to make this edition of ATT a success: from authors and reviewers to hosts and chairs of the AAMAS conference.

Estoril, May 2008 — Ana Bazzan, Franziska Klügl, Sascha Ossowski

Programme Committee

Theo Arentze (TU Eindhoven, Netherlands)
Vicent Botti (Technical University of Valencia, Spain)
Brahim Chaib-Draa (Laval University, Canada)
Paul Davidsson (Blekinge Institute of Technology, Sweden)
Hussein Dia (University of Queensland, Australia)
Kurt Dresner (University of Texas at Austin, USA)
Khaled Ghedira (SOIE/MIAD-ISG, Tunisia)
Josefa Hernández (Technical University of Madrid, Spain)
Tom Holvoet (KU Leuven, Belgium)
Koichi Kurumatani (AIST, Odaiba, Japan)
Kai Nagel (TU Berlin, Germany)
Luis Nunes (Universidade de Porto, Portugal)
Eugenio Oliveira (Universidade de Porto, Portugal)
Guido Rindsfüser (Emch&Berger, Bern, Switzerland)
Rosaldo Rosetti (Universidade de Porto, Portugal)
Andreas Schadschneider (University of Cologne, Germany)
Kardi Teknomo (Arsenal Research, Austria)
Harry Timmermans (TU Eindhoven, NL)
Sabine Timpf (University of Augsburg, Germany)
Kagan Tumer (University of Oregon, USA)
Guiseppe Vizzari (University of Milan, Italy)
Peter Wagner (DLR, Germany)
Joachim Wahle (TraffGo GmbH, Germany)
Danny Weyns (KU Leuven, Belgium)

Referees

Alberto Fernández Gil
Cornelia Triebig
Dominik Grether
Eduardo Camponogara
Gustavo Kuhn Andriotti
Imen Boudali
Jan Persson
Johanna Törnquist Krasemann
Johannes Illenberger
Julien Laumonier
Luis Paulo Reis
Rutger Claes

Organizers

Ana L. C. Bazzan, UFRGS (Brazil)
Franziska Klügl, University of Würzburg (Germany)
Sascha Ossowski, University Rey Juan Carlos (Spain)

Table of Contents

Incorporating time and income constraints in dynamic agent-based models of activity generation and time use: approach and illustration

Theo A. Arentze
Eindhoven University of Technology
PO Box 513, 5600 MB
The Netherlands
+31 40 247 2283
t.a.arentze@bwk.tue.nl

D. Ettema
Utrecht University
PO Box 80115, 3508 TC
Utrecht, The Netherlands
+31 30 2532918
d.ettema@geo.uu.nl

Harry J.P. Timmermans
Eindhoven University of Technology
PO Box 513, 5600 MB
The Netherlands
+31 40 247 3318
h.j.p.timmermans@bwk.tue.nl

ABSTRACT
Existing theories and models in economics and transportation treat households' decisions regarding allocation of time and income to activities as a resource-allocation optimization problem. Arguably, this stands in contrast with the dynamic nature of day-by-day activity-travel choices. Therefore, in the present paper we propose a different approach to model activity generation and allocation decisions of individuals and households that acknowledges the dynamic nature of the behavior. We propose an agent-based model where agents, rather than acting on the basis of a resource allocation solution for a given time period, make resource allocation decisions on a day by day basis taking into account day-varying conditions and at the same time respecting available budgets over a longer time horizon. Agents that share a household interact and allocate household tasks and budgets among each other. We introduce the agent-based model and formally discuss the properties of the model. The approach is illustrated on the basis of simulation of behavior in time and space.

General Terms
Human Factors, Theory.

Keywords
Travel demand modeling, agent-based modeling, activity generation, time use, income constraints.

1. INTRODUCTION

The importance of financial constraints on individuals' time-use and activity choice has long been recognized particularly in the context of households' long-term mobility decisions (e.g., [1], [2], [3], [4], [5]). Long-term decisions such a residential location, job location, working hours, car possession and so on, may have significant implications for the amounts of time and money available for daily activities such as shopping, recreation and social activities. For example, a decrease of working hours increases available time but decreases income that can be spent in daily activities. As another example, a change of residential location to a place farther away from work increases commuting times and, hence, reduces available time budgets for activities. Because of such implications, long-term mobility decisions generally require trading-off utility derived from activities against utility of spending time and money in living, traveling or luxury goods.

Since the seminal work of Becker [6], households' decisions regarding the allocation of time and income to activities have been conceptualized and modeled as a resource-allocation optimization problem. In this approach, the allocation of time and income to activities of a household in a given time period is determined based on an objective to maximize a total household utility [8], [9]. Although later studies have accomplished important refinements and elaborations [7], the basic assumption of framing resource allocation behavior as a global optimization problem has not been questioned. Arguably, this stands in contrast with the dynamic nature of day-by-day activity choices. The physical and social environment in which activities are implemented, and the needs, desires and constraints for these activities are to an important extent stochastic and non-stationary. This dynamics imply that objectives and conditions are never exactly the same and decisions to implement and spend a certain amount of money and time in activities often need to be adapted to current circumstances.

Therefore, in the present paper we propose a different approach to model activity-resource allocation decisions of individuals and households that acknowledges the dynamics of the behavior. We propose an agent-based model where agents, rather than acting on the basis of a resource allocation solution for a certain time period, make resource allocation decisions on a day by day basis taking into account the specific conditions of the moment and at the same time respect available budgets over a longer time horizon. We show that by using a local decision rule the agents are able to act flexibly and at the same time maximize a utility they derive from activities over a longer term, given existing budget constraints. To accomplish this, the rule uses for each resource a threshold parameter representing the scarcity of the resource. The appropriate value of each threshold is not known a-priori. Through a process of learning based on experience, the agents gradually find the threshold values that optimize their behavior globally.

The proposed model will be incorporated in an integrated land-use, transportation system called PUMA [5], [10]. The model fits in current activity-based approaches to travel demand modeling. In existing conceptual frameworks, the programming and scheduling of activities are considered different, successive phases in a decision process for generating an activity schedule for a given day [15]. Programming decisions

determine which activities are conducted for how long and possibly also (tentatively) the locations where the activities are conducted. Activity scheduling decisions then determine the sequence of the activities, the exact timing of each activity, trip-chaining characteristics (e.g., insert yes or no a return home trip between two consecutive out-of-home activities) and transport modes used for the resulting tours. In this same scheduling phase, location choices for activities may be reconsidered, for example, to utilize opportunities for saving travel time, given trip-chains. The model we propose in the present study deals with activity programming decisions. This means that it needs to be complemented with an activity scheduling model, before it can be used for predicting individuals' activity-travel patterns. Furthermore, we note that, in contrast to existing approaches, our model is dynamic in the sense that the activity needs of an individual are considered to be dependent on the activity history of the individual. Rather than predicting an activity program for an average or typical day, the proposed model, as a consequence, generates activity programs that fit in a longitudinal activity pattern of an individual (of arbitrary length).

The paper is structured as follows. In Section 2, we first introduce the basic concepts of the approach. Then, in Section 3, we propose specifications of the components of the framework. Next, in Section 4, we describe results of simulations that we conducted to illustrate the system. Finally, we conclude the paper by discussing major conclusions and avenues for future research.

2. THE AGENT-BASED APPROACH

In this section, we describe the basic framework. We first consider the utility functions and a decision rule for a single agent and next discuss how this can be extended to take within household interactions into account.

2.1 Utility functions

Assume the activity repertoire of an agent is given by a list of activities $A = \{A_1, A_2, \ldots, A_n\}$. On each day of a given time period, the agent decides which activities it will conduct for how long and how much money it will spent (possibly zero). Mandatory activities, such as, for example, a work or school activity, may be scheduled for the current day as well. These activities are considered as given and fixed and reduce the available time for other activities. When mandatory activities and activities selected from the list have been programmed there may be time left on the day. Our model assumes that, by definition, this time is used for leisure-at-home purpose and as such generates utility as well. To put this in another way, the model assumes that time allocated to leisure (at the end of the day at home) is not a separate decision, but a result of all other activity decisions. We use the term slack time and slack activity to refer to the remaining time and use of remaining time on a day.

An activity produces utility and requires time and possibly other resources as inputs. If time is a scarce resource, then utility of time, defined as utility per unit time spent (denoted as UoT), is a useful concept. If time is scarce, then an optimality condition for the allocation of time to activities is that UoT is as much as possible equal across activities conducted (including the slack activity). This is easy to see: if the condition does not hold then it is possible to increase the utility by transferring time from an activity where it is less productive (in terms of utility) to an activity where it is more productive (in terms of utility). Time is merely one resource that is constrained by a budget. In addition, at least some activities also require monetary expenses. In full analogy, if money is a scarce resource, then utility of money, defined as the utility per monetary unit spent (denoted as UoM) is a key issue. If money is scarce then, for the same reason, it should be allocated to activities such that the UoM is the same across implemented activities.

A core assumption of the framework that we propose is that the utility of an activity is dependent on the activity history of the agent. Generally, the longer ago the last time an activity A_i has been conducted, the larger its current utility will be. When (positive or negative) substitution relationships exist between activities, the history of other activities should be taken into account as well. Although such interactions can be readily incorporated in the present framework, for clarity of presentation we leave them out of consideration here. Utility is furthermore dependent on travel demands involved, if any, the location where the activity is conducted and possibly chosen levels of time efficiency and quality. The following utility function captures these notions:

$$U_{di}(m,l,b,q) = V_d^{\mathrm{M}}(m,l) + V_{di}(t_i,l,b,q) + \varepsilon_{si} \quad (1)$$

where d is the current day, t_i is elapsed time since the last time an activity A_i was conducted, s denotes the day this happened ($s = d - t_i$), l is the chosen location for the activity, m is the chosen transport mode ($m = 0$ if no trip is involved), V_d^{M} is the travel-related utility, V_{di} is the activity-related utility, b is a chosen level of time efficiency, q is a chosen level of quality and ε_{si} is an error term. The d subscript of the activity-related component indicates that utility may be dependent on the day when the activity is conducted. This is the case for example when an agent has a specific intrinsic preference for a day of the week to conduct a certain activity (e.g., going out on Saturday). Time efficiency b is a relevant parameter if the agent can choose between conducting the activity in a hurry or at one's leisure or between a slow (and cheap) service and a fast (and expensive) service. This concept is comparable to the concept of activity intensity coined by Ashiru *et al.* [11]. In addition, quality q is a relevant parameter when the activity and location involve goods or services at different consumer price levels. An increase of time efficiency, when it implies an increase of effort rather than a faster service, reduces utility, whereas an increase of quality increases utility.

Besides generating utilities, activities and travel also use time. We model the time spent as follows:

$$T_{di}(m,l,b) = T_d^{\mathrm{M}}(m,l) + T_{di}(t_i,l,b) \quad (2)$$

where, as before, m and l represent transport mode and location choices, b represents choice of time efficiency for the activity, T_d^{M} represents travel time as a function of mode and location and possibly dependent on day (e.g., day-varying congestion levels), and T_{di} is time used for the activity as a function of elapsed time, location and chosen time efficiency. Since the need an activity intends to satisfy increases with elapsed time, the time needed (at the chosen efficiency level) is an increasing function of elapsed

time (e.g., a longer shopping list). Furthermore, given the time elapsed, T is a decreasing function of time efficiency b. Finally, the function depends on the day when the activity is conducted (e.g., shorter queues on Monday).

Apart from time, activities and travel may also involve monetary expenses. We model the money spent as:

$$E_i(m,l,q,b) = E_i^{\mathrm{M}}(m,l) + E_i(t_i,l,b,q) \qquad (3)$$

where E_i^{M} is costs of traveling to the chosen location with the chosen mode and E_i is the amount spent for the activity at the location. Expenditure for the activity, the latter component, is an increasing function of size of the need when the activity is conducted and, hence, of elapsed time. Furthermore, expenditure is an increasing function of quality level and it is an increasing function of time-efficiency in as far efficiency is a characteristic of a service that needs to be paid for.

Given the above definitions, the utility of time (UoT) and utility of money (UoM) can now be defined in a straight-forward way as:

$$u_{di}^{\mathrm{T}} = \frac{U_{di}(m,l,b,q)}{T_{di}(m,l,b)}, \quad u_{di}^{\mathrm{E}} = \frac{U_{di}(m,l,b,q)}{E_i(m,l,b,q)} \qquad (4)$$

where u_{di}^{T} and u_{di}^{E} are the UoT and UoM an activity i could generate when conducted on day d conditional upon the choices of mode, location, quality and efficiency. Note that by choosing an efficiency and quality level (and location and mode) an agent is able to adapt the time and money spent. Duration and expenditure are not fully symmetric in that respect. An increase of time efficiency may come at the cost of paying a higher price for a service and, then, increases expenditure (money can buy time). On the other hand, the model assumes that quality cannot be increased by spending more time. Furthermore, the concept of efficiency needs some clarification. In the model, efficiency is increased either by investing more effort (doing things in a hurry) or by paying more for a faster service. One could argue that service and effort refer to different dimensions. Although it is possible to treat the dimensions as separate variables in the model, we assume that within activities either one of the two dimensions is relevant, so that a single variable, with an activity-context dependent meaning, suffices.

To elaborate and extend the latter issue, we distinguish the following three activity types, which we denote as Type I, II and III activities:

1. Type I: Neither time-efficiency nor quality can be increased by spending more money.
2. Type II: time-efficiency can be increased by spending more money.
3. Type III: quality can be increased by spending more money.

Obviously, these categories are not mutually exclusive: an activity can be of Type II and Type III at the same time. In case of a Type-I activity, no resource allocation choice is left when location and transport mode have been chosen: the amount of time and money spent are determined by elapsed time. In case of a Type-II activity, time and money are to some extent compensatory in the sense that a lack of time can be compensated by spending more money (e.g., choosing a more expensive service that is faster). In case of a Type-III activity, the individual can spend more money (e.g., pay a higher price for a higher-quality service) and increase the utility derived from the activity without time consequences.

2.2 A local decision rule

The activity utility function given by Equation (1) describes the utility of a particular activity on the current day as a function of time efficiency, quality, location and mode decisions. The function is however dynamic as it takes into account current needs (elapsed time) and intrinsic preferences for a day. The agent-based model we propose assumes that agents use a local decision rule in the sense that they make the decisions on a day-by-day basis. Although this is plausible in terms of what individuals do in reality, it seems to be in conflict with the fact that time and money budgets are defined for a longer time frame than a day. As we argued in previous work [12], however, time budget constraints can be adequately dealt with by a local decision rule of the following form:

R_1: Implement an activity on the earliest day when the UoT of the activity under optimal time efficiency, quality, mode and location choice exceeds a threshold value for that day of the week.

A threshold value is included for each day of the week to account for possible day-by-day variation on time spent on mandatory activities (e.g., more time available in weekend). If the threshold value for each day of the week is appropriately chosen, this rule makes sure that 1) each implemented activity produces approximately an equal UoT and 2) available time is fully used in the sense that utility of slack time equals the utility of activity time. Note that the utility as well as the required duration of an activity increases over (elapsed) time. If utility increases with a faster rate than required duration, then UoT increases over time and a moment will come that it exceeds the threshold. The day-of-the-week threshold values that produce this result are, however, not a-priori known. We also showed how the threshold values can be found through an iterative adjustment procedure based on trial and error. Starting with an arbitrary initial value (for each day of the week), an activity plan for a sufficiently long time period is generated using R_1. Next, utility of slack time is compared to utility of activity time for each day of the week:

R_2: If the utility of slack time is higher than the utility of activity time, then adjust the threshold upwards. If the utility of slack time is lower than the utility of activity time, then adjust the threshold downwards.

where, as before, slack time is time left on a day after having conducted mandatory and selected activities from list A. If the threshold for one or more days is adjusted, then an activity plan is re-generated for the same period and R_2 is applied again. This cycle is repeated until convergence. In equilibrium, the utility of slack time is equal to the utility of activity time and the utility of time of each activity is at or just above the threshold and, hence, approximately the same.

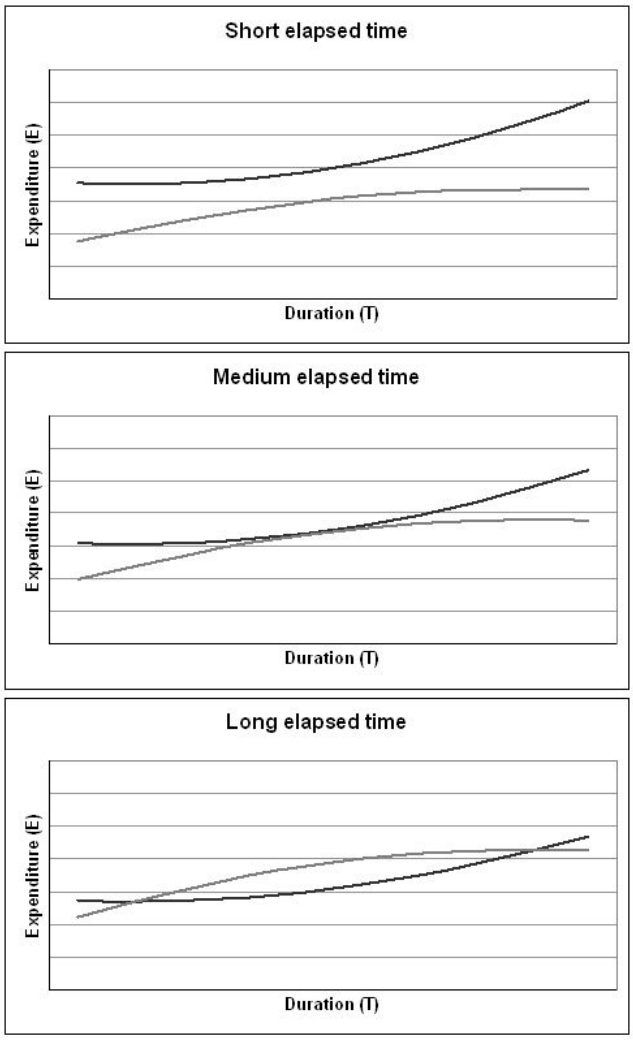


Figure 1. Utility of time (black line) and utility of money (gray line) Influence of elapsed time on for a Type-III in three stages of need development

The activity-day selection rule R_1 and threshold adjustment rule R_2 focus on the time-budget constraint and do not take expenditure into account. Generalization of the rules is straightforward. The extended activity-day selection rule can be written as follows:

R_1' Implement an activity on the earliest day when the UoT and UoM of the activity under optimal time efficiency, quality, mode and location choices both exceed their threshold value for that day of the week.

Note that for UoM a single threshold value for each day of the week suffices, as financial budgets do not vary on a daily basis. The appropriate threshold value for UoM can be found by a similar threshold adjustment rule as in the case of time. In full analogy, the complementary rule comes down to:

R_3: If total expenditure exceeds the money budget, then adjust the threshold upwards. If not all available money is used, then adjust the threshold downwards.

We illustrate the behavior of the system based on three graphs shown in Figure 1. As an example, the graphs relate to a Type-III activity and depict three moments in the cycle of an activity, i.e. where the need is low (top graph), a later moment where the need is medium (middle graph) and a still later moment where the need is high (bottom graph). In each graph, the black curve represents the combinations of duration and expenditure where UoT equals its threshold value and the gray curve represents the resource allocations where the UoM equals its threshold value. Thus, above the black curve are resource allocations allowed by the time-budget constraint and below the gray curve are the resource allocations allowed by the money-budget constraint. From early to the late stage, the UoT threshold curve (black line) shifts downwards indicating that, as the need increases, increasingly lower levels of expenditure suffice to meet the threshold. At the same time, the UoM threshold curve (gray line) moves upward indicating that, as the need increases, increasingly higher levels of expenditure are allowed given this threshold. In the second stage (the middle graph), the need has reached a level where the two threshold curves start to overlap. Hence, this is the first moment when implementing the activity is feasible. In the late stage, multiple resource allocations are feasible, namely all allocations falling in the area enclosed by the two graphs.

Given the rule that an activity is conducted at the earliest moment when the thresholds conditions are met, the threshold condition leaves no choice as to how much time and money to spend for the activity when it is implemented. Thus, a ready conclusion that can be drawn from the illustration is that in a system where all or at least some activities are of Type III, individuals always use their time and money budgets completely. This conclusion holds if the money curve is convex and moves down and the time curve is concave and moves up, as the need for the activity increases. Then, the first moment when the activity is feasible is always the moment when the curves intersect in a single point. Because the activity will be selected at that moment, there is no time and expenditure choice left. In reality, of course, day is a discrete variable and, hence, time does not pass in a continuous fashion. In case of Type-III activities, however, even if a choice is left, money will be spent exactly up to the point where UoM meets its threshold value. For example, if more money were spent utility would rise but too little to prevent the UoM from dropping below its threshold.

For Type-II activities the same mechanism applies. For these activities there is even another level of dynamics that makes sure that budgets are fully used. Assume for example that an agent would consistently choose to spend more money to save time. Then, not all time is used and by the working of the threshold adjustment procedure the UoT threshold would be adjusted downwards. This would continue till the moment when the UoT threshold line would intersect the UoM threshold line in a single point. At that moment the activity will be implemented without leaving a choice regarding the quantities of time and money. This self-organizing behavior makes sure that the system indeed exhausts its budgets.

For Type-I activities, however, the outcome may be different. For these activities there is a single point in time where the UoM-threshold line meets the line that represents the fixed quantity ratio between the resources. This point does not necessarily coincide with the point where the UoT equals its threshold. Thus, there are two possibilities with different outcomes. If time is the limiting factor, the UoT threshold determines when the activity is conducted and how much money is spent. On the other hand, if money is the limiting factor, the UoM threshold determines when the activity is conducted and how much time is spent. In the first case, money is left and in the second case time is left. Furthermore, we note that also under Type-II and Type-III conditions circumstances are conceivable where not all resources are used. The utility and resource functions impose limitations on the extent to which time can be exchanged by money (Type-II) or more money can be spent to increase utility (Type-III). If the money budget is large, this may mean that money is not a limiting factor for utility and may not be fully used. In such a case, the UoM threshold would drop to zero.

2.3 Multi-person households

The household context is relevant when at least some activity needs are shared by the individual agents sharing a household. If multiple agents are able to conduct such household activities, then time can be re-allocated between agents. Furthermore, an important implication is that agents may derive utility from an activity even if it is conducted by someone else. The proposed system assumes that agents have their own perception about needs and activities and interact pair-wise with each other to consider re-allocations of (household) activities using the following rule:

R_4: If $dU_{di}^{g}(h, g)$ is the change in utility for agent g when g would take over a (household) activity i from agent h on day d and $dU_{di}^{h}(h, g)$ is the change in utility for agent h of the same re-allocation, then implement the re-allocation only if $dU_{di}^{g}(h, g) + dU_{di}^{h}(h, g) > 0$. If g takes over an activity from h then g re-considers the duration of the activity based on his own perceptions.

This rule is applied iteratively to all pairs of agents and all household activities. Since all agents in the household apply R_1', this means that each household activity is always conducted on the earliest day when it meets the lowest threshold value across the agents that can conduct the activity. By using R_4 an activity then possibly may be re-allocated among the individuals.

Finally, in cases where multiple individuals share a household, multiple money budgets are relevant. If there are n agents, then there are $n + 1$ sets of needs, namely n sets of personal needs and a set of shared needs. A money budget needs to be defined for each set of needs implying that $n + 1$ UoM threshold values are required to represent existing money-budget constraints. In everyday terms, this means that individuals sharing a household should decide on how much of the household income they wish to use for shared needs and how much they wish to use for the personal needs of each individual. Note that this is another asymmetry between the two resources: for the time resource there exists no rationale for adopting a shared budget.

2.4 Concluding remark

The dynamic-process approach makes it possible to account for many irregularities that exist in the real world. Specifically, the proposed system takes the following conditions into account:

1. Mandatory activity time may vary from day to day.
2. Intrinsic preferences for certain activities may vary by day of the week.
3. Physical conditions (e.g. traffic) and, thus, time demands may vary from day to day
4. Preferences and perceptions may differ between individuals sharing the same household.

The activity-day selection rule R_1', the threshold adjustment rules R_2 and R_3 and the allocation rule R_4 are all sensitive to day-varying and person-varying conditions and perceptions and at the same time consistent with an objective of maximizing utility within resource constraints. However, it should be noted that, because of the irregularities and discrete character of activity-day selection decisions, temporal patterns where all activities generate equal UoT and equal UoM may not emerge in the system (and neither in reality). For example, it may not be possible to (further) re-allocate time between individuals or between days of the week to (further) solve existing differences in UoT between days of the week. In that sense, adjusted UoT thresholds accurately represent day-varying time pressures on an individual's agenda.

3. POSSIBLE SPECIFICATIONS

In this section, we propose further specifications of the functions involved in the above framework that suite the purposes of transportation modeling. First, regarding the costs of traveling we propose the following simple function:

$$E_i^{\mathrm{M}}(m,l) = P_m^{\mathrm{M}} + p_m^{\mathrm{M}} D(m,l) \qquad (5)$$

where $D(m, l)$ is traveled distance given mode choice m and location choice l, P_m^{M} is a constant costs for using mode m and p_m^{M} is a costs per unit travel distance for mode m. Not all terms may be relevant. For private-vehicle modes the constant costs component normally is equal to zero or used to represent parking costs. For public transport tariff structures of transport modes may be more complex. Ticket prices may not be a linear function of distance, and so on. Even in those cases, however, the above linear function may suffice for a reasonable approximation of actual costs. Furthermore, we note that in case of multi-activity tours it is not always straight-forward which part of traveling should be attributed to which activities. However, we leave this issue out-of-consideration here.

We assume that activities that are relevant for transportation modeling may be of Type I or III, but are never of Type II. That is, we assume that for none of the activities time can be saved by spending more money. This means that the time efficiency parameter, b, refers exclusively to an effort of the agent and can be dropped as argument from the expenditure function (cf. Eq. 3). Furthermore, we operationally define quality

parameter q and efficiency parameter b on a zero-one scale, where zero means lowest level and one means highest level of quality and efficiency respectively.

Given these assumptions, we propose the following simple linear function to define amount of expenditure for an activity of Type I or Type III in general, as follows:

$$E_i(t_i, l, q) = (1-q)E_i^{\min}(t_i, l) + qE_i^{\max} \quad (6)$$

$$E_i^{\min}(t_i, l) = P_{li}^{\min} + p_{li}^{\min} t_i \quad (7)$$

$$E_i^{\max}(t_i, l) = P_{li}^{\max} + p_{li}^{\max} t_i \quad (8)$$

where, as before, t_i is elapsed time, $P_{li}^{\min}$ is a constant price and $p_{li}^{\min}$ a price per unit of the need for the activity (as measured by elapsed time) at a lowest quality level and $P_{li}^{\max}$ and $p_{li}^{\max}$ are the constant and variable prices at a highest quality level. As implied by Equation (6), the choice of quality level determines the actual amount spent, given the size of the need on the day the activity is conducted and the location. Note that Type I is a special case of Equation (6) where $P_{li}^{\min} = P_{li}^{\max} = P_{li}$ and $p_{li}^{\min} = p_{li}^{\max} = p_{li}$ and, where, as a consequence, parameter q is redundant.

The proposed function for time spent on an activity can be defined in a fully analogues way as:

$$T_{di}(t_i, l, b) = bT_{di}^{\min}(t_i, l) + (1-b)T_{di}^{\max}(t_i, l) \quad (9)$$

$$T_{di}^{\min}(t_i, l) = \alpha_{li}^{\min} + \delta_{li}^{\min} t_i \quad (10)$$

$$T_{di}^{\max}(t_i, l) = \alpha_{li}^{\max} + \delta_{li}^{\max} t_i \quad (11)$$

where $\alpha_{li}^{\min}$ is a constant time investment and $\delta_{li}^{\min}$ a time demand per unit of size of the need (as measured by elapsed time) at the highest level of time efficiency and $\alpha_{li}^{\max}$ and $\delta_{li}^{\max}$ are the corresponding figures at the lowest level of time efficiency. The travel-time component T_d^{M} (Eq. 2) is a more complex function of the transportation system which can be modeled as usual and which we will leave out of consideration here.

A possible specification of the utility function which captures the notions discussed in the previous section is as follows:

$$V_{di}(t_i, l, b, q) = V_{di}^0 + (1+q)^{\gamma_i}(1+b)^{\chi_i} f_i(t_i) \quad (12)$$

where V_{di}^0 represents a constant component, which may vary between days, f_i is a need-related component that varies as a function of time elapsed and γ_I and χ_i are activity-dependent parameters. Parameter $\gamma_I \geq 0$ defines a weight of quality and parameter $\chi_i \leq 0$ a weight of efficiency in utility. Note that the multiplicative form makes sure that quality and efficiency act so as to rescale the utility derived from need satisfaction. Several functional forms for the latter need-size function f could be considered. In earlier studies ([12], [13]), we proposed a logistic function. Arguably, an essential characteristic of this function is that a subjectively felt need grows with declining rate over time. A simpler function that also displays this property and has been proposed in several empirical studies on time-use modeling (Kitamura 1984) is the following logarithmic function:

$$f_i(t_i) = \beta_i \ln(t_i + 1) \quad (13)$$

where β_i is a scaling factor representing the growth rate of the need for the activity. For example, high frequency activities are characterized by a large value for beta, and vice versa. Unity is added to elapsed time in the argument of the logarithm to make sure that the need is larger than zero after an elapsed time of one day.

In terms of UoM and UoT, the system has the following properties. Equation (6) implies that expenditure increases linearly with quality level (given the scale on which we measure quality). Equation (12) implies that utility increases with decreasing marginal utility if $0 < \gamma_I < 1$. Under that setting, therefore, utility of money decreases with increasing expenditure. In the special case where $\gamma_I = 1$, utility increases linearly with expenditure and, hence, utility of money will be independent of the amount spent. We consider the decreasing UoM behavior to be more realistic meaning that generally a setting of $0 < \gamma_I < 1$ is appropriate. As for the time resource, Equation (9) implies that time spent decreases linearly with efficiency level (again, given the scale we use to measure efficiency). Equation (12) implies that utility decreases with a decreasing rate when efficiency increases for all settings of $\chi_i < 0$. As a consequence, UoT increases with increasing time efficiency, as we would expect.

Cross-elasticities are also evident in the system. Keeping the time spent constant, spending more money for an activity means that the utility increases and, hence, also UoT increases. Vice versa, keeping expenditure constant, spending less time for an activity leads to an increase of utility and, with that, an increase of UoM. Combined with the threshold adjustment rule, these relationships give rise to the following self-organizing behavior. Spending more money for activities means that activities exceed UoT thresholds earlier in terms of elapsed time and, thus, lead to increase of time expenditure. To prevent time shortage, the UoT threshold is adjusted upward and the activity frequency will be restored. Through this mechanism, an increase of available money (for activities) drives the UoT threshold up if the change is not accompanied by an equivalent increase of time budget. Or to put it another way, the system predicts that income correlates positively with utility-of-time demands. On the other hand, spending more time on activities (doing things in a less efficient way) means that activities exceed the UoM threshold earlier and to prevent shortage of money the threshold for UoM of money is adjusted upward if the money budget does not increase. Thus, the model predicts that utility-of-money demands increase with increasing time budgets.

4. ILLUSTRATION

In this section, we describe some results of a simulation to illustrate the model system. The simulation focuses on day-to-day activity choices for a multi-week period of two hypothetic persons sharing a household.

4.1 The simulation system

The simulation was conducted using an existing agent-based system that we developed in earlier work [14]. The system is based on a needs-based model of activity generation, which assumes that activities are driven by a set of needs which grow

over time. In this system, there is not necessarily a one-to-one relationship between needs and activities: a single activity may have a (positive or negative) influence on multiple need dimensions at the same time. The model proposed here is implemented as a special case where one activity acts on one need. The system assumes a logistic (S-shaped) function to describe need-growth.

As for the resource functions, the system assumes that time spent on an activity is a function of the size of the need at time of implementation. However, time efficiency is not a parameter in this framework and, hence, this choice facet cannot be modeled. As for monetary expenditure, the system assumes as parameters of each activity and location combination a constant price (P), a price per unit duration (p) and a maximum amount of expenditure. The constant and price per unit define a minimum amount of expenditure given an activity duration. The minimum and maximum defines a range within which agents can choose the actual amount spent at the moment an activity is implemented. Utility is an increasing function of expenditure with decreasing marginal utilities in line with the specifications we proposed in this study. Furthermore, the agents use rules R_1-R_4 for their daily activity decisions. In sum, the system allows us to run the model proposed here and simulate the behavior under Type-III activities with fixed time efficiency.

Table 1 represents the activity settings assumed in the case. Shopping (daily and non-daily), service-related activities, medical activities, fitness and going-out (eating/drinking and cultural) need particular facilities in the spatial environment. In addition, the activity list includes several in-home activities including, jobs in or around the house, housekeeping, leisure passive (e.g., reading, watching TV etc.) and sleeping. Daily shopping, service-related activities and housekeeping are considered household activities, i.e. activities that can be conducted by both agents and satisfy a shared need by both agents. Leisure passive is considered a slack activity, i.e. the activity an agent is engaged in during time on a day not occupied by the activities in the list. Finally, the list specifies a work activity for each person.

The residence of the agents is explicitly situated in space somewhere in the Netherlands. For each facility-based activity, a location choice set for each transport mode is defined for each agent. Facilities for the activity are available at these locations. Travel distances by mode are calculated based on shortest paths across the Dutch road network. Fast and slow modes are distinguished as possible transport modes. Activities are consistently evaluated under best location and mode choices. For fast modes no direct implications for utility are assumed (only indirect namely through saving time). A slow mode on the other hand produces a (positive) utility depending on an existing need for physical exercise. The latter is also the need involved in the fitness activity meaning that fitness and slow mode are partly substitutable.

A beta parameter for each activity describes how fast the need for the activity grows over time (in the context of a logistic growth function). Leisure passive is considered the slack activity. Furthermore, it is a special activity in the sense that the need for the activity is supposed to grow within a day rather than across days, as the other activities do. As reflected by the scale of the beta parameter, the unit of time is a minute rather than a day. The asymptotic maximum, which is an additional parameter of a logistic function, was not differentiated and set to 100 units for each activity. As an exception, the maximum for the leisure activity is set to 300 units. Activity duration parameters (not shown) were set based on assessments of typical time requirements for activities.

Table 1. Assumed activity settings

Activity	Need (beta)	Expenditure (P, p, Emax)
Daily goods (H)	0.700	(10, 0.333, 100)
Non-daily good	0.350	(0, 0.667, 200)
Services (H)	0.250	(10, 0.333, 100)
Medical	0.080	(20, 0.5, 200)
Go-out (drink-eat)	0.500	(0, 0.2, 100)
Go-out (cultural)	0.080	(30, 0, 200)
Fitness	0.250	(5, 0.667, 100)
Social	0.600	(0, 0, 0)
House keeping (H)	0.800	(0, 0, 0)
House jobs	0.350	(0, 0, 0)
Leisure passive	0.008	(0, 0, 0)
Sleep		(0, 0, 0)
Work		(0, 0, 0)
Travel by fast mode		(0, 0.17, -)
Travel by slow mode	0.250	(0, 0, 0)

In-home activities (Jobs, Housekeeping, Leisure, Sleeping) do not involve expenditure and neither do social and work activities (except that traveling may incur costs). Activities that do involve expenditures were all considered Type-III activities, with a gamma value of 0.5. Work is considered a mandatory activity, with fixed times and durations. Since it is a fixed activity no utility function is specified for work. The weekly work schedule of each agent is predefined and determines the time budget for the flexible activities on each day for each agent. P1 and P2 differ regarding this schedule. Being a fulltime worker, P1 has a work activity of 8 hours on each weekday. P2 has a part-time job which involves four workdays (Wednesday off) of 6 hours a day.

We simulated the day-by-day activity choices of the agents arbitrarily for a period of 7 weeks; the initial sizes of activity needs (i.e., elapsed times) are randomly chosen (assuming that the agents have the same perceptions of household needs). On each day, each agent goes through its list of activities and determines for each activity the best choices for choice facets and travel time and evaluates the utility given the best choices and elapsed time for the activity. The choice facets include location, transport mode, duration and expenditure. Best choices are choices that meet the existing UoT and UoM threshold requirements and maximize utility within that constraint. Activities that meet the threshold constraints (under best choices) are put on the activity agenda for the day and person concerned.

Household activities are also included in the activity list of each person. Which person, P1 or P2, conducts the activity

is considered an additional choice facet of household activities. Each person puts household activities on an own agenda making a best person choice considering the appropriate thresholds, but without consulting the other person. This means, for example, that P1 may put a housekeeping activity to be performed by P2 on its agenda when P1 considers this feasible given (its knowledge of) P2's threshold constraints and based on its own perception of the need for the activity. Having determined their activity agendas independently of each other, the two agents start a negotiation process. In this process they apply rule R_4 to each household activity, to see if a re-allocation is desired. A change of an initial allocation may occur if opportunity costs or perceptions differ. Opportunity costs are defined as the utility loss associated with sacrificing leisure time. After completing the negotiations the agents agree with each other on who does which activity. Finally, they implement their activity agendas and update their needs depending on the activities implemented. They repeat the same process for the next day, and so on.

Available time on a day is the time not occupied by the mandatory activities working and sleeping. Expenditures are charged at a personal money budget if they serve personal needs and at a shared household budget if they serve a household need (irrespective who conducts the activity). The threshold adjustment procedure uses rules R_2 and R_3. This involves recursively generating activity agenda's for a full 7-week period under currently assumed threshold values and evaluating the budget constraint. Since available time may vary by day of the week, each agent uses a threshold value by day of the week for time use. Since threshold adjustments of one person may affect utility of time of the other person, the two agents perform the adjustment procedure simultaneously. As for expenditure, a threshold value is related to each of the three budgets. A threshold is decreased if not all money is used and increased if too much money has been spent.

Because adjustment of a money threshold may have an influence on the UoT and, vice versa, an adjustment of a time threshold may have an influence on UoM, the adjustment procedure is run for one resource nested within the adjustment for the other resource. Arbitrarily, we chose to run time-threshold adjustment within money-threshold adjustment. This means that for each implemented adjustment of a money threshold the time thresholds are re-adjusted. A linear approximation method is used to determine best guess threshold adjustments in each step of the procedure. Considering the fact that (a multi-week) activity generation is an embedded processes, the routine is very efficient and takes only several seconds to complete on a standard PC.

4.2 Some results

As an example, we consider the results of a simulation where the total household budget was (arbitrarily) set to 1800 Euros per month. As it appears, allocating this budget as 658 Euro, 702 Euro and 433 Euro to the personal budget of P1, the personal budget of P2 and the shared budget, respectively, yields an approximately equal utility of money of 1.92 (P1), 1.83 (P2) and 1.81 (shared) across budgets. This means that, if the interests of P1, P2 and Shared have equal weight, this allocation maximizes the overall household utility (given the activity settings). The UoT of each person differs between days of the week. On average, they are 0.38 (P1) and 0.31 (P2). Tables 2-4 portray results per activity based on activity patterns for a 7 week period after adjustment of thresholds.

Table 2 represents activity frequencies (working and sleeping not included). On average, P1 and P2 perform 3.8 and 5.2 activities per day, respectively. The larger activity frequency for P2 eventually is due to the fact that P2 works less hours a week and, therefore, has a larger time budget resulting in a lower time threshold for activities. Also note that P2 conducts considerably more household activities (daily goods and housekeeping). This reflects a task allocation effect. Fitness is conducted by none of the two persons. As it appears, both persons prefer to use the slow mode every now and then to satisfy a need for physical exercise.

Table 2. Activity frequency by activity and agent (average number per day)

	P1	P2
Daily goods	0.07	0.21
Non-daily good	0.14	0.21
Services	0.04	0.04
Medical	0.04	0.04
Go-out (drink-eat)	0.25	0.36
Go-out (cultural)	0.04	0.04
Fitness	0	0
Social	0.36	0.57
House keeping	0.04	0.68
House jobs	0.14	0.43
Total	3.83	5.15

Table 3 shows the monthly expenditure per activity related to each budget (P1, P2 and Shared). Expenditures for social and work activities relate only to travel costs (by fast mode). In sum, P2 gets a larger budget for its (personal) activities since the shorter work hours for this person means that more time can be spent and, thus, more value can be generated per unit expenditure on activities.

Table 3. Expenditures by activity and budget (Euro / month)

Activity	P1	P2	Shared
Daily goods	0	0	350
Non-daily good	165	209	0
Services	0	0	83
Medical	41	40	0
Go-out (drink-eat)	277	294	0
Go-out (cultural)	31	31	0
Fitness	0	0	0
Social	48	51	0
Work	95	76	0
Total	658	702	433

Table 4. Utility of money by activity and budget

Activity	P1	P2	Shared
Daily goods	0	0	1.81
Non-daily good	1.67	1.52	0
Services	0	0	1.81
Medical	1.69	1.54	0
Go-out (drink-eat)	1.68	1.52	0
Go-out (cultural)	1.86	1.54	0
Fitness	0	0	0
Social	8.29	7.97	0
Total	1.92	1.83	1.81

Finally, Table 4 shows the UoM per activity and budget (P1, P2 and shared). Expenditures for social and work activities relate only to travel costs (by fast mode). The high UoM for social activities follows from the fact that these activities generate utility against only travel costs. In sum, P2 gets a larger budget for its (personal) activities since the shorter work hours for this person means that more time can be spent and, thus, more value can be generated per unit expenditure on (non-work) activities.

5. CONCLUSION AND DISCUSSION

In this paper, we showed how time and money constraints can be incorporated in a dynamic agent-based model of activity-travel choice. The standard economic approach assumes that activity generation and time use behavior is based on global solutions for an entire time period. In contrast, the model we proposed shows that by using a local decision rule an equivalent result can be obtained provided that agents are given time to learn based on experience. The dynamic agent-based approach makes it possible to describe behavior under day by day variation in budgets, physical conditions and preferences for activity choices. Furthermore, the model accounts for interactions between agents within households in terms of task allocation. In stark contrast to existing approaches, the model imposes virtually no restrictions on the level of detail of the used activity classification. For example, individuals' activity repertoires may include a large set of mandatory, discretionary in-home and out-of-home activities. The model simultaneously deals with choice facets of activities such as location and transport mode. Finally, we mention, that unlike global optimization approaches, the agent-based system is computationally very efficient. Agents have only limited memory requirements. All they need to remember is their current needs and dynamic utility thresholds regarding the use of time and money. The activity generation and negotiation process only requires linear list processing, which requires only a minimum of computation. Given good initial threshold settings, the threshold adjustment process is very efficient as well. This means that the model can be used in large-scale micro-simulation systems without causing excessive computation times.

Several problems and ways to extend the model could be considered in future research. A first issue relates to the estimation of parameters of activity utility functions. As the model is dynamic, existing one-day or two-day activity-travel diary data, which are collected in standard surveys in many countries, may not suffice. At least the surveys should be extended to reveal for the day observed the activity-history in terms of elapsed time for each activity that can be conducted (not just the activities that are conducted) on the observed day. Longitudinal activity diary data would offer more information but clearly also incur higher costs of data collection. Furthermore, data on the amount of money spent in the context of activities is needed to estimate quality choice parameters in the present framework. This would require an extension of existing survey instruments too, as this data is generally not covered in existing activity diary data collections in transportation research.

Second, our model focuses on the activity programming process and does not consider scheduling behavior. In a scheduling phase agents may be able to economize traveling by using opportunities for trip-chaining and adapt location, mode and possibly activity choices to utilize such opportunities. Furthermore, the timing of activities within a day is not considered in the present model. Opening hours (e.g., of stores) or commitments (e.g., joint activity participation) generally restrict choice opportunities for the timing of activities. Consequences of such restrictions are not limited to the activities for which they hold but also impact the time windows for preceding and succeeding activities and so on. In exceptional cases, timing conflicts may render an activity program infeasible and necessitate a subject to cancel an activity (e.g., postpone it to the next day). A complementary scheduling model is needed to cover these aspects.

6. REFERENCES

[1] Jara-Diaz, S.R., and F.J. Martinez (1999) On the specification of indirect utility and willingness to pay for discrete residential location models, Journal of Regional Science, 39, 675-688.

[2] Moeckel, R., Schurmann, C. and Wegener, M. (2002), Microsimulation of urban land use, Paper presented at the 42nd European Congress of the Regional Science Association, Dordmund, August 27-31, 2002.

[3] Waddell, P. (2002), UrbanSim, modeling urban development for land use, transportation, and environmental planning, Journal of the American Planning Association, 68, 297-314.

[4] Hunt, D.J. and J. Abraham (2003) Design and application of the PECAS land-use modeling. Paper presented at the 8th International Conference on Computers in Urban Planning and Urban Management, Sendai, Japan.

[5] Ettema, D.F. and H.J.P. Timmermans (2006), Multi-agent modelling of urban systems: The PUMA system, In: Proceedings of the 7th AMUS Conference, ISS, Aachen, pp. 165-172.

[6] Becker, G. A. (1965) Theory of the allocation of time, The Economic journal, 75, 493-517.

[7] DeSerpa, A. (1971) A theory of the economics of time, The Economic Journal, 81, 828-846.

[8] Kato, H., and M. Matsumoto (2007) Intra-household interaction analysis among a husband, a wife and a child using the joint time-allocation model, Proceedings of the 85th Annual Meeting of the Transportation Research Board, Washington, D.C., (CD-ROM)

[9] Kockelman, K.M. (2001) A model for time- and budget-constrained activity demand analysis, Transportation Research B, 35, 255-269.

[10] Ettema, D., K. de Jong, H. Timmermans and A. Bakema (2006), *PUMA: multi-agent modelling of urban systems*. In: E. Koomen, A. Bakema, J. Stillwell and H. Scholten (eds.): *Land use modeling*, Springer, pp. 237-258.

[11] Ashiru, O., J.W. Polak, R.B. Noland (2004) The utility of schedules: a theoretical model of departure time choice and activity time allocation with application to individual activity schedules, Proceedings of the 82th Annual Meeting of the Transportation Research Board, Washington, D.C., (CD-ROM).

[12] Arentze, T.A. and H.J.P. Timmermans (2007) A dynamic model for generating multi-day, multi-person activity agendas: approach and illustration In: Proceedings of the 85th Annual Meeting of the Transportation Research Board, Washington, D.C., (CD-ROM).

[13] Arentze, T.A. and H.J.P. Timmermans (2006), A new theory of dynamic activity generation. In: Proceedings of the 85th Annual Meeting of the Transportation Research Board, Washington, D.C., (CD-ROM: 20 pp.)

[14] Arentze, T.A., D. F. Ettema and H.J.P. Timmermans (2007) Micro-simulation of individual space-time behavior in urban environments: a new model and first experiences, Paper presented at the 10th International Conference on Computers in Urban Planning and Urban Management, Foz de Iguacu, Brazil.

[15] Gärling, T., M.-P. Kwan and R.G. Golledge (1994) Computational-process modeling of household travel activity scheduling. Transportation Research B, 25, 355-364.

An Agent oriented approach to Transportation Regulation Support Systems *

Flavien Balbo
Université Paris-Dauphine
LAMSADE/CNRS - GRETIA/INRETS
Place du Maréchal de Lattre de Tassigny
75775 PARIS Cedex 16 Paris, France
balbo@lamsade.dauphine.fr

Suzanne Pinson
Université Paris-Dauphine
LAMSADE/CNRS
Place du Maréchal de Lattre de Tassigny
75775 PARIS Cedex 16 Paris, France
pinson@lamsade.dauphine.fr

ABSTRACT

This paper presents an agent-based approach to design a Transportation Regulation Support System (TRSS). Based on a multi-agent modeling of a urban transportation network, the objective of our approach is to integrate the functionalities of the existing information system with the functionalities of a decision support system. The TRSS monitors the network activity and adjusts itself to the environment changes, that is to say it automatically detects incoherent data (regulation under normal conditions) and traffic disturbances and then it automatically proposes solutions to optimize the traffic flow (regulation under disturbed conditions). To demonstrate our approach, a transportation regulation support system called SATIR (Système Automatique de Traitement des Incidents en Réseau - Automatic System for Network Incident Processing) is presented. SATIR has been tested on the Brussel transportation network (STIB). Lastly, we show how using the multi-agent paradigm opens perspectives regarding the development of new functionalities to improve the management of a bus network.

General Terms

public transportation network management

Keywords

agent-based applications, environment, Decision Support System

1. INTRODUCTION

The development of the surface public transportation networks (SPTN) is a main issue from the ecologic, economic and societal viewpoints. But, this means of transportation competes against the comfort of the personal vehicles and contrary to the guided transportation (train and subway) it has to support the traffic disturbances. The result is that SPTN are often considered as not reliable. If the projects like the Bus Rapid Transit highlight the benefits of an improvement of the quality of the infrastructures, a better management of the available resources is less costly. For Intelligent Transportation System, the objective of the network management is to improve the attraction for public transportation with a decrease of the waiting time and an increase of the commercial speed. This improvement has to be done without an increase of the management costs. Efficient planning algorithms and a more precise knowledge of the network state improve the management of the resources. Nevertheless this improvement of the theoretical supply has to go with an improvement of the management of the network in real time. Indeed, the optimum that is the theoretical supply may become obsolete according to the evolution of the urban traffic. Regulators (the staff in charge of monitoring the bus networks) have to ensure the success of the transportation plan, in the sense of adapting the theoretical supply to the real evolution of the demand. In order to achieve their complex task, regulators use systems known as Automatic Vehicle Monitoring systems (AVM). In this paper, we show that if the use of an AVM is the first step to the computerization of the transportation network activity, this system is limited to coping with disturbances linked to unanticipated demands and to traffic conditions. The collecting and shaping of data are insufficient to help regulators and this system has to be completed with a Decision Support System (DSS) able to analyze this information and to give in real time a dynamic and contextual assessment of the problems. Our proposition takes place in the Multi-Agent Decision Support System (MADSS) domain. The DSS should not be based on black boxes ([13]) but on paradigms that are more collaborative and active. By reification of actors and of their use of the information and by distribution of control, a Multi-Agent System (MAS) makes explicit the process it has to manage.

Section 2 describes the real time management of an urban transportation network, the advantages and difficulties of the use of the current information system are underlined. Section 3 presents the alternatives for the integration of a DSS to the current information system. Section 4 describes the SATIR (Système Automatique de Traitement des Incidents en Réseau) project that is our proposition to integrate a DSS to an AVM. Section 5 details our experimentation. Section 6 proposes a conclusion.

*(Produces the copyright information for AAMAS 2008). For use with aamas2008.CLS and aamas2008-short.cls

2. URBAN TRANSPORTATION SYSTEMS: STATE OF THE ART

In this section, the processing of information for the management of an urban bus network in real time is presented. The first part presents the model of data for the urban transportation domain and the information system (AVM) based on it. The second part is related to the regulation task.

2.1 An Automatic Vehicle Monitoring (AVM) System

2.1.1 Data Model

In Europe, the first project for a data model has been CASSIOPE (1989-1992) the result being the Transmodel[1]. This model is the European reference of the conceptual data modeling for public transportation domain. Transmodel has been improved with the European projects Eurobus (1992-94) and Harpist (1995). Finally, the Titan project (1996-98) (Transmodel based Integration of Transport Application and Normalisation) has completed and validated this work. The result is a modeling based on the Entity/Association formalism. The objective is to represent in the same modeling the physical and timetable configuration of a network. A hierarchical decomposition of the information has been adopted. The physical configuration is based on the decomposition of the network in lines and for each of them in routes. A route is decomposed in sections and for each of them there are a stop and an inter-stop distance. The timetable configuration is based on the decomposition of the timetable in missions and for each of them in runs. This modeling has been done to answer to a specific need: to access in the easier way to a specific component of the timetable, like the description of the route of a run or for a section the list of the schedules for a period of time.

2.1.2 Functionalities

In order to use this information modeling, the transportation networks use a specialized information system called Automatic Vehicle Monitoring (AVM). With the theoretical information (network description and timetable), the AVM computes in real time the theoretical state of the network. In order to compute the real state of the network, an AVM uses the data coming from vehicles located by sensors. AVM compares the actual positions of vehicles (captured by the sensors) with their theoretical positions in order to provide the regulator with an overview of the routes. In this way, the regulator can see whether vehicles are running ahead of timetables or are running late. By comparing theoretical information with real one, the AVM system tries to detect delays and advances of buses on the network. Some AVM systems propose detections depending on a geographical condition like delay/advance alarm in a town-centre or depending on a timetable condition like the detection of the delay on the next departure. In real time, an AVM organizes the collecting and shaping of data; it facilitates the access to this data and computes some alarms. The main objective of the AVM is to give a basic information computation. An AVM works according to a classical way for the diagnostic domain: model-based approach relying on normal and faulty behavioral models. This approach is based on the comparison between a theoretical modeling of the diagnosed system and an artefact of the real system. Since the theory of the regulation domain is not complete, the AVM is only able to support this approach and the result is mainly dependant of the experience of the regulators.

[1] http://www.transmodel.org

2.2 The regulation task

This section presents the new functionalities of the AVM to support the work of regulators. We first describe the regulation process and highlights the limits of the AVM system in real-time management of bus network.

2.2.1 The regulation process

Analysis of the work station of network regulators from the Brussels Intercity Transport Company (Société de Transport Intercommunale de Bruxelles - STIB) enabled us to identify four phases required in the regulation process. First the regulator begins by monitoring the network.

The diagnostic phase begins with the detection of a problem and ends with the assessment of its consequences on the network activity. The interface of the AVM facilitates this phase. The current alarms are visually accessible and relieve regulators of computation. As soon as a disturbance is chosen by the regulator, he has to complete his knowledge of the problem. This process is complex because disturbances evolve independently along three axes. The Time axis measures the seriousness of a disturbance according to the timetable. The Space axis measures the seriousness of a disturbance according to its position on the network. The Shape axis measures the consequences a disturbance may have on the network activity. To determine its importance, a disturbance must be evaluated according to these three axes. For example, a vehicle having off-peak hour difficulties in a suburb (a disturbance that is not critical a priori) may cause a real problem if bus frequency is low.

When a disturbance has been detected and assessed, the planning phase begins. The type of the risk gives the primary objective to the regulator: to increase/decrease the supply at one part of the network. He computes the feasible procedures according to the current state of the network. This complex process involves various information sources like the real-time information, the theoretical timetables or the information coming from drivers and from other regulators, etc.

The decision phase is the last phase of the regulation process. There are two parts, firstly the regulator chooses the regulation procedure and secondly he monitors its execution. If the result of the planning phase is a set of feasible procedures, the regulator has to find a compromise between contradictory constraints. For example, the empty runs or the failures of the regulation's procedures have to be limited. After this choice, the regulator has to monitor that the resources related to the procedure and the state of the network evolve according to the forecasting of the regulator, otherwise, he may have to change his choice.

Figure 1 sums up the tasks of the regulator and the links between them. The regulator is involved in all of them. This multiplicity of tasks makes his work very complex. Note that the decision task that is based on the result of the diagnosis and planning tasks should be his main activity.

2.2.2 Regulation process issues

The main advantage of the AVM is to facilitate the access to numerous and heterogeneous information sources, but the

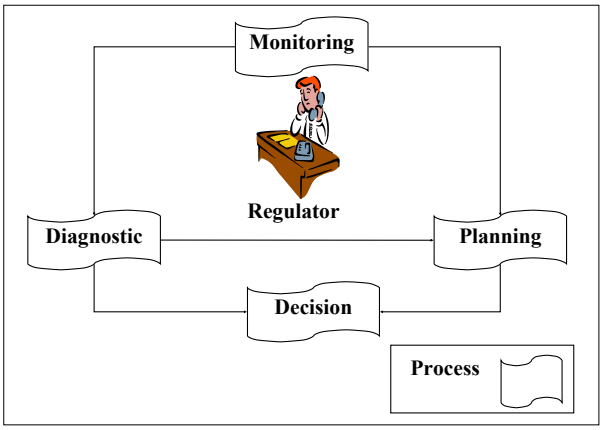


Figure 1: The regulation process in a urban transportation system

regulator has to use his experience to extract the right information at the right time. This information organization implies that the regulation process is strongly dependent on regulator experience.

The first issue is related to the information incompleteness. An AVM gathers information but part of them may be missing according to the success of the collecting process. Moreover, the regulator work is based on completing information as timetables miss the necessary knowledge to compute them. This knowledge is essential for a better management of the transportation plan. The theory of the domain proposes different rules (called logics) for the regulation of a network and they underlie the computation of the timetable. If the description of these logics is not in the scope of this paper, remember that each of them is related to a specific objective. For instance, the logic of regularity aims at minimizing the average waiting time of the travelers and to divide the supply on all the buses in the line. On the contrary, the logic of taking away aims at respond to a local important demand. In this case, the timetable is computed in order to concentrate the resources on the most critical points in the line in order to keep all the passengers. Consequently, in case of a disturbance, the regulation process should not be based on the theoretical state of timetable but on its objectives. This change in the regulation process is only possible for experimented regulators, who find the missing information with their knowledge of the network.

The second issue is related to the distribution of the regulator's attention. Because of the urban traffic, numerous vehicles in the transportation network are late especially at the peak hours. A regulator can not take into account all these delays. An experimented regulator uses his knowledge of the line structure (the position in the city, the presence of difficult areas) and of the demand structure to determine the most critical lines according to the schedule. For instance, information indicating that a vehicle is too late to do its next departure may be useless if this vehicle can make up for the lost time on the last part of the run. The regulator uses the computed alarms as primary indicators and analyzes its real importance according to its context.

The third issue is related to the assessment of a disturbance by the regulator. This task implies that the problem evolution on the network is taken into account. Primary alarms on the advance/delay of each bus provide instantaneous picture of line conditions. Monitoring all these alarms in their space-time development is almost impossible and leads to extra work for regulators. The regulator has to forecast in a time-window the evolution of the situation. This forecasting takes into account the future buses positions in order to estimate the real importance of the problem. The regulator has to choose the information according to its own expertise and the estimate state of the network.

The fourth and last issue is related to the lack of a global vision. The splitting up of monitoring by line and the high number of the lines to be monitored (each regulator tracks 13 lines with 5-20 buses running during the day in the STIB network) prevent global management of the network. Regulators use their expertise to relate disturbances without visible links, in order to propose global solutions.

These issues are based on the lack of information modeling to help a novice regulator. Our proposition is to integrate a Decision Support System into the information processing in order to support regulators task. This evolution from an information system to a DSS corresponds to the evolution of applying computer science in the transportation domain. The first step has been to formalize the domain knowledge in order to create an information system for an efficient data management. The objective of the next step is to automate the use of this data through the design of a DSS.

3. INTEGRATING A DSS AND AN AVM SYSTEMS

The cooperation between a man and a system during a decisional process can be done in two different ways: 1) vertical cooperation; 2) horizontal cooperation. The difference between them depends of the distribution of the control. The first choice is to use the DSS as an information source and the operator is responsible of the decision. The second choice is to have a DSS that computes in parallel to the operator in order to reduce its work. This part discusses the choice of the cooperation type that is the best suitable for the urban transportation regulation domain.

3.1 Choice of the system architecture

The issue is to find the right place of the DSS during the current information process. In a horizontal architecture, the DSS and the AVM compute in loop and the operator is the censor of this information process. From our point of view, the choice of an "autonomous" architecture is premature in the urban network management domain. The information quality and the low formalization of the domain's knowledge are difficulties that are hard to solve. Indeed, in urban environment the information is not sure because their collect is not an easy process. That is the reason why, a regulator spends an important part of his time-work in order to confirm the information coming from the AVM as soon as he has doubts on the correctness of the data. The generalization of the localization technologies like Global Positioning System (GPS) will solve this issue. Nevertheless this technical improvement can not easily replace qualitative information coming from drivers. For example, the management of a disturbance related to a misplaced car is not the same that if the reason of the disturbance is a person accident. An automatic system will detect that the vehicle is late but

it will not be easy to know the reason of this delay.

In a vertical architecture, the DSS is a guide or a server of solutions. The AVM that has been created to facilitate the access to theoretical and real data is a basic guide but it has been shown that this role is not enough to help the regulator. A DSS as a solution server may be based on a simulation tool or it may have access to the information in the AVM. The regulator uses it as a simulation tool according to the *what-if* model. For example, Brezillon ([3]) proposes a DSS to the management of the parisian subway which is activated by the regulator in the case of a disturbance. This DSS proposes solutions according to information provided by the regulator. That means that the regulator filters the data and solves the quality data problem. With this organization of information processing the problem is the difficulty of managing disturbances in real time in a dynamic environment because the regulator has a new task, the information filtering.

In order to solve the problems related to the information management, a solution would be to link the DSS with the data.

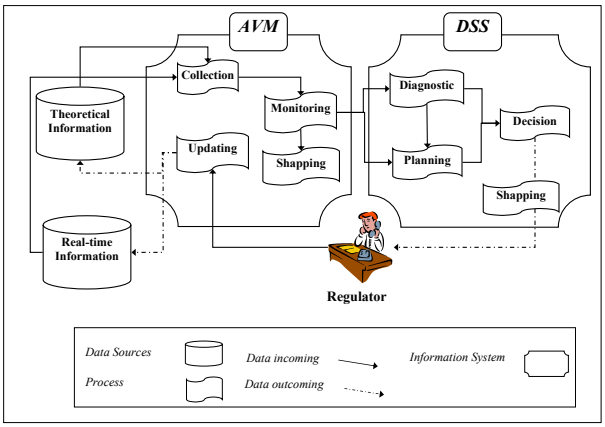


Figure 2: Integration of a DSS and AVM systems

Figure 2 shows an architecture that is a response to the data management problem. The AVM preserves its initial function: the collecting and shaping of data and the DSS access to useful data for its reasoning. The DSS proposes solutions to regulators which evaluate their validity because they access to the two systems. This architecture is based on a vertical organization of the cooperation between operators and systems. The main advantage is to reuse the existing system but it implies implicit information duplication. For example the position of the buses is used by the AVM for the interface and the alarm computation but it is also used by the DSS in order to compute a solution. It will be difficult to propose a modeling of the data qualification. For example, how could the DSS detect that the delay of a bus is due to a problem or to an error in the position collect (a captor is breakdown) without a comparison with the precedent position of the bus or the precedent records of the captor. The last issue is organizational, some AVM already have some of the functionalities that are related to the DSS like the computation of alarms.

The criticism on the positioning of the DSS, AVM and regulator leads to the proposition of a new generation of information system that we call: Transportation Regulation Support System (TRSS). Our proposition includes the functionalities of the AVM system and is based on the multi-agent approach in order to organize data and processing (Figure 3).

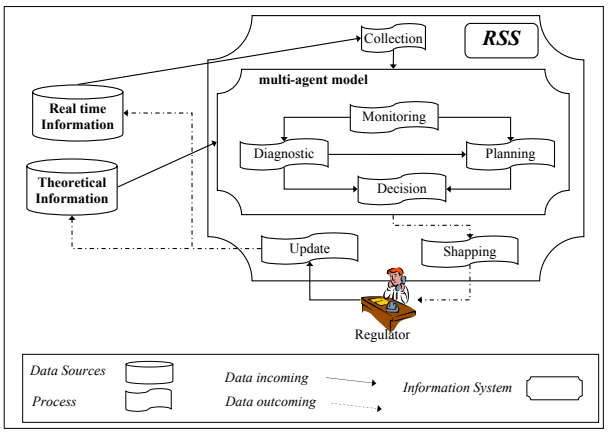


Figure 3: A Transportation Regulation Support System Architecture

The gathering of a DSS and an AVM system solves the problem of the duplication of the data. The same data are used to manage the transportation network under normal operating conditions (monitoring function) and also under disturbed conditions (diagnostic and planning functions). In order to avoid a superposition of complex functions, our proposition is based on the modeling of the dynamic of the data processing by a multi-agent reification of the static hierarchical data model. A multi-agent system has been defined to manage the transportation network under normal conditions (network monitoring, dynamic schedule management, data inconsistencies management) as well as under disturbed conditions ([1]). Next section justifies our choice of the multi-agent paradigm.

3.2 Choice of a multi-agent modeling approach

Several researches in the multi-agent community ([12, 5]) have been done in the transportation domain. In ([5]), the authors underline that 63% of the research are related to the conception of DSS. Mathematical system and Interactive Decision Support System have been important in the modeling of the decision process in the transportation domain but they are often black boxes that hide the decision process. Moreover these systems give synthetic results that have to be analyzed by the regulator in order to become a final diagnostic. These models compute with numerical data and it is not easy for them to take into account qualitative data, like the relative importance of a delay according to the position of the bus in the network. Moreover these models are not suitable to compute with uncertain and incomplete data. Most of all, these models presume that the data is available and reliable but as written before this is a strong hypothesis in the urban transportation network management.

A MAS has a different approach for the conception of a DSS. The first difference is related to the objective of these systems that is the comprehension of the process that they

manage. A Multi-agent system makes easier the comprehension of a complex reality, by the reification of the components of the system to manage. This underlining of the components and of their links facilitates the comprehension of the regulation process that is at the beginning of its formalization. The multi-agent approach paradigm is well adapted to the transportation domain since it facilitates an approach by analogy in a domain where the objective is the management of distributed entities. The second difference is related to the management of quantitative, qualitative and symbolic description that the multi-agent paradigm facilitates. This point is useful to put into perspective the importance of the alarms.

In the domain of public transportation management, the works of ([10, 4]) are based on the use of a simulator. Ossowski *et al.* introduce an organizational and communicative model of decision support environments applied to transportation management. These systems are not integrated and are not directly fed with real-time data coming from vehicle sensors and in that sense are closer to the architecture shown in Figure 2. At last [14] proposes a multi-agent architecture for system related to the management of traffic. This work is mainly related to the issue of the management of entities that are physically distributed rather than to the transportation problem.

3.3 Choice of the environment modeling approach

Recently, the Environment for Multi Agent Systems technical forum group (E4MAS) has produced valuable outputs concerning the poorly exploited concept of the environment, notably in terms of roles, responsibilities, architecture, as well as practical applications, thus opening many challenges in terms of modeling, methodology and engineering [16]. In [16], the authors enumerate the responsibilities of the environment and some of them have already be applied in the transportation domain.

Because the environment is a shared space for the agents, resources and services, its first responsibility is the structuring of the MAS. The environment modeling is a solution to give a space-time referential to a transportation application. Moreover, its privileged intermediary role makes the environment a good candidate to support spatially and temporally decoupled coordination models and thus it simplifies the design of solutions taking into account dynamic real environment. In [15], the environment contains fields that are propagated in the environment in a certain range and used by the agents to organize the task assignment for Automatic Guided Vehicles. The MAS environment becomes the common referential that enables agents to adapt their behavior according to the dynamic of the real environment.

Because the environment has its own process, its second responsibility is to maintain its own dynamic. According to its structuring responsibility, the environment can also manage the dynamic of the transportation environment, ensuring the coherence of the MAS. In [15], the environment ensures the propagation of the fields. Moreover, the environment can ensure services that are not at the agent level or in order to simplify the agent design. In a traffic light control system [2] based on evolutionary game theory, the environment, that has a global point of view, gives rewards or penalties to self-interested agents according to their local decision.

Because the environment with its own dynamic can control the shared space, its third responsibility is to define rules for the multi-agent system. The environment can controls the execution of the MAS ensuring for example that the coordination rules will be respected. In a bus network simulation [9], the main role of environment is to constraint perceptions and interactions of agents. Indeed, a Bus agent and a Traveler agent can interact only when they are located at the same bus stop. For transportation applications that have an incomplete knowledge, it simplifies the design of the MAS by a clear separation between the roles of the agents and their organization. In the coordinated monitoring of traffic jams application [7], the environment provides organizations, which it dynamically evolves according to the current context. Organizational evolution uses a set of laws, which define the way organizations and role (played by agents) should evolve given the current context.

Because the agents are "users" of the services of the environment and to really create a common knowledge, the last responsibility of the environment is to let observable and accessible its own structure. From our point of view, this last responsibility simplifies the reification of the MAS components in the environment and their manipulation by the agents.

4. A TRSS APPLICATION: THE SATIR SYSTEM

4.1 Multi-agent modeling of a transportation network

The agent modeling takes into account our initial proposition that is to gather in the same unit the data and the knowledge useful to its process. For instance the agents not only use the position of the buses but they also detect the inconsistencies. That is the reason why cognitive agents have been chosen. The static modeling of the data is based on a hierarchical organization. The stop is the elementary component of the physical and timetable information; the vehicle is the access point to the real-time information. We also define two types of agents: 1) The STOP agents that represent the theoretical structure of the network and compute the theoretical timetable; 2) The BUS agents that represent the dynamic part of the network. Every BUS agent is the abstract model of an actual vehicle running on the transportation network and reports its movements to the STOP agents.

The interaction between agents in the coordination protocols is based on the exchange of messages. In the domain of urban traffic control, the sender does not always know the name of its receivers because the receiver of the message is often identified according to its position. For example, when a bus has to contact its nearest bus, it does not know its identification. Usually, the simplest protocol is a broadcast protocol (more or less limited). The drawback of this solution is the high communication cost, mainly in real-time systems like urban transportation systems since the location of buses is updated very frequently (every 40 seconds in our application). Another simple solution is the use of acquaintances; the interaction problem is solved by an increase in the interaction knowledge of agents. In a TRSS, this solution is inadequate because the problem remains when an agent is not able to link its needs to an agent identifier. A third

solution is the use of a middle-agent. This approach called, "capability-based coordination", is a preference/capability matching, used to identify the best provider for a given capability search. In our transportation problem this solution has no sense because all STOP agents have the same capability (idem for BUS agents). Because the dyadic interaction solutions are not adapted to the transportation domain, we propose to base our interaction model on the mutual awareness principle. An important part of the interaction in real-life situations comes from other means than direct transmissions [6], and is related to a particular state of the participants: awareness. Although it has long been considered as a passive state, we consider that awareness is an active state and not only the result of stimuli. Work in the fields of psychology and sociology have discussed whether or not there also has to be an active participation of the "perceiver". For example, Heath [8] says that awareness is not only the perceiver's availability to be aware of the environment, but also his ability to "filter relevant information which is of particular significance". Mutual awareness is based on the sharing of interactions. To be efficient, this principle implies that agents share a common communication media. As a consequence, an agent has to find only messages that it is interested in. In the reactive agent community, the environment is already used as a common interaction medium. In the cognitive agent community, we have proposed the EASI model (Environment as Active Support of Interaction) [11], which enables cognitive agents to use the environment to exchange messages. More precisely, it enables an agent to send messages to another agent that is located by the environment, and also enables agents to perceive every message exchanged.

For this purpose, we consider that the environment contains descriptions of messages and agents. The problem is how the agents use these descriptions to locate messages according to the environment state. This implies matching those descriptions and the needs of the agents. We therefore propose to represent all the components of the environment (agents and messages) as entities. Each entity has a Public Layer containing the properties, accessible through the environment. Agents have the ability to put filters in the environment and these filters are logical expressions on properties. When a message is added to the environment, these filters determine by pattern matching whether the agent is interested in it, in which case it will receive it. In this way, the filters enable agents to create their communication space where each filter corresponds to a precise communication need:

- Reception filters: communication is based on a need that is common to two agents. The sender specifies the values of the characteristics searched for in the receiver. This description is matched against a communication filter of the category of the agents contacted.
- Emission filters: communication is based on a need of the sender which doesn't match any expectation of the category of contacted agents. The consequence of this lack of interest among the receivers is the absence of suitable filters which would make it possible for the environment to find the agents concerned. Consequently, the sender must put the appropriate filter in the environment at the same time as it sends its message. An emission filter may require the comparison of the potential receivers and therefore requires the use of predicates. This implies that the filter matches all the potential receivers of a message and then searches for the exact receiver of that message.
- Interception filters: the interception filters are the sensors which allow the agents to receive the messages which are not sent to them but the content of which may be of interest for them. The goal of these filters is to make full use of the environment as a shared work context where every communication is potentially available for all of the participants.

The organization modeling has to take into account the hierarchical organization of the information and the treatment of this information. The hierarchical organization corresponds to the view of the agents according to different abstraction levels. A STOP agent is located on a line and a route. The gathering of the agents according to a line or a route gives the needed hierarchical level. The monitoring of the network under normal condition is based on this organization. In order to isolate in this data set the information related to a disturbance, another organization will gather the agents involved.

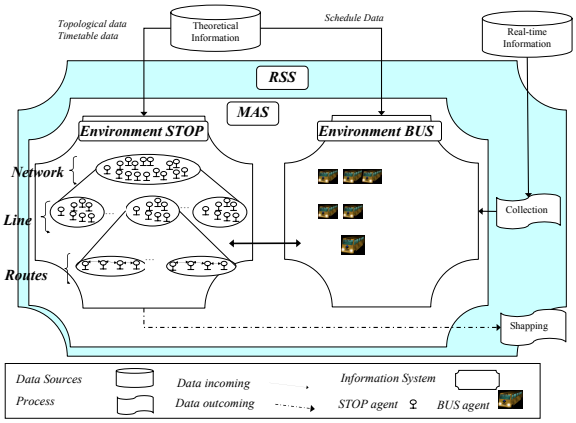


Figure 4: A Transportation Regulation Support System application: the SATIR system

Figure 4 shows the proposed multi-agent model. The agents are gathered in separate environment according to their functionalities. In each of these environment, the communication needs are not the same and the set of communication filters is therefore adapted. The communication is allowed inside and between environments.

At the beginning of the process, the theoretical data are distributed between the STOP agents. The STOP agents have the topological information on their network position: the line, the route, the distance to the next STOP agent (with the identification of this agent) and more particular information like the physical possibility to do a U-turn. A data table is associated with each stop of the network for each period of time (classified from 0 (fluid) to 2 (heavy traffic)). These data are used to define the theoretical state of traffic and of passenger demand and to compute schedules. For instance, at the stop p1 the estimated value of traffic is 1 (the circulation is normal) and the estimated value of flow of passengers is 2 at 8 pm (p1 may be a school). The

dynamic schedule computing replaces the use of a theoretical timetable that not remains a valid reference when the network is disturbed (see 2.2). The STOP agents have the knowledge (traffic problems and passenger flow) used by the graph makers (staff that compute the theoretical supply) in order to establish a theoretical timetable. This knowledge is also used in the assessment process and in the search for solutions to a disturbance.

When a vehicle passes a stop on the real network, a warning message is sent from the BUS agent to the involved STOP agent. The STOP agent updates its timetable by removing this vehicle from its list of expected vehicles. A STOP agent which does not receive any message detects an anomaly and triggers the disturbance processing presented in paragraph 4.2.

In order to only process the important disturbance, the trigger of the disturbance management process depends on the value of a time parameter that may be different for each STOP agent. A default value of 7min has been used (STIB norm). When a vehicle is no longer located, the STOP agents on the bus route have not been informed about the passage of the bus, but they will intercept all new transit announcements sent by vehicles not running to timetable. The interceptor agents receive the message and update their timetable. Overhearing provides an efficient solution to the problem of the inconsistent data, in that it needs few information: the BUS agents have no information on the topology of the routes; and only one warning message is sent for a transit event in the network and the MAS state is updated according to the reaction of agents to this event in their own context.

The BUS agents are the intermediates between theoretical data (STOP agents) and real-time data. This repartition of roles ensures the modularity of our proposition. For example, a modification in the data collects implies a modification of the BUS agents, the rest of the MAS remaining the same. Moreover, this decomposition facilitates the development of more evolved functionalities as diagnostic or planning.

4.2 Multi-agent modeling of the diagnostic process

The STOP agents have knowledge about the theoretical structure of the network and the BUS agents have knowledge about the actual activity of the network and also the theoretical activity of the vehicles (each BUS agent manages its own timetable). We have proposed to put together within a specific organization called the Incident model ([1]) all the STOP agents and BUS agents which are related to a given disturbance. For this purpose, we define three information sets, also called areas:

- *The Successor area*: This area brings together all the stops waiting for the successor of the late bus, it measures the risk assessment of a bus train (the late vehicle is caught up by the following one).
- *The Critical area*: This area brings together all the stops where the vehicle is late, it measures the risk assessment of a gap (the late vehicle is left behind by the preceding bus).
- *The Predecessor area*: This area brings together all the stops where the late vehicle is due but not yet late, it measures the risk assessment of a gap.

The set of these three areas constitutes the Incident model. The Incident model gathers in a single entity all the information necessary to manage a given disturbance thus helping the regulator to do its diagnosis job. The figure 5 pictures three disturbances (c) with their respective areas. For each disturbance, a specific environment is created (a) where the organization of the agents is hierarchic as explain below (b).

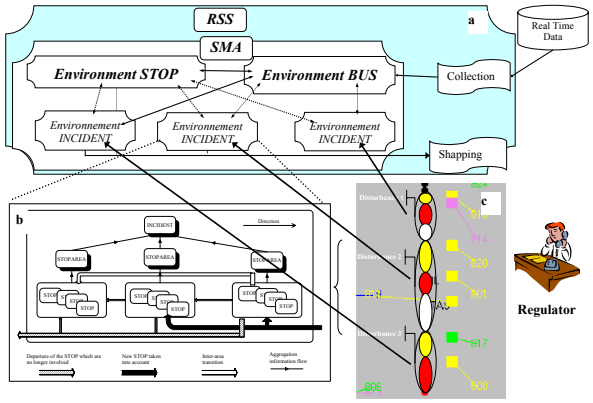


Figure 5: Multi-agent dynamic disturbance modeling in a TRSS

To measure qualitatively the importance of a delay, we have taken into account its consequences on the activity of the network. We have defined two measures of risk linked to a disturbance that are detailed in ([1]). These measures are based on the study of a priori progression difficulties of vehicles involved with the disturbance and take into account the intrinsic dynamics of a disturbance.

The initial organization of the multi-agent system (in lines and routes) is completed with a hierarchical organization of the agents in order to aggregate information and to compute feasible solutions (Figure 5 (b)). At each level of the hierarchy, information is aggregated by the agents. The two new types of agents are the STOPAREA agents and the INCIDENT agent. The lowest level of the hierarchy is composed of the elementary entities, the STOP agents. The middle level is composed of the STOPAREA agents that make an initial summary of the information. They collect basic information such as theoretical traffic evaluation and passenger flow from the STOP agents linked to them and they compute the progression coefficient (an indicator of the problem of the area). The INCIDENT agent represents the top of the hierarchy where feasible solutions are computed and is the link between regulators and the system.

This organization is dynamic because at each cycle, STOP agents move from one area to the other within the hierarchy, and from and towards the outside of the organization, according to traffic direction (Figure 5).

When the disturbance disappears, this organization survives during some cycles to keep the continuity of the disturbance process. After this period, the created agents related to the disturbances disappeared too.

The originality of our approach is the dynamic modeling of a disturbance process from its beginning to its end and its integration in a multi-agent system. We have defined a model, called the Incident model that allows information synthe-

sis that is useful for decision making. Through this model, knowledge relative to the network structure and knowledge relative to the network dynamics (stored in STOP agents and in BUS agents, respectively) are gathered within a single entity that is reified by a specific environment (one environment by disturbance) where all the involved agents are brought together. This entity allows the follow-up of the disturbance over space and time; it is deleted when the disturbance is solved.

4.3 Multi-agent modeling of the planning process

When a disturbance has been detected and assessed, our TDSS computes the feasible regulation procedures. Initially, the transport service matches the theoretical demand to the bus supply. However, when a disturbance appears, there is a discrepancy between the service provided and the passenger flow. Thus, the task of the system is to adjust the initial supply in order to satisfy the needs according to the changes to the network.

4.3.1 Static feasible action planning

Thanks to predefined procedures, regulators modify the transport service according to the state of the network and to the possible actions of the buses on the line. They cancel or modify the vehicle timetable in order to shift the service to another point on the network. One of the original features of SATIR is that BUS agents play the role of regulators, enabling a micro-regulation of the network. At the beginning of its activity, each BUS agent receives the list of the runs it is supposed to do (timetable) and that it may modify dynamically, thus acting as a regulator. When a disturbance is detected, the late BUS agent requests a new run that each BUS agent on the same line tries to insert into its timetable. For each BUS agent, this insertion implies a modification of its timetable. And one or more regulation procedures can be used. Timetable processing within the multi-agent system is implemented in three steps.

- step 1, the availability of the BUS agents is computed in order to eliminate vehicles that are not potential solutions. Using conditions related to its own characteristics (example: its size) or related to network rules (example: the last run of a vehicle is never changed), a vehicle is eliminated.

- step 2, the profiles of the available BUS agents are computed. We call profile the characteristics of a group of BUS agents with the same relative position compared to the late BUS agent (i.e. before, after, same direction,etc.). For each profile, the regulation procedures are the same. For example, the AlightingOnly procedure may be feasible for all BUS agents located before the late BUS agent but it is useless for the BUS agents located after it. Using the BUS profile may limit the number of tests: some procedures may be forbidden or limited to specific profiles.

- step 3, the feasibility of the regulation procedures is computed. Every regulation procedure has constraints that BUS agents must satisfy in order to be considered as feasible. For example, a vehicle cannot make a U-turn if there is no location to do so.

Breaking this timetable processing down into three steps offers several advantages. Since network authorities have their own regulation rules, they do not apply the same constraints on vehicles and on regulation procedures. The three steps described above enable a network to adapt the planning process to its own rules and constraints. Moreover, the distribution of BUS agents into profiles limits the solution space to the only feasible procedures. from a micro-regulation viewpoint, another advantage of this planning process is that it can be distributed and automatically applied by BUS agents.

4.3.2 Dynamic feasible action planning

The adaptation process begins with an inform message sent by an INCIDENT agent to the late BUS agent. Because computation of the regulation procedures depends on its position and on the information related to its current run, the late BUS agent looks for the missing information on the network. Firstly, it sends a message to the STOP agents located between its own position and the end of its current run to collect the missing data, such as passenger flow and run length; secondly it forwards it to the BUS agents that are on the same line by sending a request message. The next step is done by the BUS agents that apply the steps of the planning process described above. Thanks to its local knowledge of the network and of its own activity, a BUS agent may propose feasible regulation procedures. We propose a general model of the regulation procedure as follows (Figure 6).

For each regulation procedure we define three preconditions and a computation process. The preconditions are related to the steps of the planning process and the computation process computes the insertion of a new run requested by the late BUS agent into the BUS agent timetable. Let H be the set of hard preconditions related to the characteristics of the BUS agent (see step 1). Each condition takes into account the internal state of the BUS agent and does not require any additional information to be evaluated. Let P be the conditions related to the profile of the BUS agent (see step 2). The BUS agent computes its own profile given the current disturbance and computes the regulation procedure that is linked to its profile. Let S be the soft conditions that are related to the availability of the procedure (see step 3). If we take the example of the AlightingOnly procedure it is limited to the profiles of BUS agents that are located at the beginning of the line and before the late vehicle. The soft preconditions, defined by the STIB Belgium bus network, are the following ($H = \emptyset$ which means that AlightingOnly is always possible):

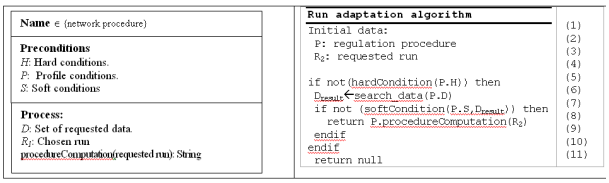


Figure 6: The regulation algorithm

- the distance (number of stops) between the BUS agent and the late BUS agent is less than 5. This procedure means that the passengers of the late vehicle have to wait for the following vehicle and that the waiting time must not be too long.
- the position of the terminus of the BUS agent involved must be superior to the position of the late vehicle terminus. If this is not the case, the procedure will be done only on part of the run of the late vehicle.
- the distance between the late vehicle and its predecessor is greater than 10 stops. Within this procedure the late vehicle will make up lost time. The aim of this precondition is to avoid the creation of a bus train with the previous bus.

In order to compute these conditions, the BUS agent has to look for the missing information. The vehicles charge is not taken into account because the STIB network does not have captors to give this information. For the AlightingOnly procedure, it has to know the position of the predecessor of the late vehicle. The name of this data and that requested for the computation process are recorded in the data set called D (figure 6).

Let R1 be the run that will be modified to insert the requested run. R1 is the current run or a run that is chosen according to its departure time. For the AlightingOnly procedure, the chosen run is the current run of the BUS agent and the adaptation consist of the computation of a new timetable. To compute this timetable, the BUS agent looks for the following data: number of stops, traffic and passenger flow values between the vehicle and the late one. In Figure 6, the algorithm gives the steps of the new run computation for a BUS agent. Only the procedures that belong to the profile of the BUS agent are taken into account. If the hard conditions are false (hardCondition line 5), the BUS agent looks for the missing information (search_Data line 6).

If the soft conditions are true, then the function procedureValidation returns the result of the computation. Using the data on the chosen run R1, the requested run R2 and the collected data D_{result}, this function computes the run-time of the regulation procedure.

To close this process, each BUS agent sends the result of its computation to the INCIDENT agent that gathers and organizes this information for the regulator. In the next section, the result of this process is presented.

5. EXPERIMENTATION AND RESULTS

A prototype has been implemented in C++. In order to study the feasibility of our SATIR system, the prototype was tested using real data recorded every 40 seconds from buses on the Brussels Intercity Transport Company network (STIB). In this section we give some result about the planing process. The result about the assessment process are given in [1]. The data was recorded on tape for around 30 buses, on one line, over 8 days and represents more than 43,000 items. The SATIR system was run over time through cycles on this data representing the movement of buses on the network; it detected 300 incidents and it recorded the disturbances data on file. For each day, the run time is only a few minutes.

In the following example, line number 54 has been studied as follows: 1) Day 1: the monitoring data of the regulator that manages line 54 was recorded. Each of the managed disturbances and the chosen solutions were recorded. 2) Day 2: SATIR was tested with the data related to the line activity of day 1. A serious disturbance managed by both the regulator and SATIR was identified. 3) Day 3: SATIR was shown to the regulators and the solutions to the same disturbance were compared.

The disturbance is the following: a badly parked vehicle blocked bus #54806 that ran more and more behind schedule. The regulator was informed by the driver at 15:33. The regulator chose to call the vehicles near this problem in order to organize a diversion, but there is a technical problem and the driver was unable to get the call. The consequence was a bus train at 15:56:14.

Since the disturbance was detected by the regulator after a call, SATIR did not have this information. As a consequence the disturbance was detected by SATIR after 7 min (which avoids a false alarm), at 15:38. The vehicle was at the stop called ARLON and its STOP agent triggered the disturbance assessment process (Figures 7). Every 3min, the new INCIDENT agent updated the assessment. This disturbance was close to line number 80. The regulator used this proximity to choose a vehicle that was at the end of its runs to substitute for the blocking bus. This new vehicle does not exist in the AVM system and a fictitious reference was created for it. Since vehicle #54806 had a substitute it made a U-turn to make up for lost time. The regulator planned the U-turn where the substitute bus was inserted (called vehicle #1). When the blocking car had gone, vehicle #54806 continued its runs to the stop where vehicle #1 was waiting. At this location, vehicle #54806 made a U-turn and the vehicle continued the run of vehicle #54806. SATIR was not able to find this regulator's solution because this solution implied an external resource (i.e. a new bus). Bearing this in mind, the SATIR planning process detected two BUS agents that could provide a feasible solution, vehicles #54806 and #54827.

Vehicle #54806 has two proposals that take into account the next vehicle (#54830). The common objective is to increase the speed of vehicle #54806 by modifying the run mode. The first feasible procedure is an empty run (M = Without-passengers, Figure 7.A); the second choice is a run with alighting only (M = AlightingOnly, Figure 15.B). These procedures take into account the next vehicle because they are feasible if this vehicle is close enough to pick up the passengers that are not picked up by vehicle #54806. All this information is displayed to the regulators through the interface (Figure 7 (A and B)). For each procedure, several items of data are given: the current delay (7 mn and 4s), that part of the delay that can be made up, how many stop are necessary to make it up. For instance, in the first procedure (M = Without-passengers), 15 stops are needed and in the second one (M = AlightingOnly), 23 stops are needed.

Each regulator proposes solutions according to his own knowledge, habits and experience. For the situation described in this paper, the regulator may propose several solutions that are different to those proposed by SATIR. In our example, the regulator has chosen external resources although there are internal solutions. In order to validate our model, the SATIR proposals were studied by the regulators, who approved them as being feasible. If they did not choose the AlightingOnly procedure chosen by SATIR, this is because it may be efficient for certain lines but not for

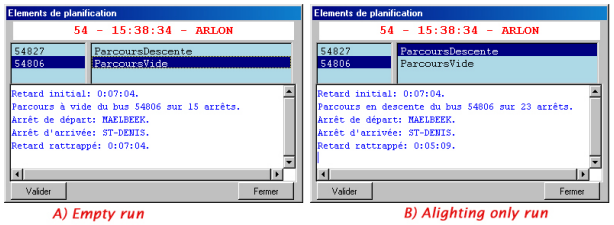


Figure 7: Two SATIR proposals with vehicle #54806 to made up a delay

others (remember that SATIR was tested on only one line and more testing has to be done).

6. CONCLUSION

In this paper, we have presented a Transportation Regulation Support System that represents a global approach to the regulation task on a transportation network. Our proposition is based on a multi-agent modeling of the transportation network. The MAS organization enables to reproduce the functionalities of the existing information system and thus manages the transportation network under normal operating conditions (where are the buses located ?). Under disturbed conditions (where are disturbances (bus delays, bus advances) located ? What action has to be taken to solve the problem?), this organization is completed by a new organization of the agents concerned by a disturbance. The experience gained from the development effort of our system supports the proposition that the multi-agent paradigm is an appropriate framework for transportation network modeling and simulating.

7. REFERENCES

[1] Flavien Balbo and Suzanne Pinson. Dynamic modeling of a disturbance in a multi-agent system for traffic regulation. *International Journal of Decision Support System*, 41 (1):131–146, 2005.

[2] Ana L. C. Bazzan. A distributed approach for coordination of traffic signal agents. *Autonomous Agents and Multiagent Systems*, 10(1):131–164, March 2005.

[3] P. Brezillon, C. Gentile, I. Saker, and M. Secron. Sart : a system for supporting operators with contextual knowledge. In *Proceedings of the Conference on Modeling and Using Context*, Brasil, 1997.

[4] Zhao J. Bukkapatnam S., Dessouky M. distributed architecture for real-time coordination in transit networks. Technical Report metrans project 00-13, 2003.

[5] Paul Davidsson, Larry Henesey, Linda Ramstedt, and all. An analysis of agent-based approaches to transport logistics. In *Transportation Research part C: emerging technologies*, volume 13(4), pages 255–271. Elsevier, 2005.

[6] J. Dugdale, J. Pavard, and B. Soubie. A pragmatic development of a computer simulation of an emergency call center. In *Designing Cooperative Systems: The Use of Theories and Models*, pages 241–256. IOS Press, 2000.

[7] Robrecht Haesevoets, Bar Van Eylen, Danny Weyns, and all. Context-driven dynamic organizations applied to coordinated monitoring of traffic jams. In Danny Weyns, Sven Brueckner, and Yves Demazeau, editors, *Proceedings of Workshop Engineering Environment-Mediated Multiagent Systems*, pages 126–143, 2007.

[8] Christian Heath, Marcus Sanchez Svensson, Jon Hindmarsh, Paul Luff, and Dirk vom Lehn. Configuring awareness. *Comput. Supported Coop. Work*, 11(3):317–347, 2002.

[9] David Meignan, Olivier Simonin, and Abderrafiaa Koukam. Adaptive traffic control with reinforcement learning. In *4th Workshop on Agents in Traffic and Transportation (ATT)*, pages 50–56, 2006.

[10] Sascha Ossowski, Josefa Z. Hernández, and all. Decision support for traffic management based on organisational and communicative multiagent abstractions. In *Transportation Research part C: emerging technologies*, volume 13(4), pages 272–298. Elsevier, 2005.

[11] Julien Saunier and Flavien Balbo. Regulated multi-party communications and context awareness through the environment. *International Journal on Multi-Agent and Grid Systems*, 0(0), 2008. to appear.

[12] R. Schleiffer. Intelligent agents in traffic and transportation. In *Transportation Research part C: emerging technologies (special issue)*, volume 10C 5-6, pages 325–527. Elsevier, 2002.

[13] R. Vahidov and B. Fazlollahi. Pluralistic multi-agent decision support system: a framework and an empirical test. *Information and management*, 41:883–898, 2004.

[14] Fei-Yue Wang. Agent-based control for networked traffic management systems. *IEEE Intelligent Systems*, 20(5):92–96, 2005.

[15] Danny Weyns, Nelis Boucké, and Tom Holvoet. Gradient field-based task assignment in an agv transportation system. In *AAMAS '06: Autonomous agents and multiagent systems*, pages 842–849, New York, NY, USA, 2006. ACM.

[16] Danny Weyns, H. Van Dyke Parunak, Fabien Michel, Tom Holvoet, and Jacques Ferber. Environments for multiagent systems, state-of-the-art and research challenges. *Lecture Notes in Computer Science Series*, 3374:2–52, 2005.

Multi-agent Model Predictive Control of Signaling Split in Urban Traffic Networks*

Lucas Barcelos de Oliveira†
Dept. of Automation and Systems Engineering
Federal University of Santa Catarina
Florianópolis, SC 88040-900, Brazil
lucas@das.ufsc.br

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

ABSTRACT

Urban traffic networks are large, dynamic systems which remain a challenge in control engineering despite all of the scientific and technological progress. The sheer size, wide spread of sensors and control devices, and nonlinearities make such systems complex, beyond the scope of existing models, let alone control algorithms. To this end, control engineers have looked for unconventional means for modeling and control, in particular the technology of multi-agent systems whose appeal stems from their composite nature, flexibility, and scalability. This paper contributes to this evolving technology by proposing a framework for multi-agent control of linear, dynamic systems. The framework decomposes a centralized model predictive control problem into a network of coupled, but small sub-problems that are iteratively solved by the distributed agents. Theoretical results ensure convergence of the distributed iterations to a globally optimal solution. The framework is applied to the signaling split control of traffic networks. Experiments conducted with simulation software indicate that the multi-agent framework attains performance comparable to conventional control, such as the traffic-responsive urban control strategy.

Categories and Subject Descriptors

F.2.1 [**Analysis of Algorithms and Problem Complexity**]: Numerical Algorithms and Problems; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence; J.6 [**Computer-aided Engineering**]

Keywords

Urban traffic networks, split control, distributed agents, model predictive control

1. INTRODUCTION

Much of the improvements in urban traffic control can be attributed to past advances in science and technology. The existing technology is changing the way traffic systems are designed and operated. Today, modern operating centers receive traffic-flow data from distributed sensors and implement control policies in response to the prevailing traffic conditions. Yet, there is room for further improvements in urban traffic control to cope with the increasing volume of traffic which incurs pollution, excessive fuel consumption, and prolonged journey times.

Over the last few years, the **T**raffic-responsive **U**rban **C**ontrol (TUC) strategy has drawn attention for its robustness and good performance, specially so under saturated traffic conditions [11]. Such results have been corroborated in field applications in cities as Munich, Glasgow, Southampton, and Chania [3, 12, 16]. The TUC framework models traffic flow using a variation of the store-and-forward model originally proposed in [13], which uses purely continuous state and control variables allowing the computation of control policies with efficient algorithms. In its standard form, TUC calculates the control signals with a two-stage multi-variable regulator [11]: the first stage solves an unconstrained linear-quadratic-regulator (LQR) problem that minimizes a quadratic function on queue lengths and control signals; the second stage recovers feasibility of the control signals produced by LQR, whereby an optimization problem is solved to minimize the distance of the infeasible solution to the feasible space. However, such two-stage procedure does not guarantee optimality [4]. To this end, a model predictive control (MPC) approach was proposed to explictly handle the constraints, this way guaranteeing control feasibility and improving solution quality [9].

Alongside the progress on traffic-flow modeling and control, a great deal of research has advanced the technology of multi-agent systems, notably in the fields of artificial intelligence and software engineering [15, 18]. This evolving technology seeks to assemble agents of limited knowledge and abilities in a multi-agent organization to perform tasks that are beyond the expertise of its individual members. Such agents not only encapsulate information, but they also exhibit semi-autonomous behavior by employing some form of reasoning to cooperate with others for the interest of the whole organization, negotiate to resolve conflicts, and even compete when driven by self-interest. The problem-solving ability of a multi-agent system emerges from the interactions and collective effort of the agents, not only their intelligent behavior.

Multi-agent frameworks were originally restricted to the field of computer science where typical applications are of discrete nature such as puzzle solving, planning, and combinatorial arrangement. More recently, control engineers realized that such frameworks can be extended to operate dynamic systems, specially complex distributed systems such as petrochemical plants and transportation networks [19, 22, 23]. The operation of such complex, spatially-distributed dynamic systems is a formidable challenge to control engineering, to a great extent due to the intrinsic complexity, sheer size, and nonlinearities. Control engineers have turned their atten-

*This research was supported in part by Conselho Nacional de Pesquisa e Desenvolvimento Tecnológico (CNPq) under grants 551050/2005-5 and 473841/2007-0.
†Supported by CNPq Fellowship Program.

tion to multi-agent systems whose appeal stems from their composite nature, flexibility, and scalability [24].

However, multi-agent systems are still a long way from delivering this promise to complex dynamic systems. Much of the literature offers methodologies, general guidelines, or otherwise ad hoc procedures lacking formal methods that ensure convergence and stability. To this end, this paper proposes a framework for controlling linear dynamic systems with a network of distributed control agents. These dynamic systems arise from the interconnection of linear sub-systems with local input constraints. Applications are found in signaling split control in traffic networks and reaction control in petrochemical plants. Given dynamic equations and algebraic constraints, our framework formulates the optimization problem arising from model predictive control and proceeds to decompose the MPC problem into a network of coupled, but small sub-problems to be solved by the agent network. An agent senses only the state variables and sets the values of the control variables of its sub-system, communicating with agents in the vicinity to obtain the values of neighborhood variables and coordinate their actions. With a well-crafted problem decomposition and coordination protocol, the solution iterates produced by the agents can be shown to converge to a globally optimal solution to the MPC problem.

In essence, the decision-making and cooperative control behavior of the agents emerges from the solution of optimization problems. The work reported here builds upon preceding work on distributed control [6, 8] by exploiting the linear dynamic structure to develop simpler models and algorithms.

In a representative traffic network, computational experiments are conducted to assess the performance of the proposed multi-agent MPC framework. The purpose of the experiments is twofold. First, the experiments compare a single, centralized agent with a network of distributed control agents at solving the MPC problem for a number of initial conditions. Second, they compare the multi-agent approach with the TUC strategy using metrics provided by a professional simulation package.

The remaining sections are structured as follows. Section 2 offers some basic concepts about urban traffic networks, along with a description of the store-and-forward model of traffic flow and the LQR strategy used by the TUC approach. Section 3 formulates the MPC problem for split control as a linear dynamic system consisting of a network of dynamically coupled sub-systems, one for each intersection. Last but not least, the section develops a perfect decomposition of the MPC problem into a network of sub-problems and outlines a distributed algorithm for the agent network, which can be shown to converge to an optimal solution. Section 4 reports results from numerical analyses designed to compare the centralized and distributed solution of the MPC problem and from simulated experiments aimed to compare the TUC LQR strategy with the multi-agent MPC approach. Section 4 makes some final remarks and suggests directions for future research.

2. URBAN TRAFFIC CONTROL

An **U**rban **T**raffic **N**etwork (UTN) comprises a set of roads, arterials and streets, known as *links*, interconnected by *junctions* that may be controlled [11]. The traffic inside the network is divided into *streams* of vehicles. Streams grouped in a same link define its *saturation flow*, which is the mean flow crossing the stop line of an approach when the respective stream has the right of way (r.o.w.), a sufficiently large upstream queue, and unobstructed downstream links. The repeated sequence of signal combinations at a junction is named *signal cycle*. Its duration is called *cycle time* or simply *cycle*. A *stage*, or *phase*, is a portion of a signal cycle in which a set of streams has the r.o.w. For safety measures, stages are interposed by constant *lost times* of a few seconds avoiding interference amongst conflicting streams (Fig. 1).

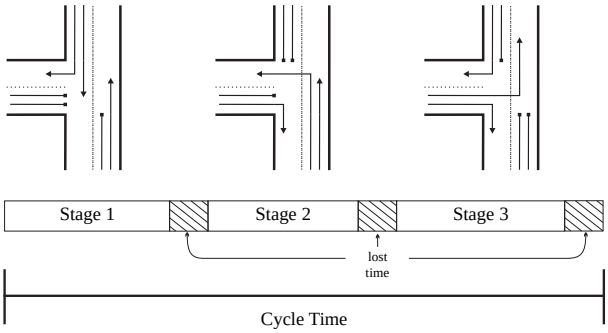


Figure 1: Signal cycle, lost time and cycle.

The influence of traffic lights on traffic depends on four factors [10, 20]: stage specification, cycle duration, offset among junctions, and signaling split. Where split refers to the relative green percentage of the cycle time assigned to each stage.

2.1 UTN Modeling

A UTN modeled in accordance with the TUC strategy [11, 12] is represented as a directed graph with links $z \in Z$ and junctions $j \in J$. Sets I_j and O_j denote, respectively, the incoming and outgoing links of junction j. Cycle times C_j, lost times L_j, turning rates $t_{z,w}$, $z \in I_j$, $w \in O_j$, and saturation flows S_z, $z \in I_j$, are considered constant and known. For the sake of simplicity, $C = C_j$ is assumed for all junctions $j \in J$. Finally, the control signal of junction j has a fixed number of stages belonging to the set F_j, where subset $v_z \subseteq F_j$ represents those where link z has the r.o.w.

Letting $u_{j,i}$ denote the green time of phase i at junction j, the constraint $\sum_{i \in F_j} u_{j,i} + L_j = C$ must be enforced. Additionally, $u_{j,i} \in [u_{j,i}^{min}, u_{j,i}^{max}]$ where $u_{j,i}^{min}$ and $u_{j,i}^{max}$ are the minimum and maximum allowable green times, respectively.

The main differential of this strategy is the use of a variation of the store-and-forward model, where the control cycle is required to be greater than every cycle of the network. Therefore traffic flow is modeled as purely continuous, allowing the use of efficient algorithms on the control signal computation.

The dynamics of network link z is given by equation:

$$\Delta x_z(k+1) = T[q_z(k) + d_z(k) - f_z(k) - s_z(k)], \qquad (1)$$

where: x_z denotes the number of vehicles in link z; q_z and f_z are, respectively, the inflow and outflow of link z during period $[kT, (k+1)T]$, where $k = 1, 2, \ldots$ is a discrete time index and T is the control interval; d_z is the demand, vehicles entering the network not originating from adjacent links; and, finally, s_z is the exit flow at time k.

Since exit rates are known, the exit flow may be replaced for the following equality: $s_z(k) = t_{z,0} q_z(k)$. In addition, one may formulate the inflow of link z as $q_z(k) = \sum_{w \in I_j} t_{w,z} f_w(k)$, where $t_{w,z}$ is the turning rate towards link $z \in O_j$ coming from link $w \in I_j$. Assuming that inflows and outflows of link z with r.o.w. are equal to their saturation flow, S_z, equation (1) is written as:

$$x_z(k+1) = x_z(k) + T\left[d_z(k) - \frac{S_z}{C}\sum_{i \in v_z} u_{j',i}(k) + (1 - t_{z,0}) \sum_{w \in I_j} \frac{t_{w,z} S_w}{C} \sum_{i \in v_w} u_{j,i}(k)\right], \qquad (2)$$

where the control signal $u_{j,i}(k)$ is the green time for vehicles going through junction j during phase i, whereas $\sum_{i \in v_z} u_{j',i}(k)$ is the green time for vehicles leaving link z. Notice that z leaves junction j and enters j'. Generalizing equation (2) for all network links leads to the matrix equation:

$$\mathbf{x}(k+1) = A\mathbf{x}(k) + B\mathbf{u}(k) + T\mathbf{d}(k), \tag{3}$$

where: $\mathbf{x}(k)$ is the state vector; $\mathbf{u}(k)$ is the control vector containing signals $u_{j,i}$, $\forall i \in F_j$, $\forall j \in J$; $\mathbf{d}(k)$ is the vector containing demands d_z, $\forall z \in Z$; and $A = I$, B, and T are the state, input, and disturbance matrices, respectively.

2.2 Split Control

In a traffic-responsive control strategy the signaling split must be optimized according to the demands of involved streams. In standard form, the TUC strategy uses the LQR theory to find an efficient time-invariant gain matrix, which is simpler than optimizing a physical criterion [11] but invariably achieving a sub-optimal control law. By assuming $\mathbf{d}(k) = 0$, the dynamic system (3) becomes:

$$\mathbf{x}(k+1) = A\mathbf{x}(k) + B\mathbf{u}(k), \tag{4}$$

allowing the application of the LQR methodology. The control law thereof does not account for feedforward terms, which is plausible since the main goal is to attain a satisfactory gain matrix rather than an optimized criterion.

Intending to minimize the risk of oversaturation and spillback, minimization of proportional occupancy of links is attempted, i.e. x_z/x_z^{max}, where x_z^{max} is the capacity of link z. A quadratic criterion to this end has the form:

$$\mathcal{J} = \frac{1}{2}\sum_{k=0}^{\infty}(\|\mathbf{x}(k)\|_Q^2 + \|\mathbf{u}(k)\|_R^2), \tag{5}$$

where Q and R are diagonal positive weighting matrices, respectively semi-definite and definite. According to the LQR theory, an infinite time horizon is used in (5) to achieve a time-invariant control law. As matrix Q weighs the states, that is, the number of vehicles in the roads, the goal of minimizing the average occupancy is obtained by making its diagonal elements equal to $1/(x_z^{max})^2$, for the corresponding link $z \in Z$. Matrix R reflects the penalty imposed on control effort, usually defined as $R = rI$, where r is found experimentally.

Minimizing criterion (5) leads to the control law:

$$\mathbf{u}(k) = \mathbf{u}^N - L\mathbf{x}(k), \tag{6}$$

where: $\mathbf{u}(k)$ is the vector with green times $u_{j,i}, \forall j \in J, \forall i \in F_j$; $\mathbf{u}^N$ is the matching vector containing the nominal green times; and L is Ricatti's gain matrix, depending on A, B, Q, and R, though with small susceptibility to their variation [11].

As control constraints are not considered in the aforementioned control law, they are imposed in an ad hoc manner, through the following optimization problem for each junction $j \in J$:

$$\min_{U_{j,i}} \quad \sum_{i \in F_j} (u_{j,i} - U_{j,i})^2 \tag{7a}$$

s. to:

$$\sum_{i \in F_j} U_{j,i} + L_j = C_j \tag{7b}$$

$$U_{j,i} \in [u_{j,i}^{min}, u_{j,i}^{max}],\ \forall i \in F_j, \tag{7c}$$

where $U_{j,i}$ is the closest feasible solution in Euclidean space to $u_{j,i}$.

This problem is solved in real-time for each junction j with an efficient algorithm [10], whose convergence is guaranteed in a number of steps less than or equal to the number of stages $|F_j|$ of the junction. Though this approach gives a feasible solution, it does not satisfy the optimality conditions for system (4). Additionally, because no predictions are made, the multivariable regulator behaves in a purely reactive way to unknown disturbances. On the other hand, the structure of matrix L provides the regulator a gating effect preventing oversaturation in downstream links.

Previously published works [1, 9] report that significant improvement may arise from the replacement of the usual LQR procedure with a solution that accounts for system constraints, such as a model predictive control strategy. Generally speaking, a model predictive control approach is composed by [4, 17]:

- a *prediction model* describing satisfactorily the process dynamics in a finite time horizon;
- a *cost function* which gives the control signal when minimized; and
- a *sliding horizon* of prediction and control, which is translated a step forward at each sample period, requiring the computation of new control actions from which only that of the actual time is implemented.

Following these premises the MPC problem for split control is cast as:

$$P: \min \quad \sum_{k=1}^{T} \frac{1}{2}[\mathbf{x}(k)^T Q\mathbf{x}(k) + \mathbf{u}(k-1)^T R\mathbf{u}(k-1)] \tag{8a}$$

s. to: $\forall k \in \mathcal{T}$:

$$\mathbf{x}(k+1) = A\mathbf{x}(k) + B\mathbf{u}(k) \tag{8b}$$

$$C\mathbf{u}(k) \geq \mathbf{c} \tag{8c}$$

$$D\mathbf{u}(k) = \mathbf{d} \tag{8d}$$

where: $\mathbf{x}(k)$ is the system's state and $\mathbf{u}(k)$ the control input at time k; Q is positive semi-definite and R positive definite weighting matrices; C and $\mathbf{c}$ define the inequality constraints; D and $\mathbf{d}$ define the equalities; and $\mathcal{T} = \{0, \ldots, T-1\}$ is the time horizon.

3. MULTI-AGENT MPC

This section develops an MPC formulation for systems consisting of the interconnection of linear dynamic sub-systems with local constraints, hereafter called linear dynamic networks. The split control problem is cast as an MPC problem over a linear dynamic network, where sub-systems correspond to intersections, state variables correspond to vehicle queues, and control variables represent green times. After the problem formulation, the section presents a decomposition of the MPC problem into a network of coupled, but small sub-problems that are solved iteratively by the agent network. This section reports theoretical properties of the decomposition, relating the MPC problem and sub-problem network, and outlines a distributed protocol to synchronize agent iterations. Conditions are given for the iterations of the agents to arrive at a solution to the centralized MPC problem.

3.1 Modeling and MPC Formulation

The dynamic representation of the traffic-flow derived from the store-and-forward modeling approach is conveniently represented as a system of M interconnected sub-systems, one for each junction. Sub-system m's local state is $\mathbf{x}_m \in \mathbb{R}^{n_m}$ and control signal is $\mathbf{u}_m \in \mathbb{R}^{p_m}$. A directed graph $G = (V, E)$ models the couplings among the sub-systems: an arc $(i, j) \in E$ means that the

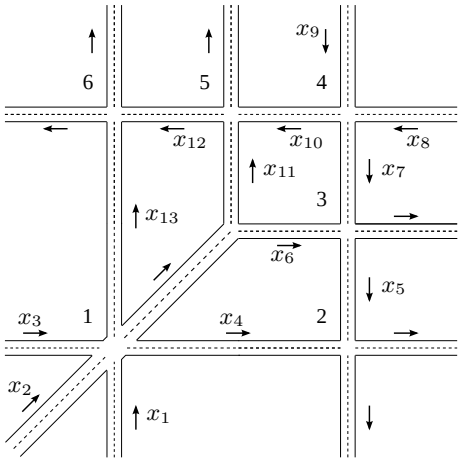


Figure 2: Traffic network.

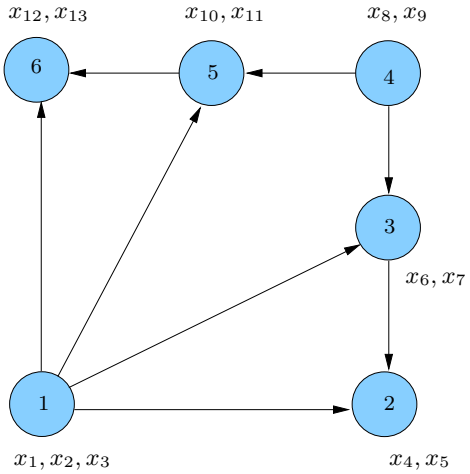


Figure 3: Dynamic coupling graph.

control signals from sub-system i influence the state of sub-system j directly. Assuming discrete-time dynamics, the state equation for sub-system m is:

$$\mathbf{x}_m(k+1) = A_m \mathbf{x}_m(k) + \sum_{i \in I(m)} B_{mi} \mathbf{u}_i(k) \tag{9}$$

where $I(m) = \{m\} \cup \{i : (i,m) \in E\}$ is the set of *input neighbors* of sub-system m including m, that is, the sub-systems affecting the state of m. Given the current state of the network, $\mathbf{x}(0)$, a centralized agent following the MPC strategy would solve the problem below at each sample instant:

$$P: \min \quad \sum_{m=1}^{M} \sum_{k=1}^{T} \frac{1}{2} [\mathbf{x}_m(k)^T Q_m \mathbf{x}_m(k) + \mathbf{u}_m(k-1)^T R_m \mathbf{u}_m(k-1)] \tag{10a}$$

$$\text{s. to: } \forall m \in \mathcal{M},\ k \in \mathcal{T}:$$

$$\mathbf{x}_m(k+1) = A_m \mathbf{x}_m(k) + \sum_{i \in I(m)} B_{mi} \mathbf{u}_i(k) \tag{10b}$$

$$C_m \mathbf{u}_m(k) \geq \mathbf{c}_m \tag{10c}$$

$$D_m \mathbf{u}_m(k) = \mathbf{d}_m \tag{10d}$$

where: $\mathbf{x}_m(k)$ is the state of sub-system m at time k and $\mathbf{u}_m(k)$ is its control input; Q_m is positive semi-definite and R_m is positive definite; C_m and $\mathbf{c}_m$ define the inequality constraints; D_m and $\mathbf{d}_m$ define the equalities; and $\mathcal{M} = \{1, \ldots, M\}$ is the set with the indices of the sub-systems.

The test bed is the traffic network depicted in Fig. 2 with 13 one-way roads and 6 junctions. Sub-system 3 has state $\mathbf{x}_3 = (x_6, x_7)$ with the number of vehicles in roads 6 and 7, while the control vector is $\mathbf{u}_3 = (u_6, u_7)$ with the green time for each road. The coupling graph G appears in Fig. 3. The set of input neighbors to sub-system 3 is $I(3) = \{1, 3, 4\}$. Matrix B_{33} expresses the discharge of queue $\mathbf{x}_3$ as a function of green times $\mathbf{u}_3$, while B_{31} (B_{34}) expresses how queue $\mathbf{x}_3$ builds up as $\mathbf{x}_1$ ($\mathbf{x}_4$) is emptied. The inequality constraints impose minimum and maximum green times on the phases. The equalities state that the total green time plus lost time (yellow time) must add up to cycle time. This is a rough explanation of the store-and-forward model proposed in [11, 21].

Notice that sub-system m's state at time k is a function of initial state and control signals prior to time k:

$$\mathbf{x}_m(k) = A_m^k \mathbf{x}_m(0) + \sum_{l=1}^{k} \sum_{i \in I(m)} A_m^{l-1} B_{mi} \mathbf{u}_i(k-l)$$

By using this relation, collecting the control variables in vector $\bar{\mathbf{u}}_m = (\bar{\mathbf{u}}_m(0), \ldots, \bar{\mathbf{u}}_m(T-1))$, and dropping the constant term from the objective [5], P becomes:

$$P: \ \min \quad f(\bar{\mathbf{u}}) = \frac{1}{2} \sum_{m \in \mathcal{M}} \sum_{i \in I(m)} \sum_{j \in I(m)} \bar{\mathbf{u}}_i^T H_{mij} \bar{\mathbf{u}}_j + \sum_{m \in \mathcal{M}} \sum_{i \in I(m)} \mathbf{g}_{mi}^T \bar{\mathbf{u}}_i \tag{11a}$$

$$\text{s. to: } \bar{C}_m \bar{\mathbf{u}}_m \geq \bar{\mathbf{c}}_m,\ m \in \mathcal{M} \tag{11b}$$

$$\bar{D}_m \bar{\mathbf{u}}_m = \bar{\mathbf{d}}_m,\ m \in \mathcal{M} \tag{11c}$$

where H_{mij}, $\bar{C}_m$, and $\bar{D}_m$ are suitable matrices and $\mathbf{g}_{mi}$, $\bar{\mathbf{c}}_m$, and $\bar{\mathbf{d}}_m$ are suitable vectors. Here, the issue is how a network of distributed agents solves P instead of a centralized agent. In what follows, we develop a decomposition of P into a set of coupled sub-problems $\{P_m\}$ and outline a distributed solution protocol.

3.2 Multi-agent Distributed Control

In our framework for multi-agent control, an agent m decides upon the values of $\bar{\mathbf{u}}_m$ to control sub-system m. For the problem decomposition to be perfect, each agent m solves a local optimization problem P_m encompassing all the terms of f and constraints that depend on $\bar{\mathbf{u}}_m$. Let:

- $\bar{I}(m) = \{i : m \in I(i),\ i \neq m\}$ be the set of *output neighbors* of sub-system m;
- $C(m) = \{(i,j) \in I(m) \times I(m) : i = m \text{ or } j = m\}$ be the sub-system pairs of quadratic terms in Φ_m that depend on $\bar{\mathbf{u}}_m$;
- $C(m,k) = \{(i,j) \in I(k) \times I(k) : i = m \text{ or } j = m\}$ be the pairs of quadratic terms in Φ_k, $k \in \bar{I}(m)$, that depend on $\bar{\mathbf{u}}_m$.

In the traffic network, $I(1) = \{1\}$, $\bar{I}(1) = \{2, 3, 5, 6\}$, $C(1) = \{(1,1)\}$, and $C(1,3) = \{(1,3), (1,4), (1,1), (3,1), (4,1)\}$. Notice that $\bar{\mathbf{u}}_m$ appears in sub-systems $i \in I(m) \cup \bar{I}(m)$, but can

be coupled to other sub-systems—sub-system 1 is coupled to sub-system 4 via sub-system 3, but $4 \notin I(1) \cup \bar{I}(1)$. The notion of neighborhood will establish the interdependence among sub-systems. Models and algorithms for imperfect problem decomposition are found in [7]. According to agent m's view of the system, the control variables are divided in three sets:

- *local variables:* the variables in vector $\bar{\mathbf{u}}_m$;
- *neighborhood variables:* all the variables in vector $\bar{\mathbf{y}}_m = (\bar{\mathbf{u}}_i : i \in N(m))$ where $N(m) = I(m) \cup \{i : (i,j) \in C(m,k), k \in \bar{I}(m)\} - \{m\}$ is the neighborhood of agent m. Notice that $\bar{I}(m) \subseteq N(m)$.
- *remote variables:* the other variables which consist of vector $\bar{\mathbf{z}}_m = (\bar{\mathbf{u}}_i : i \notin N(m) \cup \{m\})$.

According to the perfect decomposition, $P_m(\bar{\mathbf{y}}_m)$ is obtained from P by i) discarding from the objective f the terms not involving $\bar{\mathbf{u}}_m$ and ii) dropping the constraints not associated with agent m. More formally, agent m's local problem is:

$$P_m(\bar{\mathbf{y}}_m): \quad \min \quad f_m = \frac{1}{2}\bar{\mathbf{u}}_m^T H_m \bar{\mathbf{u}}_m + \mathbf{g}_m^T \bar{\mathbf{u}}_m \tag{12a}$$

$$\text{s. to:} \quad \bar{C}_m \bar{\mathbf{u}}_m \geq \bar{\mathbf{c}}_m \tag{12b}$$

$$\bar{D}_m \bar{\mathbf{u}}_m = \bar{\mathbf{d}}_m \tag{12c}$$

where H_m is a suitable matrix and $\mathbf{g}_m$ is a vector. For each agent m, the perfect decomposition ensures that:

$$f(\bar{\mathbf{u}}) = f_m(\bar{\mathbf{u}}_m, \bar{\mathbf{y}}_m) + \bar{f}_m(\bar{\mathbf{y}}_m, \bar{\mathbf{z}}_m)$$

for a given function $\bar{f}_m$. Hereafter $\{P_m(\bar{\mathbf{y}}_m)\}$ will denote the set of sub-problems for all $m \in \mathcal{M}$.

3.2.1 Properties

Below, we report some properties relating P and $\{P_m(\bar{\mathbf{y}}_m)\}$ which are useful to design a distributed algorithm for the agent network. Demonstrations and illustrations are found in [5].

PROPOSITION 1. *A solution $\bar{\mathbf{u}}$ satisfies first-order optimality (KKT) conditions for P if, and only if, $(\bar{\mathbf{u}}_m, \bar{\mathbf{y}}_m)$ satisfies KKT conditions for $P_m(\bar{\mathbf{y}}_m)$ for each $m \in \mathcal{M}$.*

DEFINITION 1. (Feasible Spaces) *The feasible spaces are:*

- *$U_m = \{\bar{\mathbf{u}}_m : \bar{C}_m \bar{\mathbf{u}}_m \geq \bar{\mathbf{c}}_m,\ \bar{D}_m \bar{\mathbf{u}}_m = \bar{\mathbf{d}}_m\}$ is the feasible space for $P_m(\bar{\mathbf{y}}_m)$;*
- *$U = U_1 \times \cdots \times U_M$ is the feasible space for P; and*
- *$Y_m = \times_{i \in N(m)} U_i$ is the feasible space for the neighborhood variables of agent m.*

ASSUMPTION 1. (Compactness) *The feasible space, U, is a compact set.*

ASSUMPTION 2. (Strict Feasibility) *There exists $\bar{\mathbf{u}} \in U$ such that $\bar{C}_m \bar{\mathbf{u}}_m > \bar{\mathbf{c}}_m$ and $\bar{D}_m \bar{\mathbf{u}}_m = \bar{\mathbf{d}}_m$ for all $m \in \mathcal{M}$.*

Compactness is a plausible assumption since control signals are invariably bounded. So is the strict feasibility assumption: if the interior of U is empty, then some inequalities are indeed equalities and should be regarded as such.

PROPOSITION 2. *Problem P given by* (11a)–(11c) *is convex.*

COROLLARY 1. *Sub-problem $P_m(\bar{\mathbf{y}}_m)$ is convex.*

PROPOSITION 3. (Optimality Conditions) *[2] Because f is a convex function and U is a convex set, $\bar{\mathbf{u}}^\star$ is a local minimum for f over U if and only if:*

$$\nabla f(\bar{\mathbf{u}}^\star)^T (\bar{\mathbf{u}} - \bar{\mathbf{u}}^\star) \geq 0,\ \forall \bar{\mathbf{u}} \in U \tag{13}$$

A point $\bar{\mathbf{u}}^\star$ satisfying condition (13) is called *stationary point*.

COROLLARY 2. (Local Optimality Conditions) *$\bar{\mathbf{u}}^*$ is a local minimum for P if, and only if, $(\bar{\mathbf{u}}^*_m, \bar{\mathbf{y}}^*_m)$ is a local minimum for $P_m(\bar{\mathbf{y}}^\star_m)$ for all $m \in \mathcal{M}$.*

A control vector that cannot be improved unilaterally by a single agent, a fixed point, is locally optimal for all the sub-problems and therefore optimal for P.

3.2.2 Distributed Agent Solution

In what follows, we outline a distributed algorithm for the agent network to arrive at a stationary solution to $\{P_m\}$. The agents follow an iterative protocol whereby $\bar{\mathbf{u}}^{(k)} = (\bar{\mathbf{u}}_1^{(k)}, \ldots, \bar{\mathbf{u}}_M^{(k)})$ denotes the solution at iteration k. Starting with a feasible control vector $\bar{\mathbf{u}}^{(0)}$, the agents exchange information locally, synchronize their computations to preclude coupled agents from acting simultaneously, and iterate until convergence is attained.

ASSUMPTION 3. (Synchronous Work) *If an agent m revises its decisions at iteration k, then:*

(i) agent m uses $\bar{\mathbf{y}}_m^{(k)} = (\bar{\mathbf{u}}_i^{(k)} : i \in N(m))$ to obtain an approximate solution to $P_m(\bar{\mathbf{y}}_m^{(k)})$ which becomes $\bar{\mathbf{u}}_m^{(k+1)}$;

(ii) all the agents in the neighborhood of agent m keep their decisions at iteration k, that is, $\bar{\mathbf{u}}_i^{(k+1)} = \bar{\mathbf{u}}_i^{(k)}$ for all $i \in N(m)$.

ASSUMPTION 4. (Continuous Work) *If $\bar{\mathbf{u}}^{(k)}$ is not a stationary point for all problems in $\{P_m\}$, then at least one agent m changes its decisions from $\bar{\mathbf{u}}_m^{(k)}$ to $\bar{\mathbf{u}}_m^{(k+1)}$ by approximately solving $P_m(\bar{\mathbf{y}}_m^{(k)})$ such that $\bar{\mathbf{u}}_m^{(k)}$ is not a stationary point to P_m.*

Condition (ii) of Assumption 3 and Assumption 4 are ensured if the agents iterate in a sequence $\langle S_1, \ldots, S_r \rangle$ where $S_i \subseteq \mathcal{M}$, $\cup_{i=1}^r S_i = \mathcal{M}$, and all distinct pairs $m, n \in S_i$ are non-neighbors for all i. $\langle S_1, S_2, S_3 \rangle$ is such a sequence for the illustrative scenario with $S_1 = \{2, 4, 6\}$, $S_2 = \{3, 5\}$, and $S_3 = \{1\}$. Time-varying sequences and synchronization protocols are alternatives.

Another key issue is how an agent m solves P_m approximately, as stated in condition (i) of Assumption 3, so that $\bar{\mathbf{u}}^{(k)}$ converges to a stationary point for $\{P_m\}$. To this end, we developed an algorithm based on the feasible direction method, which is fully developed in [5] and outlined below. Related frameworks and algorithms for other settings appeared in [6, 8].

At the current iterate $\bar{\mathbf{u}}^{(k)}$, agent m computes a *locally descent direction* $\bar{\mathbf{d}}_m^{(k)} = \hat{\mathbf{u}}_m^{(k)} - \bar{\mathbf{u}}_m^{(k)}$ by solving a linear programming (LP) problem that minimizes $\nabla f_m(\bar{\mathbf{u}}_m^{(k)}, \bar{\mathbf{y}}_m^{(k)})^T (\hat{\mathbf{u}}_m^{(k)} - \bar{\mathbf{u}}_m^{(k)})$ subject to the original constraints imposed on the decisions of agent m. It then produces the next iterate $\bar{\mathbf{u}}_m^{(k+1)} = \bar{\mathbf{u}}_m^{(k)} + \alpha_m^{(k)} \bar{\mathbf{d}}_m^{(k)}$ by finding a step $\alpha_m^{(k)}$ that satisfies the *Armijo rule*. Given $(\bar{\mathbf{u}}_m^{(k)}, \bar{\mathbf{y}}_m^{(k)}) \in U_m \times Y_m$, $\bar{\mathbf{d}}_m^{(k)} \neq 0$ is a *locally feasible direction* at $(\bar{\mathbf{u}}_m^{(k)}, \bar{\mathbf{y}}_m^{(k)})$ if $\bar{\mathbf{u}}_m^{(k)} + \alpha_m \bar{\mathbf{d}}_m^{(k)} \in U_m$ for all $\alpha_m > 0$ that are sufficiently small. A locally feasible direction $\bar{\mathbf{d}}_m^{(k)}$ at a nonstationary point $(\bar{\mathbf{u}}_m^{(k)}, \bar{\mathbf{y}}_m^{(k)})$ is a *locally descent direction* if $\nabla f_m(\bar{\mathbf{u}}_m^{(k)}, \bar{\mathbf{y}}_m^{(k)})^T \bar{\mathbf{d}}_m^{(k)} < 0$.

Assumption 3, Assumption 4, and agent iterations as delineated above—which use a locally descent direction obtained by solving

an LP problem and satisfy the Armijo rule—ensure that $\bar{\mathbf{u}}^{(k)}$ arrives at a stationary point of $\{P_m\}$ and, thereby, a solution to P. Effectively, the agent network implements a distributed feasible direction method for quadratic programming.

The constraint structure in split control admits simplifications in the iterative processes of the agents. For each junction j and phase i, suppose the maximum green time $u_{j,i}^{max}$ is $C_j - L_j - \sum_{i \in F_j} u_{j,i}^{min}$. Then the MPC problem P, given by (11a) through (11c), can be recast using control variables $\Delta\bar{\mathbf{u}}_m(k)$ with green times in excess to the minimum, namely $\Delta u_{j,i} = u_{j,i} - u_{j,i}^{min}$. This variable change simplifies the inequality constraints (11b) which become simple bounds of the form $\Delta\bar{\mathbf{u}}_m(k) \geq 0$. As a result, the linear program for computing a locally descent direction is solved analytically: the LP constraint structure consists of block-diagonal equalities (one for each time period) and simple variable bounds; the solution is obtained by examining the coefficients of the gradient $\nabla f_m(\bar{\mathbf{u}}_m^{(k)}, \bar{\mathbf{y}}_m^{(k)})$.

The agents are not limited to using the feasible direction method sketched above. They can apply any quadratic-programming algorithm that meets the Armijo rule or otherwise solves the subproblem up to optimality. The active set and gradient projection methods [2] are candidates to replace the feasible direction algorithm.

4. EXPERIMENTAL STUDIES

This section presents results from the application of the TUC LQR strategy and the multi-agent MPC framework for the signaling split control of the UTN depicted in Fig. 2. While TUC uses equation (4) to model the system and objective (5) to compute a feedback gain matrix, it is more appropriate for the MPC framework to express the control problem in terms of objective (10a) subject to constraints (10b) through (10d), as control signals must lie within bounds and hold constant cycle periods.

4.1 Network Set-up

Additional parameters must be specified to fully model the UTN depicted in Fig. 2, such as saturation flows, turning rates, traffic demands, and exit rates. The simulation environment was further simplified: the exit rates of the network are null; lost times in between phases are four seconds; all offsets are zero; and all network links have equal length so that their occupancy contributes with the same weight in the objective function.

Up to this day, fixed-time signaling is still the most usual type of split control worldwide. Since fixed-time control is not adaptive, drivers *learn* and *predict* network dynamics which induce a matching behavior between drivers and the traffic system. Therefore, from a practical perspective, traffic engineers favor a control policy that penalizes deviation from the nominal fixed-time split, rather than a control strategy formulated in terms of absolute control values.

Table 1 presents the nominal splits of the sample UTN and the other aforementioned parameters—nominal splits were obtained with Webster's procedure. Some turn rate parameters are irrelevant and not presented, namely the ones that do not take part in the inflow of another controlled link—e.g., in link 11, **12**:0.5 means that 50% of that link exit flow enters link 12, while the remaining vehicles take a route outside the scope of the controlled network and are not accounted for. In the simulated analysis, all junctions have a constant cycle of 120 s and have two phases, with the exception of junction 1 which has three phases. The mean inflows in the input links are: $q_1 = 800$ veh/h; $q_2 = 1300$ veh/h; $q_3 = 900$ veh/h; $q_8 = 900$ veh/h; and $q_9 = 700$ veh/h.

Table 1: Network specification.

Link (z)	Sat. Flow (S_z) (veh/h)	Nom. Split (u_z^N) (s)	Turn Rate ($t_{z,w}$) (to link: f%)
1	3600	29	**4**:0.2; **6**:0.05; **11**:0.05; **13**:0.7
2	3600	49	**4**:0.25; **6**:0.3; **11**:0.3; **13**:0.15
3	3600	32	**4**:0.65; **6**:0.05; **11**:0.05; **13**:0.15
4	3600	72	—
5	3600	40	—
6	1800	57	**5**:0.5
7	3600	55	**5**:0.8
8	3600	63	**7**:0.4; **10**:0.6
9	3600	49	**7**:0.6; **10**:0.4
10	3600	60	**12**:0.8
11	1800	52	**12**:0.5
12	3600	55	—
13	3600	57	—

4.2 Numerical Analysis

To validate the proposed multi-agent control framework, a set of MPC problems were solved covering a range of initial conditions. A centralized agent solved the global MPC problem P, while a multi-agent system solved the corresponding sub-problem network $\{P_m\}$ for each of the initial conditions, allowing their performance to be compared. Both approaches used a standard quadratic-programming (QP) algorithm [14, active set method] and the feasible direction method outlined above. For the distributed feasible direction method, experiments showed that the acceptance degree $\sigma = 0.3$ and the step-contraction parameter $\beta = 0.3$ for the Armijo rule induce the best convergence rate in the given scenarios. Notice that the distributed QP approach implicitly satisfies the Armijo rule. A set of ten randomly obtained queues defined the initial conditions in the experiments, whose results appear in Table 2.

As the tolerance of the optimization package used for the centralized QP computation could not be modified, results illustrate the computational effort to reach the actual optimal cost. In other instances we assume that the solution has converged once it is within a 0.1% error margin from the optimal objective, previously computed by the centralized QP algorithm.

The experimental results show a trade-off between the complexity of the algorithm and the number of iterations required for convergence. On the other hand, the distinction between the distributed and centralized approach is small, specially so with respect to the feasible direction method. Most importantly, the numerical results confirm the multi-agent control theory outlined above, satisfying optimality conditions for diverse initial conditions.

4.3 Simulated Analysis

The simulated results are from AIMSUN© replications of the sample UTN. AIMSUN© has a powerful micro-simulator for traffic applications which provides accurate modeling of complex networks. Furthermore, it offers a useful API module with the ability to interface, through Python and C++ routines, with almost any external module that needs access to internal data during simulation run time.

The solution of the optimization problems required by the LQR strategy and the multi-agent MPC used several tools: the PSFL licensed Python 2.5 programming language; the OSI-Approved Open Source numerical package for Python, NumPy; the GNU licensed

Table 2: Computational results for a set of ten initial conditions with 0.1% error tolerance.

	Quadratic Programming				Feasible Direction			
	CPU Time (ms)		Iterations		CPU Time (s)		Iterations	
	Mean	Max	Mean	Max	Mean	Max	Mean	Max
Agent 1	21.9	46.9	3.3	7	2.16	8.12	76	290
Agent 2	15.6	46.9	2.1	4	1.98	8.20	72	290
Agent 3	10.9	46.9	2.8	8	1.85	7.59	69	280
Agent 4	14.1	31.2	3.0	6	1.85	7.50	76	294
Agent 5	1.6	15.6	2.4	5	1.83	7.70	69	280
Agent 6	6.3	46.9	1.2	2	1.81	7.67	67	280
Multi-agent	75.0	156.2	14.8	25	11.58	47.16	425	1710
Centralized	26.6	46.9	3.9	7	13.59	48.09	370.9	1299

Table 3: Average results for ten AIMSUN© replications.

	Travel Time (s/km)		Density (veh/km)	
	Avg.	Std. Dev.	Avg.	Std. Dev.
LQR	182.759	3.928	17.853	0.355
MA-MPC	180.288	3.619	17.564	0.314

CVXOPT optimization package; and the solver MOSEK©.

The simulated scenario had a duration of one hour with the inflow patterns given above. Although inflows have constant mean, vehicles do not necessarily enter the network at a constant rate because the simulator uses an exponential feed algorithm. All frameworks share the same weighting matrices, with state matrix $Q = I$ and control matrix $R = rI$, $r = 0.003$, as is usual in the TUC policy. As mentioned earlier, the store-and-forward model requires the control interval to be greater than any cycle in the network. Following this premise, a control cycle of 200 s was defined for the sample UTN. Furthermore, the multi-agent MPC achieved best results when both the prediction and control horizon were set to a single control step. Table 3 reports the average results covering a set of 10 random initial conditions.

4.4 Discussion

The experimental results show that the proposed multi-agent MPC framework can perform slightly better than the TUC strategy in signaling split of urban traffic networks. Nevertheless, some aspects need further investigation.

The first aspect is the length of the prediction horizon. The fact that wider prediction degrades system performance indicates that the TUC store-and-forward model may not be adequate for prediction, suggesting that a more precise model could improve overall system performance.

Another issue is the effect produced by the weighting matrices, particularly the effect on control signals. The weight chosen in the simulation penalizes control deviation from nominal signals very lightly, allowing drastic changes without incurring substantial cost increase in the objective function. This increases the responsiveness of the control system but, on the other hand, affects the synchronization among consecutive junctions.

Although offset and stage specification were not object of study in this paper, they influence the performance metrics and should be accounted for in simulated analyses. This justifies the design of the sample UTN with only one-way links, which ease the specification of stages. Although such measure increases the reliability of the experimental model, circumventing the distortion caused by the lack of synchronization control and offset dimensioning is another issue to be investigated.

Some considerations regarding the practical implementation of the proposed strategy are pertinent. First, the method requires full knowledge of system state, either through the installation of inductive loop detectors or other means such as image detection devices. Although the exchange of messages is necessary at every control cycle, it does not constrain the application of multi-agent MPC due to the large control interval. The same communication infrastructure used in the centralized control scheme can be used for distributed control with a centralized message relay. Finally, several practical issues should be analyzed in field applications, such as the influence of noise, delays, and poor synchronization.

The slightly better multi-agent MPC performance is further endorsed by other advantages of this approach. First, the multi-agent MPC circumvents the lack of reconfigurability of the TUC strategy [11], as the addition of nodes to the network affects only the sub-systems in the vicinity. Another advantage is the use of more precise traffic-flow models, such as the non-linear representation proposed in [1].

5. SUMMARY AND FUTURE WORK

This work has contributed to the state-of-the-art by proposing a framework for multi-agent control of dynamic systems. The class of systems comprises linear dynamic networks that are assembled by interconnecting dynamically-coupled sub-systems. This representative class encompasses dynamic networks that use the store-and-forward model to represent traffic flow dynamics, conveniently capturing the local couplings between neighboring junctions.

The signaling split control for the store-and-forward model entails solving a constrained, infinite time, linear quadratic regulator problem. The TUC approach obtains a feedback control law with the unconstrained LQR technique by disregarding the constraints on control signals, whose feasibility is recovered by solving a quadratic program. Model predictive control handles constraints in a systematic way by using a finite time, rolling horizon and solving optimization problems on-line. This paper proposed a decomposition of the MPC problem in a set of locally coupled sub-problems that are iteratively solved by a network of distributed agents. Under certain mild conditions and synchronous work, the iterations of the multi-agent control system can be shown to be drawn towards a fixed point that induces a globally optimal solution to the MPC problem. Numerical experiments illustrate the convergent behavior of the multi-agent system and compare its speed with that of an ideal, centralized agent that solves the problem single-handed. Simulated studies corroborate the hypothesis that a control algorithm that handles constraints explicitly can outperform strategies that treat constraints in an ad hoc manner.

The work reported heretofore is in its nascent, opening up a number of opportunities for research and studies with multi-disciplinary contributions across the fields of multi-agent technology, control engineering, and transportation systems. Some directions for research are:

- numerical and simulated studies with very large networks aimed to confirm the potential of the multi-agent MPC framework;
- the formulation and application of new traffic models, representing more accurately the flow of vehicles;
- studies demonstrating the flexibility and scalability of multi-agent systems, such as the reconfiguration of a traffic junction which would demand only local adjustments, involving the junction and its immediate neighboring intersections; and
- the formal extension of the framework to handle constraints on state variables, such as limits on queue lengths.

6. REFERENCES

[1] K. Aboudolas, M. Papageorgiou, and E. Kosmatopoulos. Control and optimization methods for traffic signal control in large-scale congested urban road networks. In *Proceedings of the American Control Conference*, pages 3132–3138, New York, USA, July 2007.

[2] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA, 1995.

[3] C. Bielefeldt, C. Diakaki, and M. Papageorgiou. TUC and the SMART NETS project. *IEEE Transactions on Intelligent Transportation Systems*, pages 55–60, 2001.

[4] E. F. Camacho and C. Bordons. *Model Predictive Control*. Springer-Verlag, 2004.

[5] E. Camponogara and L. B. de Oliveira. Distributed optimization for model predictive control of linear dynamic networks. Technical report, UFSC, 2007. http://www.das.ufsc.br/∼camponog/papers/tech-dmpc-tuc-07.pdf.

[6] E. Camponogara, D. Jia, B. H. Krogh, and S. N. Talukdar. Distributed model predictive control. *IEEE Control Systems Magazine*, 22(1):44–52, February 2002.

[7] E. Camponogara and S. Talukdar. Designing communication networks to decompose network control problems. *INFORMS Journal on Computing*, 17(2):207–223, 2005.

[8] E. Camponogara and S. N. Talukdar. Distributed model predictive control: synchronous and asynchronous computation. *IEEE Transactions on Systems, Man, and Cybernetics – Part A*, 37(5):732–745, September 2007.

[9] 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, Brazil, October 2007.

[10] C. Diakaki. *Integrated Control of Traffic Flow in Corridor Networks*. PhD thesis, Department of Production Engineering and Management, Technical University of Crete, Greece, 1999.

[11] C. Diakaki, M. Papageorgiou, and K. Aboudolas. A multivariable regulator approach to traffic-responsive network-wide signal control. *Control Engineering Practice*, 10(2):183–195, 2002.

[12] C. Diakaki, M. Papageorgiou, and partners of the project TABASCO. Urban integrated traffic control implementation strategies. Technical Report Project TABASCO (TR1054), Transport Telematics Office, Brussels, Belgium, September 1997.

[13] D. C. Gazis and R. B. Potts. The oversaturated intersection. In *Proceedings of the Second International Symposium on Traffic Theory*, pages 221–237, 1963.

[14] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization*. Academic Press, London, UK, 1981.

[15] N. Jennings. On agent-based software engineering. *Artificial Intelligence*, 117:277–296, 2000.

[16] E. Kosmatopoulos, M. Papageorgiou, C. Bielefeldt, V. Dinopoulou, R. Morris, J. Mueck, A. Richards, and F. Weichenmeier. International comparative field evaluation of a traffic-responsive signal control strategy in three cities. *Transportation Research Part A: Policy and Practice*, 40(5):399–413, 2006.

[17] F. Kühne. Controle preditivo de robôs móveis não holonômicos. Master's thesis, Graduate Program in Electrical Engineering, Federal University of Rio Grande do Sul, Brazil, 2005.

[18] F. P. Maturana, R. J. Staron, and K. H. Hall. Methodologies and tools for intelligent agents in distributed control. *IEEE Intelligent Systems*, 20(1):42–49, 2005.

[19] R. R. Negenborn, B. D. Schutter, and J. Hellendoorn. Multi-agent model predictive control for transportation networks: serial versus parallel schemes. To appear in *Engineering Applications of Artificial Intelligence*, 2007.

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

[21] M. Papageorgiou, C. Diakaki, V. Dinopoulou, A. Kotsialos, and Y. Wang. Review of road traffic control strategies. *IEEE Proceedings*, 91(12):2043–2067, 2003.

[22] D. Srinivasan and M. C. Choy. Cooperative multi-agent system for coordinated traffic signal control. *IEE Proceedings. Intelligent Transport Systems*, 153(1):41–49, 2006.

[23] E. Tatara, I. Birol, F. Teymour, and A. Çinar. Agent-based control of autocatalytic replicators in networks of reactors. *Computers & Chemical Engineering*, 29:807–815, 2005.

[24] E. Tatara, A. Çinar, and F. Teymour. Control of complex distributed systems with distributed intelligent agents. *Journal of Process Control*, 17:415–427, 2007.

Geosimulation of Parking in the City

Itzhak Benenson
Department of Geography and Human Environment,
University Tel Aviv
Ramat Aviv
Tel Aviv, 69978, Israel
+972-54-4522-570
bennya@post.tau.ac.il

Karel Martens
Institute for Management Research,
Radboud University Nijmegen
P.O. Box 9108
6500 HK, Nijmegen, the Netherlands
+31-24-361-2740
k.martens@fm.ru.nl

ABSTRACT
We present PARKAGENT, an agent-based, spatially explicit, model for parking in the city. PARKAGENT is based on the geosimulation approach, combining real-world GIS database with a multi-agent system. The model simulates the behavior of each driver in a spatially explicit environment and is able to capture the complex self-organizing dynamics of a large collective of parking agents within a non-homogeneous (road) space. The model is developed as an ARCGIS application and can work with a practically unlimited number of drivers. Standard model outputs include distributions of search time, walking distance, and parking costs, each of which can be generated per driver groups, per area and per time interval.

Based on field estimations of supply of and demand for parking and parameters of drivers' behavior, we apply PARKAGENT for investigating several parking scenarios in an over-saturated situation in Tel Aviv. The model shows that, while a limited amount of additional parking has hardly any impact on average search time or walking distance, it strongly affects the occurrence of extreme values. We also compare the effects of a concentrated versus a spatially distributed addition of new parking facilities in a Tel Aviv neighborhood, where demand for parking essentially exceeds supply.

Categories and Subject Descriptors
H.1.0 Information systems, Models and principles, General

General Terms
Algorithms, Measurement, Economics, Experimentation, Human Factors

Keywords
parking, GIS, agent-based modeling, spatially-explicit modeling, on-street parking

1. INTRODUCTION
In this paper we present a geo-simulation model of parking in the city, termed PARKAGENT. The model is developed as a Geographic Automata System [3], and represents urban reality by means of interacting inanimate objects, representing urban infrastructure elements, and animated objects, representing drivers. The model simulates driver's behavior at all stages of the parking process and includes description of driving towards the destination, searching for parking, and exiting the parking place after a variable period of time. Traditional approaches to studying parking in the city aggregate individual drivers into an "average one", who, in turn, reacts to an "average" environment [e.g. 6; 8]. However, averages are inherently conservative and are relatively insensitive to policy interventions. The disaggregate view of parking makes it possible to capture the diversity in terms of driver behavior, urban structure, and transport policies, as well as the dynamic interplay between them. For example, a disaggregate view makes it possible to determine the fraction of unsatisfied drivers – those who either pay too much or search too long for parking, or who park too far from the destination. By using a geosimulation approach, PARKAGENT makes it possible to investigate *spatial and temporal distributions* of payments, search time, distance to destination, etc., which is crucial within the modern city with its highly heterogeneous parking facilities, traffic situation and parking demand.

We are aware of only two models of similar kind [11; 12]. These models, however, focused on simulating the behavior within a constant spatial setting. They neither account for the continuous change in on-street and off-street parking capacity as a result of cars entering and exiting parking facilities, nor for drivers' immediate adaptation to the changing parking situation when driving toward destination.

Our model, in contrast, is able to analyze the instantaneously varying parking situation created by the drivers themselves. We employ it to study residential parking in the evening hours, which has received relatively only limited attention in the literature. In contrast to commuters, who can respond in various ways to changes in parking policies, car-owning residents have little choice when returning home with their cars at the end of the day. In center of most cities in the industrialized world, the majority of them will have to find an on-street parking place, preferably close to the location of residence, for overnight parking. We use the model to analyze how these resident-parkers respond to different parking situations and policies at the home-end of the trip. The case material is taken from the city center of Tel Aviv.

2. GEOSIMULATION AS AUTOMATA-BASED MODELING
Most generally, geosimulation models deal with interacting discrete objects and their behaviors [3]. The methodology of

geosimulation makes use of automata-based techniques, particularly cellular automata for immobile and multi-agent systems for mobile objects. The relationships between the objects are interpreted according to the logic of the entity-relationship model, which considers relationships separately from the objects that are related [2].

Contemporary GIS provides much support for the development of geosimulation models. Each GIS layer can be used to define a class of objects and supply the initial location, form and characteristics of the immobile and mobile objects. GIS can be further employed for storing objects' geographic and attributive properties as they change in time. Non-mapable tables are used for storing and managing relationships of any degree [2].

3. THE PARKAGENT MODEL

The PARKAGENT model has been developed according to the geosimulation principles. It is a *spatially-explicit model*, that is, its dataset contains high-resolution urban GIS, with the layers representing *every inanimate entity of traffic infrastructure important for investigating the parking process* - street segments, on-street parking places, off-street parking places, and buildings (as destinations). Animate objects – drivers – *behave*, i.e. they drive to the destination, search for a parking place, park, and leave the parking place when their activity has ended.

The model is developed as an ArcGIS application, and, despite the very high spatial and temporal resolution, can work with a practically unlimited number of drivers whose destination can be located anywhere in the city. The model interface contains a set of tools for preparing the model database on the base of GIS layers of the city under study, for establishing model scenarios, and for storing the simulation results. The results of the runs can thus be analyzed further.

3.1 The Discrete Representation of Driving and Parking

The focus on the parking process determines the spatial resolution of the model, which, in turn, entails the temporal resolution. The spatial resolution of the model is defined by the typical distance between parking cars $\mathbf{\Delta d}$. The temporal resolution of the model $\mathbf{\Delta t}$ is determined by the time it takes a car to pass $\mathbf{\Delta d}$ at the typical maximal speed $\mathbf{v_{max}}$ of a car during parking search. The values of $\mathbf{\Delta d}$ and $\mathbf{v_{max}}$ in case of Tel-Aviv, as estimated in the field surveys, are: $\mathbf{\Delta d \approx 4m}$, $\mathbf{v_{max} < 18\ km/h = 5\ m/sec.}$ Consequently, we use $\mathbf{\Delta t = \Delta d / v_{max} \approx 1\ sec}$ as temporal resolution of the model for Tel Aviv case. By using this resolution, we are able to adequately capture the parking search process of drivers and their continuous decisions to park or to continue searching for a parking space.

3.2 The Location of the Animate Objects

Inherent to geosimulation, animate objects are *located by relationship*. That is, the location of a driving car is given by means of linear referencing, while the location of a parked car is given by reference to a parking place. Both driving along a street and occupying and vacating a parking place is represented by means of standard database operations. That is, the advance of a car **c** from location $\mathbf{p_1}$ on segment $\mathbf{s_1}$ to location $\mathbf{p_2}$ on segment $\mathbf{s_2}$, is represented by deleting the row $\mathbf{(c, (s_1, p_1))}$ from the relationship table and inserting the row $\mathbf{(c, (s_2, p_2))}$. It is worth noting that according to the geosimulation paradigm, the cars do not "see" each other; to recognize whether advancement on a street segment is possible, car $\mathbf{c_1}$ has to retrieve its location $\mathbf{(c_1, (s_1, p_1))}$, and then retrieve the location $\mathbf{(s_1, p_2)}$ next to $\mathbf{(s_1, p_1)}$. If the result of this transaction is **NULL**, then $\mathbf{c_1}$ can advance to $\mathbf{(s_1, p_2)}$; otherwise the street segment is jammed and the car can not advance in the current time-step (model iteration).

3.3 The GIS Database

The main components of the model GIS database are (Figure 1):

- street segments (line layer) characterized by driving and parking permissions and prices per segment, turn permissions (non-mapable table of relationships) and junctions (point layer);
- buildings (polygon layer), characterized by type of use and capacity; house entrances (point layer) employed as destination points; and
- off-street parking lots (polygon layer), characterized by capacity and price.

The model tools enable the construction two additional layers. The layer of ***Lanes*** is constructed in order to discretely represent one- and two-way streets. Depending on whether one- or two-way traffic is permitted along a street segment, one or two series of points are constructed, with the points located at a distance of $\mathbf{\Delta d}$ from each other. The series of points is either located on a segment centerline (in case of a one-way street) or on two separate lines at both sides of a street centerline (in case of a two-way street). Each series of points is thus used for representing driving in one direction only (Figure 1). The two lines of a two-way street segment touch each other at junctions. The traffic direction along one or two constructed lines is derived from the traffic permission of the segment.

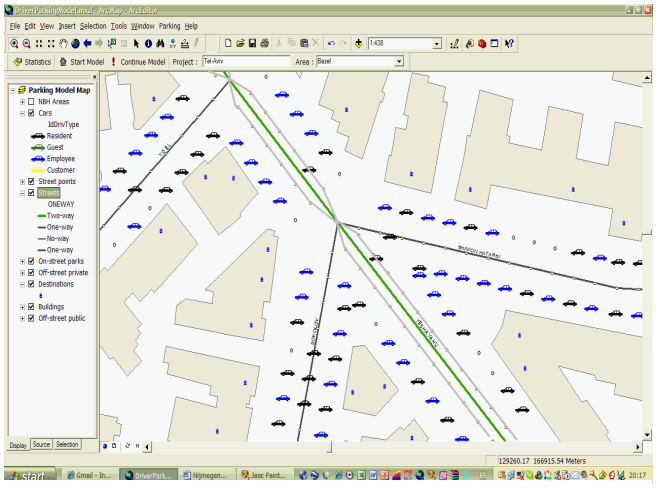


Figure 1. The basic and derived layers of the PARKAGENT model in the ArcGIS model window.

The second additional layer consists of on-street ***parking places***. These are represented by a layer of points always constructed at both sides of the segment centerline. Each parking place is located at a distance of $\mathbf{\Delta d}$ from the next parking place (Figure 1). In Tel Aviv this distance $\mathbf{\Delta d}$ is about 4 meters and this value is employed in the current version of the model.

The layer of parking places contains all physically existing places for parking, including places where parking is not allowed yet

feasible. The actual legal right to park for vehicles of a specific type, for specific time intervals, as well as the price for each group of drivers (including zero price) are either transferred from the road network line features or updated after the layer is constructed. For instance, in case parking is only allowed on one side of a street segment, the parking spots on one side of the center line receive the attribute 'parking not permitted'.

4. DRIVER'S PARKING BEHAVIOUR

The model works in a discrete time and space; at each time-step (iteration) Δt every non-parking vehicle can make a move, the size of which is determined by the vehicle's speed. As mentioned above, the model's temporal resolution is very high: Δt = 1 sec. In case Δd and Δt are changed, all model calculations are automatically adjusted to the new values.

4.1 Representation of Car Advance

Formally, the single car street speed of $v_{s(treet)}$ km/h is recalculated into speed $v_{m(odel)}$ measured in Δd lengths units per Δt. The value of v_m is then represented as

$$v_m = v_{m,int} + v_{m,dec} \qquad (1)$$

where $v_{m,int}$ is an integer part of v_m and $v_{m,dec}$ is a decimal part.

To illustrate, let **c**'s speed be 15 km/h, Δd = 4m and Δt = 1 sec. In this case $v_m = 1.04\ \Delta d/\Delta t$, thus resulting in $v_{m,int} = 1$, $v_{m,dec} = 0.04$.

To simulate driving at a "non-integer" speed v_m, we then generate uniformly distributed on (0, 1) a random number **r** and assume that in case there are no cars in front of car **c**, it advances for d_c units of Δd, where

$$d_c = v_{m,int} + 1 \quad \text{if } v_{m,dec} > r$$

$$d_c = v_{m,int} \quad \text{otherwise} \qquad (2)$$

For example, for a car that drives at 15 km/h, $v_{m,int} = 1$ and $v_{m,dec} = 0.04$ and it advances one Δd = 4m unit deterministically and then one more unit with a probability of 0.04.

During parking search, car velocity is low. As recorded during test trips with drivers, a driver tends to decrease her velocity to 20-25 km/h when starting to estimate the state of parking in an area. The speed is further decreased to 10-12 km/h ($v_{m,int} = 0$ and $v_{m,dec} \sim 0.69 - 0.83\ \Delta d/\Delta t$) when a driver actually starts searching for place to park [5] . In the search phase, we thus ignore the possibility of acceleration as employed in e.g. car following models [9]. However, as mentioned above, to account for interaction between parking cars, the model drivers adjust their movements with respect to the car in front of them. Before advancing each Δt distance d_c, the driver checks whether the position in front of her is free or not. If occupied, all further advancements during the given Δt is cancelled.

The order of the cars for advancing is established randomly, anew at every Δt.

4.2 Route Choice

A driver located on a street segment advances according to the formulae (1) – (2), accounting for the cars in front of her. When passing a junction, the driver has to decide which direction to take in order to approach towards the destination. In the model, driver's decision is based on the comparison of the distance to the destination from all "next" junctions, which are defined as the first junction on the street segments the driver can choose from for advancement (Figure 2). To avoid looping we assume that the driver does not choose any of two last visited junctions.

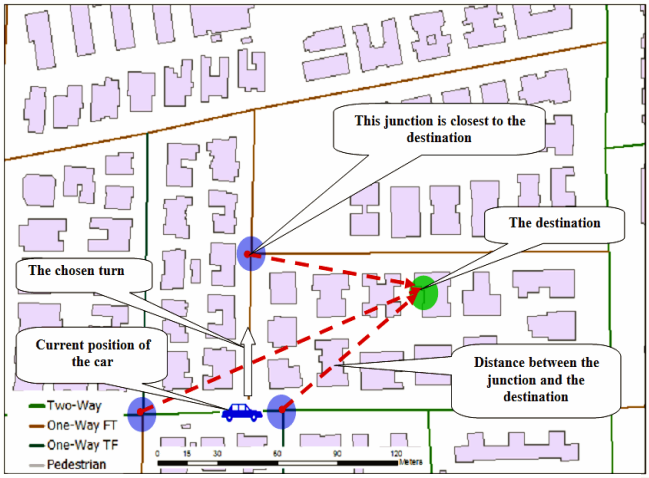


Figure 2. Schematic presentation of the route choice component of driver's behavioral algorithm

We assume that the model driver possesses some knowledge of the city street network and thus selects the segment which next junction is closest to the destination. The model thus follows the approach of [4], who view route choice as the result of a sequence of decisions – one at each intersection encountered.

A driver enters the system at a distance $D_{awareness}$ from her destination, where the driver becomes "aware" of the need to start searching for parking. She begins searching for an actual parking place at a distance $D_{parking}$. In the current version of the model these distances are set at 300 and 150 meters, respectively, following field observations.

4.3 Representation of Driver's Parking Behaviour

The rules of agent's behavior in the model depend on the stage of the parking process. Generally, approaching the $D_{awareness}$ distance the driver follows a sequence of decisions that can be described as follows: (1) Decrease velocity, continue driving and estimate parking supply → (2) Decrease velocity to be able to park and try to park for free as close as possible to the destination → (3) If parking search lasts too long, search for a free place further from the destination or park at an affordable price → (4) If failed, search until finding any parking place, ignoring price or distance. Note that this sequence of decisions holds for Tel Aviv residents returning home, as they can park for free on-street in their own neighborhood. When drivers have to pay for parking, the sequence of decisions will obviously be more complex in nature and depend on parameters like willingness-to-pay for parking and value-of-time of each individual driver.

To formalize this view we introduce the following behavioral components:

1. Driving towards destination, estimating parking supply,
2. Searching for parking and parking before reaching the destination,

3. Searching for parking and parking after passing the destination,
4. Staying at the selected parking place,
5. Leaving the parking place and driving out of the system.

Each of these stages is formalized as follows:

Stage 1: Driving towards destination

From $\mathbf{D_{awareness}}$, a driver estimates the available parking supply by estimating the fraction $\mathbf{p_{unoc}}$:

$$\mathbf{p_{unoc} = N_{unoc}/(N_{unoc} + N_{occ})} \quad (3)$$

where $\mathbf{N_{occ}}$ is the number of occupied, and $\mathbf{N_{unoc}}$ is the number of unoccupied parking places.

Stage 2: Searching for parking and parking before reaching the destination

At distance $\mathbf{D_{parking}}$ the model driver decreases her velocity to 12 km/h. At any distance $\mathbf{D < D_{parking}}$ she estimates the expected number of free parking places $\mathbf{F_{exp}}$ to be found before reaching the destination as:

$$\mathbf{F_{exp} = p_{unoc*}D/\Delta d} \quad (4)$$

The driver decides to continue driving with probability $\mathbf{P_{drive}}$, which depends on $\mathbf{F_{exp}}$ in a piecewise-linear manner:

$$\begin{aligned} &\mathbf{P_{drive} = 0} && \mathbf{\text{if } F_{exp} < F_1} \\ &\mathbf{P_{drive} = (F_{exp} - F_1)/(F_2 - F_1)} && \mathbf{\text{if } F_1 \le F_{exp} \le F_2} \quad (5) \\ &\mathbf{P_{drive} = 1} && \mathbf{\text{if } F_2 < F_{exp}} \end{aligned}$$

In the model applications the values of $\mathbf{F_1 = 1}$ and $\mathbf{F_2 = 3}$ are used.

To guarantee driver's reaction to the changes in local parking supply as observed during driving, we assume that each driver instantaneously re-estimates parking supply on the way to the destination.

This algorithm results in drivers parking close to the destination in case of a sufficiently high supply of free on-street parking places.

Stage 3: Searching for parking and parking after passing the destination

At this stage, given the fact that on-street parking is for free for residents, the decision to park follows the commonsensical view that "the driver simply tries to find an unoccupied parking place not too far from the destination". We express this in the model by a steady increase of $\mathbf{D_{parking}}$ distance. In what follows we assume that $\mathbf{D_{parking}}$ grows linearly in time at a rate of $\mathbf{\Delta D_{parking}}$, i.e. $\mathbf{D_{parking}(t) = D_{parking} + \Delta D_{parking}*t}$, until reaching the maximal value of $\mathbf{D_{parking,\ max}}$. We also assume that after passing the destination and choosing a street segment at a junction, the driver always tries to stay within or as close as possible to the area with $\mathbf{D_{parking}(t)}$ radius around the destination.

In addition, we assume that the driver whose accumulated search time exceeds $\mathbf{T_{search,max}}$, just parks at the paid parking lot closest to the destination. We follow Tel Aviv reality and assume that an off-street paid parking place is always available.

Stage 4 and 5: Staying at parking place and leaving the system

Each driver parks for the time interval that is attributed exogenously to the driver. After this given parking duration, she drives towards one of the exit points of the system, which is an attribute of the driver as well, following the algorithm (1) – (2). Data on parking duration are derived from field surveys and depend on the type of driver.

4.4 Groups of Drivers

In the model, we distinguish between **Residents** and **Visitors**, who currently differ in terms of the required parking fee and the time they enter and leave the area.

4.5 Model Output

Our model records the life-path of every model driver, based on which a number of aggregate outputs are produced. In this paper we focus on the following aggregate outputs relevant to residents: the *distributions* of parking search time and air distance to destination for the drivers who succeeded to park in the area, the dynamics of the number of free parking places, and the overall number of drivers searching for a parking place longer than 10 minutes. Note, that many other aggregate parameters can be generated, such as the distribution of paid parking fees. All aggregate characteristics are constructed for each group of drivers searching for a parking place in a *specific* area during a *specific* time interval.

The interaction between the basic components of the model could be also analyzed, such as the relationship between the duration of the parking search and the distance between parking place and destination.

5. ESTIMATION OF KEY PARAMETERS

The key parameters of the rules that guide driver's driving, parking search, parking and leaving behavior are based on a number of street surveys carried out in Tel Aviv in 2005-2006. The survey results have furthermore been used for establishing the initial and boundary conditions of the Tel Aviv simulations.

Two main surveys were carried out in the case-study area, the Basel neighborhood in Tel Aviv, during two consecutive weeks. We distinguished between visitors and local residents based on the presence of a local tag on a car (which is supplied by the municipality to local residents only). The total area of the Tel-Aviv center is about 20 km^2. It is divided into nine parking areas of largely equal in size. Only local residents are allowed to park for free on-street within their parking area. The fines for illegal parking in Tel-Aviv are high and the enforcement is tight; in practice, virtually every resident of Tel-Aviv has a local tag. Visitors have to pay parking fees and can park on-street for a maximum of three hours.

In the daytime, the main results of the surveys are as follows:

- Close to 60% of the on-street parking places were occupied by owners of a local area tag.
- Half of the remaining 40% of parking places was occupied by visitors, and half remained empty.

In what follows we employ 60% as an estimate of residents' on-street parking use during daytime.

The main results of the night surveys are as follows:

– All feasible parking places – both legal and illegal – are occupied. The illegal parking is the consequence of the fact, well-known among local residents, that parking enforcement runs only between 6:30 - 21:00h.

– The fraction of visitors recorded in the night survey was close to 10%.

– Based on the plate numbers and data from the Israeli Central Bureau of Statistics we were able to estimate the distribution of the distance between a car's parking place and the driver's destination (home location). More than 50% of the drivers who parked on-street were located at an air distance of less than 100m and more than 90% at an air distance of 250m or less from their home, the latter corresponding to less than 5 min street walk.

Extrapolating the above dependency of the fraction of parked cars on distance to destination, we set $\mathbf{D_{parking,max}} = 350$ m.

– Based on the local area tags we were able to identify area residents parking at paid parking lots in the neighborhood. Residents entering the lot were asked to estimate their parking search time before entering the lot. The vast majority of residents indicated to be searching either "more than 5 minutes for sure" or "10 minutes or so". This confirms Shoup's [10] view that resident drivers who do not have a dedicated private or public off-street parking place have a tendency to cruise for parking in order to find a free on-street parking place.

Numerically we thus assumed that the typical maximal search time of the residents is $\mathbf{T_{search,max}} = 10$ minutes and the rate of growth of the area acceptable for parking is $\mathbf{\Delta D_{parking}} = \mathbf{(D_{parking,max} - D_{parking})/T_{search,max}} = (350 - 150)/10 = 20$ m/min.

6. APPLICATION OF THE PARKAGENT MODEL

To explore the potential of the model in practice, it was employed to the analysis of the parking situation in the Basel neighborhood, which is about 1.4 km^2 in size and contains a total of about 1,550 buildings. The Basel neighborhood is suffering from a substantial imbalance between existing parking supply and demand for residential parking. The results of the survey confirm the municipality's view that the problems are most notable in the evening hours, when local residents have problems finding a parking place to park their vehicle overnight. The solution proposed by the municipality is the extension of a planned underground parking garage underneath a yet-to-be-built residential building and sale of the additional parking places to residents living nearby. This new residential building is located in the center of the Basel neighborhood.

The estimate of total parking demand in the Basel area is based on the number of apartments and registered businesses per building, and on the number of parking tags issued to the residents in the area, both available as part of Tel Aviv Municipal GIS. The estimate of on-street parking supply is based on the GIS layer of streets. All physically available spaces in the area are used for overnight parking. The only unoccupied places at night are located at the entrances to public and private off-street parking places, at pedestrian crossings, and next to junctions. All these can be easily estimated from the GIS data. Data on off-street public parking supply were also taken for the GIS data, while off-street private parking space was estimated based on the field surveys.

Based on the estimate of residents' parking demand and the observed 10% visitor parkers, the overall demand/supply ratio overnight is about 3,500/2,850 ≈ 1.23. However, the demand/supply ratio varies during the evening period 17:00-21:00h, during which the share of unoccupied parking places drops from 20% to 0%, the share of visitors parking on-street decreases from the observed 20% to 10%, and residents enter the area to find overnight parking.

6.1 Estimating the Effects of the New Parking Facility

The question is now whether the addition of off-street parking places to the existing parking stock can improve the parking situation of the local residents. Since the additional parking places are sold to local residents, they *de facto* reduce the number of drivers looking for on-street parking, assuming no impact on the motorization rate of local residents.

Given the preference of residents to park as close as possible to home, the impact of the additional parking capacity will not be the same all over the Basel neighborhood. To account for the distance between the new parking facility and drivers' destinations, we defined two concentric polygonal rings around the new parking facility with sizes of about 0.7x0.7 km (inner ring), and the remainder of the 1.4x1.0 km Basel area (outer ring). The effects of the new parking garage are estimated for each area separately. In our simulations, we have assumed that all parking places in the new parking garage are purchased by residents of the inner ring, following the reference of residents to park as close to home as possible.

The simulation encompasses the period 17.00-21.00h, during which visitors leave and residents enter the area. We focused on two performance indicators: (1) the distribution of search time; and (2) the distribution of air distance to destination. Both are calculated separately for the drivers with destinations in the inner ring and the outer ring. We run the model for a number of scenarios, differing in terms of the size N of the additional off-street parking facility. The base scenario (N=0) is compared to five scenarios, with values of N = 50, 100, 150, 200 and 250.

It is intuitively evident that even the maximal possible capacity of the new parking lot – 250 places – cannot have a large effect on the average parking situation in an area where parking supply is about 650 places below demand (3,500 - 2,850 = 650). The model investigation confirms this: even for a 250 places parking lot and for drivers whose destination is within the inner ring, the decrease in average search time and walking distance for on-street parkers is low, especially during the last hour of the investigated period (20:00-21:00h). The decrease in mean search time is about 15% (from 250 to 215 seconds); the decrease in distance to destination is almost insignificant - 10% (from 150 to 135 meters). Obviously, the effects are even smaller in case less additional capacity is provided. In the outer ring the mean search time and distance to destination show hardly any decrease.

That is, a new parking lot will hardly change the average residents' perception of the parking situation in the area as a whole. The reason for the limited impact of the additional parking supply is evident: with the increase in supply within the inner ring, drivers with destination in the outer ring will park more often

in the inner one, preserving the high demand/supply ratio there. The only residents experiencing and perceiving a real improvement in their parking situation, are the ones who purchase a parking place in the new garage.

6.2 The Size of a New Parking Lot

The construction of large underground multi-story garages for local residents is a hot issue in the public debate in Tel Aviv. To justify the construction, parking in these garages will be charged for, however, the residents' fees would be low, and they can use the garage on a day-to-day basis when need arises. In order to account for the attractiveness of a garage in the city center for visitors, we assumed that only half of the parking places in such a garage would be available for the residents in the evening. Considering the Basel neighborhood as an example, a 1,000 places garage would be close to supply a parking place to those local residents who currently fail to find a free parking place in the area and have to park at the existing, expensive, paid lots overnight, and for additional visitors who want to stay in the area overnight.

To estimate the quantitative consequences of this parking strategy, we compare two scenarios in which 1,000 off-street parking places are added to the Basel neighborhood. In one scenario the places are added as one large lot, in the other scenario they are distributed between four smaller lots of each 250 parking places located at a distance of about 500 m from each other (Figure 3). Note that the parking arrangement in this scenario is different from the scenario discussed above. In the latter scenario, the additional parking capacity was sold to specific residents who gained a dedicated parking place. In the current experiment, all residents can use the additional parking places against a low fee. Given this arrangement, we assumed that drivers continue searching for free on-street parking until (1) the parking lot closest to the destination is within $\mathbf{D_{parking}(t)}$ distance from the destination and the parking lot is closer to the driver in its current position than the destination, or (2) the maximal search time of 10 minutes is reached.

We compare the two scenarios in terms of the number of "long-searchers", i.e. the number of drivers who fail to find a free on-street parking place or a space in the new parking facility within the maximal search time of 10 minutes (and thus go to the nearest new parking lot). Note that drivers aim to park within walking distance from their home location, i.e. within a radius of 350 meter from their destination. The results show that in case of four parking lots, the number of "long-searchers" – those who do not

find a parking place within 10 minutes – varies between 250 and 300 (from about 650 in the current situation). In case one large parking lot is added, the number of long-searchers varies between 400 and 450. This substantial difference is a direct consequence of the revealed tendency to park as close as possible to the final destination (home location). With four parking lots distributed over the neighborhood, it is more likely that the nearest parking lot is at walking distance from the destination and the distance between the parking lot and the home location is *smaller* than the distance between a free on-street parking place and the home location, than in case of one large centralized parking lot. Hence, a smaller share of residents will continue looking for free on-street parking in case additional off-street parking supply is distributed over the neighborhood. That is, despite supplying the same physical place, the problems of cruising, air pollution, and parking rules violations may well remain substantial in case of centralized supply of additional parking in comparison with additional supply distributed over the neighborhood.

Note that this result was obtained while investigating the Basel neighborhood only. The slight (perceived) reduction in parking pressure in the Basel neighborhood may well induce residents of surrounding neighborhoods within the same parking zone to enter the Basel area in search of a free on-street parking space. In that case, the effects of either scenario on the fraction of "long-searchers" may be well lower than the estimates presented here.

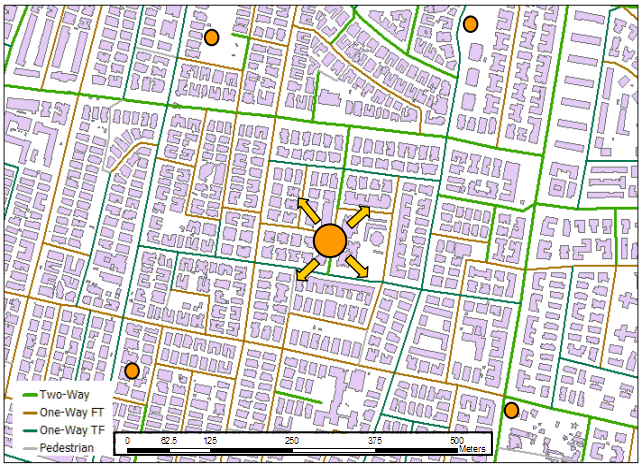


Figure 3. The scenario of one large parking lot of 1,000 places in the center of the neighborhood, versus the scenario of four parking lots of 250 places distributed evenly over the neighborhood.

Being in line with common-sense expectations, the PARKAGENT model thus enables quantifying the impact of different spatial scenarios. As the example suggests, the model could be used to compare various distributions of off-street parking facilities over the city under various conditions and for various user groups and help to determine an optimal solution.

6.3 Reflection on Results

The experimental data and model results presented above demonstrate that the parking pressure as a result of excess demand in the dense areas of central Tel Aviv can be reduced to some extent by adding a network of small parking lots at an appropriate distance between each other. However, this finding should be treated with care. Like in the case of road capacity, more supply may generate more demand for parking. Thus, the improvement in search time and walking distance may be short-term effects. If more residents will purchase cars *because* of the improved parking situation, the long-term effect of additional capacity is actually likely to be negative. Given the high parking pressure and still relatively low level of motorization in central Tel Aviv, it is not unlikely that the small improvement in the parking situation may be enough for the *marginal* resident to purchase a car, or for car-owning rather than carless households to move in.

7. CONCLUSIONS

In this paper, we have presented a spatially-explicit, agent-based, model for parking in the city. Unlike traditionally models, it

simulates the behavior of each driver in a spatially explicit environment. Because of this, the model is able to capture the complex dynamics that can occur between large sets of agents, as well as the impacts of non-homogeneous (road) space. As stressed by Arnott [1], current models are neither able to capture this heterogeneity, nor to estimate its possible impacts.

The small case-study presented in the paper provides a window on the possibilities of PARKAGENT. Instead of investigating the changes in average search time as aggregated over space, which might be relatively insensitive to changes in parking policy, the agent-based model easily captures the spatially distributed effects of changes in parking supply. Given the disproportional human reaction to extremes [7], this is a major advantage of spatially explicit models over existing parking models.

PARKAGENT's ability to simulate the complex dynamics of the parking system in detail *and* generate data about the system performance for different groups of drivers is especially important in saturated parking situations. In such situations, with an instantaneous demand/supply ratio essentially varying around one or even substantially exceeding one, averages are unlikely to capture the essential performance of the parking system due to the inherently uncertain nature of the car parking system [11]. Parking management is especially called for in saturated situations, where it is always hard to foresee the effects of interventions. In this situation, high-resolution, spatially explicit, models are able to provide details of the distribution of key parameters like search times and walking distances that result from different policy scenarios.

Given these advantages, geosimulation thus seems to be a promising tool for urban decision-makers. The ultimate goal of the model is to help decision-makers propose, assess, and evaluate spatially adaptive policy measures, including optimal pricing for on-street parking over a heterogeneous city.

8. ACKNOWLEDGMENTS

The model has been developed within the framework of a research project sponsored by the Municipality of Tel Aviv. The authors would like to thanks the municipality for their cooperation and support.

9. REFERENCES

[1] Arnott, R. 2006. Spatial competition between parking garages and downtown parking policy. Transport Policy 13, 458–469.

[2] Benenson, I., Aronovich, S. and Noam, S. 2005. Let's Talk Objects: Generic Methodology for Urban High-Resolution Simulation. Computers, Environment and Urban Systems 29, 425–453.

[3] Benenson, I. and Torrens, P. M. 2004. Geosimulation: automata-based modeling of urban phenomena. London, Wiley.

[4] Bonsall, P. and I. Palmer 2004. Modelling drivers' car parking behaviour using data from a travel choice simulator. Transportation Research Part C 12, 321–347.

[5] Carrese, S., E. Negrenti and B. B. Belles (2004). Simulation of the Parking Phase for Urban Traffic Emission Models. TRISTAN V - Triennial Symposium on Transportation Analysis, Guadeloupe.

[6] D'Acierno, L., Gallo, M. and Montella, B. 2006. Optimisation models for the urban parking pricing problem. Transport Policy 13, 34-48.

[7] Gigerenzer, G. and Goldstein, D. G. 1996. Reasoning the Fast and Frugal Way: Models of Bounded Rationality. Psychological Review 103, 4, 650-669.

[8] Lam, W. H. K. and Li, Z.-C. 2006. Modeling time-dependent travel choice problems in road networks with multiple user classes and multiple parking facilities. Transportation Research Part B 40, 5, 368-395.

[9] Nagel, K. and Schreckenberg, M. 1992. Cellular automaton model for freeway traffic. J. Physique I (Paris) 2, 2221-2229.

[10] Shoup, D. C. 2006. Cruising for parking. Transport Policy 13, 479-486.

[11] Thompson, R. G. and Richardson, A. J. 1998. A Parking Search Model. Transportation Research Part A 32, 3, 159-170.

[12] Dell'Orco, M. and D. Teodorović 2005. Multi agent systems approach to parking facilities management. N. O. Attoh-Okine and B. M. Ayyub (eds.) Applied Research in Uncertainty Modeling and Analysis. New York, Springer: 321-339.

MADARP: A Flexible Agent Architecture for Passenger Transportation

Claudio Cubillos
Pontificia Universidad Católica de Valparaíso
Av. Brasil 2241
Valparaíso, Chile
claudio.cubillos@ucv.cl

Franco Guidi-Polanco
Pontificia Universidad Católica de Valparaíso
Av. Brasil 2241
Valparaíso, Chile
franco.guidi@ucv.cl

Claudio Demartini
Politecnico di Torino
Cso. Duca Degli Abruzzi 24
10129, Torino, Italia
demartini@polito.it

ABSTRACT

This work describes the development of a distributed agent-based application devoted to the flexible transportation of passengers. The system considers the planning and control of trip requests comming from clients. The underlying 3-tier agent architecture, named MADARP, provides a set of base agents organized around an interface, a planning and a service layer. Interface agents provide GUIs for the interaction with customers and drivers, while planning agents make use of a distributed version of an improved insertion heuristic called ADARTW for the scheduling of passengers' trips. They make use of the contract-net protocol as base coordination mechanism plus a specific transportation domain&communication ontology. The agent application was designed with the agent-oriented software engineering methodology (AOSE) PASSI and implemented over the JADE agent platform. Performance tests were carried out, to evaluate the application's planning capabilities, by analyzing diverse clients' arrival rates while varying the number of hosts.

1. INTRODUCTION

The need to cover more diffuse travel patterns, varying periods of low demand, city-peripheral journeys, as well as commuting trips often make conventional public transport systems unable to guarantee the level of service required to address the user needs. The use of Demand-Responsive Transport services (DRTS), where routes, departure times, vehicles and even operators, can be matched to the identified demand allows a more user-oriented and cost effective approach to service provision.

For this approach to success it is vital an appropriate Information Technology (IT) application that adequately supports the communication and interaction among the diverse involved actors and systems. Under such scenario, software agents leverage as an interesting paradigm to design and implement this kind of software application. Software agents are defined as autonomous entities capable of flexible behavior denoted by reactiveness, pro-activeness and social ability [18]. Multiagent systems (MAS) consist of diverse agents that communicate and coordinate generating synergy to pursue a common goal. This higher level of abstraction has allowed agents to tackle the increasing complexity of nowadays open software systems.

The present work describes the development of an agent-based application for the planning and control of a passenger transportation system under a flexible approach. It gives continuity to our past research [5] on heuristics for solving the scheduling of passenger trips.

In particular the paper covers the description of the overall MAS architecture for then detailing the agents and interfaces involved from the Customer, Vehicle and Transport Enterprise sides. The used underlying optimization problem and scheduling heuristic are depicted together with some performance tests on the planning capabilities of the system.

2. RELATED WORK

Regarding the state-of-the-art research in the field of transportation scheduling, cluster-first and route-second planning techniques have been widely covered. Borndörfer et al. [2] presented such a two-phase approach applied to several instances provided by an operator in Berlin. A constructive method was presented by Madsen et al. [12] in a package named REBUS, to be used by Copenhagen Fire Fighter Services for the transportation of the elderly and handicapped.

Local search techniques have also been applied by Healy and Moll [8] presenting a local search variant based on a strategy called sacrificing, which consists of biasing the search in the direction of solutions with larger neighborhoods of feasible solutions.

In [17] Toth and Vigo have worked on a tabu threshold post-optimization procedure to improve their parallel insertion procedure. A reactive tabu search heuristic for Pickup and Delivery Problem with Time-Windows (PDPTW) was developed by Nanry and Barnes [13] where solutions that violate time-window and vehicle capacity constraints are allowed during the search. More recently, Cordeau and Laporte [4] have developed a tabu search heuristic for the multi-vehicle Dial-a-Ride Problem (m-DARP).

On the other hand, Agent research in the transportation domain has deserved an increasing interest. A Bus-holding control system was proposed by Jiamin et al. [10], which tackles the coordination of multiple lines of fixed-route buses and the different stops, seeking the global optimality. In their approach a MAS negotiation between a Bus Agent and a Stop Agent was conducted based on marginal cost calculations.

In Urban Traffic Control (UTC) systems, Ou [14] presented a UTC which adopted MAS technology based on recursive modeling method (RMM) and Bayesian learning. Ferreira et al. [7] presented a multi-agent decentralized strategy where each agent was in charge of managing the signals of an intersection and optimized an index based on its local state and "opinions" coming from adjacent agents. Agent-based systems devoted to Vehicle Routing (VRP) are presented in [11] and [15]. Both make use of the Contract-Net Protocol (CNP) for the assignment of rides. In addition, [11] uses a stochastic post-optimization phase to improve the result initially obtained. In [15] is presented the Provisional Agreement Protocol (PAP), based on a Extended CNP and de-commitment techniques.

Finally, none of the above solutions tackles the passenger transportation problem under a flexible approach and at the best of our knowledge no similar systems have been found in literature.

3. FLEXIBLE PUBLIC TRANSPORT SERVICES

Flexible or Demand Responsive Transport (DRT) services can be seen as an element of a larger intermodal service chain, providing local mobility and complementary to other conventional forms of transportation (e.g. regular buses and trams, regional trains). In this context, DRT provides a range of Intermediate Transport solutions, filling the gap between traditional public bus services and individual taxis.

The DRT service can be offered through a range of vehicles including regular service bus, mini-bus, maxi-vans, buses&vans adapted for special needs and regular cars. The use of each vehicle type depends on the transport service to offer, the covered area and the target users. The aim is to meet the needs of different users for additional transport supply. The use of flexible transport services, where routes, departure times, vehicles and even operators, can be matched to the identified demand allows a more user-oriented and cost effective approach to service provision. The adaptation of transport services to match actual demand enables cost savings to the operators, society and passengers.

With respect to process implementation and management, the flexibility of the system is expressed along two main directions: on one hand, users of DRT systems must be provided with user-friendly instruments for accessing the services (such as information, reservation, query update) in several different flexible ways (the so called "anywhere and anytime" access). On the other hand, the organization providing flexible services must be itself flexible, with the capability of managing dynamic relationships in a pool of transport resources (vehicles), which may sometimes have

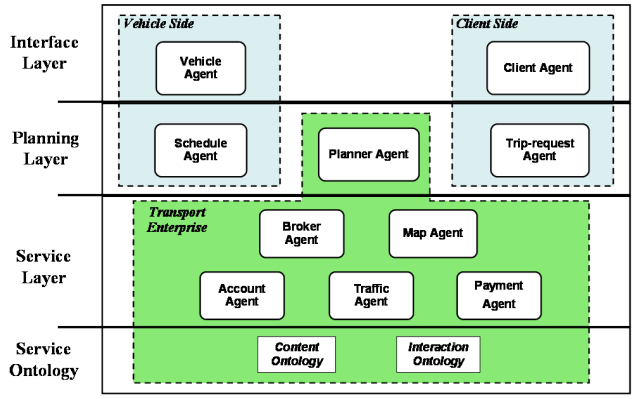


Figure 1: The multiagent transportation architecture

to change to better adapt the transport supply to the dynamic demand.

4. THE AGENT-BASED TRANSPORTATION SYSTEM

In Figure 1 is summarized the agent architecture. A first view shows how the underlying agents are grouped around functional layers to provide a coherent service. While another view shows the architecture from the perspective of the different actors involved, identifying the agents pertaining to each one.

Turning back to the first view, the diagram shows four layers that group the agents and structures according to the functionality provided. The Interface layer connects the system with the real world, providing agents capable of connecting the different actors (clients, vehicle operators) with the system. The Planning layer contains the agents devoted to perform the trips processing and planning in a distributed way. The Service layer supports the above layer providing different complementary functionalities needed for managing a complete transportation service. Finally at the bottom, the Service Ontology provides a means to integrate and make interacting the different agents and actors from the upper layers in a transparent and coherent way.

In the architecture, the control is distributed across the different layers. In general terms, the interface agents provide the input and monitoring signal, for the planning agents to adjust the vehicle's planification. The service agents support these procedures providing with the required information for the re-planning process and the ontology offers the concepts and formalizations for carrying out the control interactions.

The second view provided by Figure 1 shows the three main actors involved in the transportation chain. They correspond to the vehicles, the clients and the transportation enterprise. Each of them is modeled in terms of agents. Consequently, each vehicle actor is represented by a Vehicle agent and a Schedule agent. In a similar way, each client is characterized by a Client agent and a Trip-request agent. In both cases, the pair of agents is tightly coupled as they are modeling different aspects of the same real entity. The

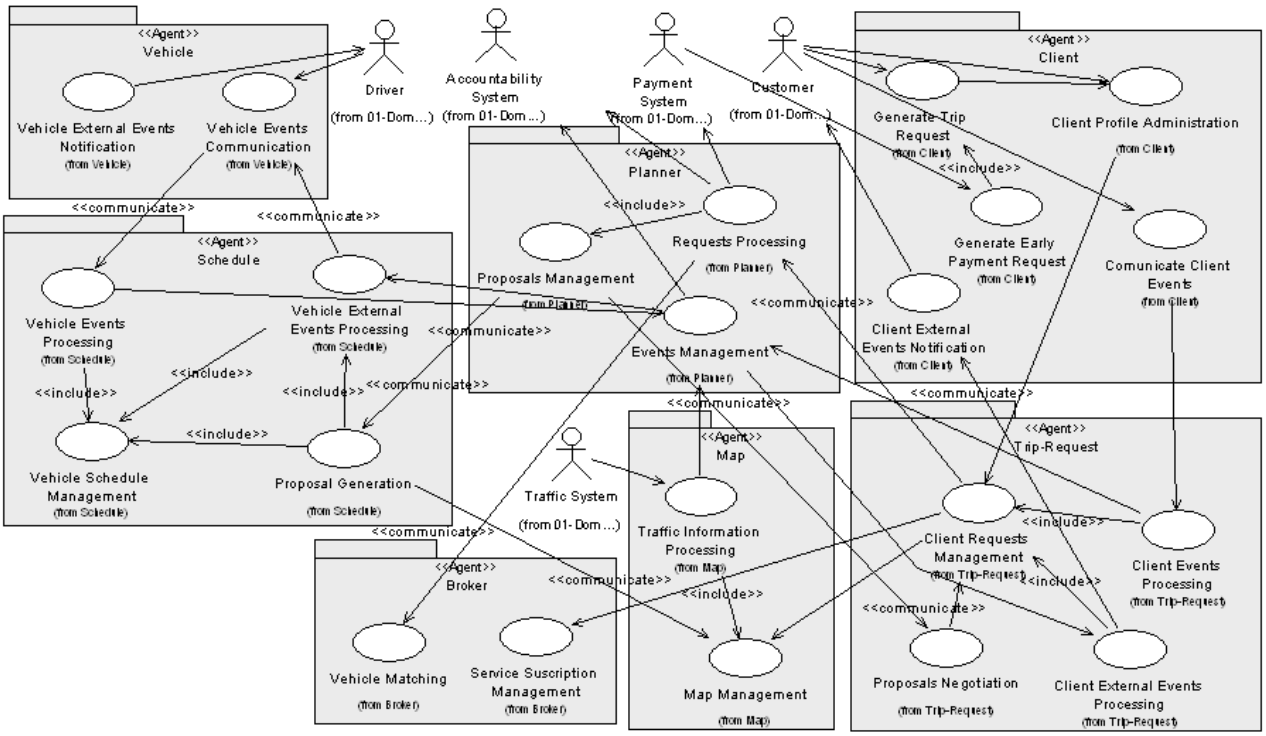


Figure 2: Portion of the Agents' Identification Diagram

third actor is the transport enterprise, which is built up by a series of agents and structures that provide support to diverse services related with the planification and control of the passenger transportation service.

The system was designed following the Process for Agent Societies Specification and Implementation (PASSI) which is made up of five models containing twelve steps in the process of building a multiagent system. Please refer to [3] for a more detailed description on the whole PASSI methodology. Models were developed with the PTK (Passi Toolkit add-on for Rational Rose) and was implemented over the Jade Agent Platform[1], which provides a full environment for agents to work.

The PASSI methodology starts capturing the system's requirements through use cases, for then grouping them together to conform the agents. The diagram in Figure 2 shows part of the use cases and agents involved in the system. Due to space restrictions some service layer agents were expressed as actors.

By starting from the transport operator side, we find the Vehicle, which is an interface agent (with a GUI) in charge of providing the monitoring of the route-schedule planned for the vehicle. In addition, it can inform the Driver about any changes to the initial schedule and can be used by him to inform any eventuality (e.g client no show, delay, detour, etc) that may happen regarding the trip and the customers. In particular its interface has been designed to work on-board the vehicle through a touch screen. This agent will be further detailed on a next section.

The Schedule agent is the one in charge of managing the trip plan (work-schedule) of the vehicle. In addition, the agent is also responsible of making trip proposals upon Planner request and in case of winning will have to update its actual plan to include the new trip. Upon changes (due to vehicle or client events) informed either by the Vehicle or the Planner agent, the Schedule agent will update the plan and reschedule the remaining requests. The Schedule agent encapsulates the underlying optimization algorithm for scheduling the trips of the vehicle. In our system Schedule agents implement a distributed version of a well known greedy insertion algorithm called ADARTW (Please refer to [6] for further details).

The Client is the second interface agent with a GUI, providing the connection between the end-user (Customer) and the transportation system. Through it, the Customer can request a trip by giving a description of the desired transportation service by means of a *Trip Request Profile*.

Other relevant agent is the Trip-Request, which acts as a proxy representing the Customer in the process of contracting a transportation service. In fact, the trip-request agent is involved in all the interaction of the Customer (through the interface agent) with the transportation system. Its activities regard the management of the client transportation requests, including any negotiation or selection of proposals coming from the Planner, together with processing any events generated by the Customer or by the system. As residing on a device with more processing power (such as a PC), this agent may have diverse degrees of autonomy for taking decisions on the trip proposal to choose and how to

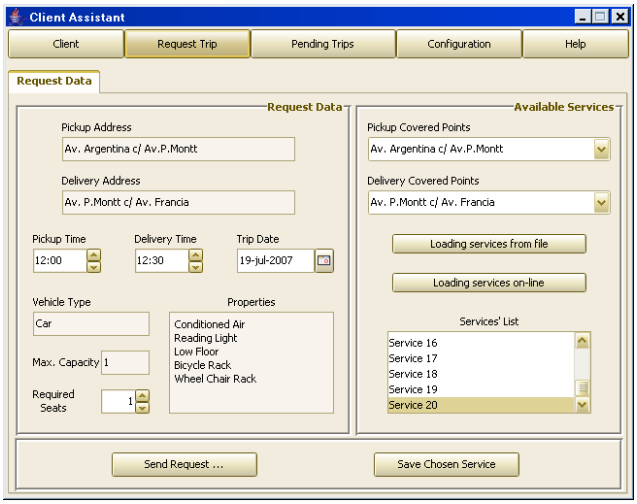


Figure 3: Client agent GUI showing the "Request Data" tab in the "Request Trip" menu

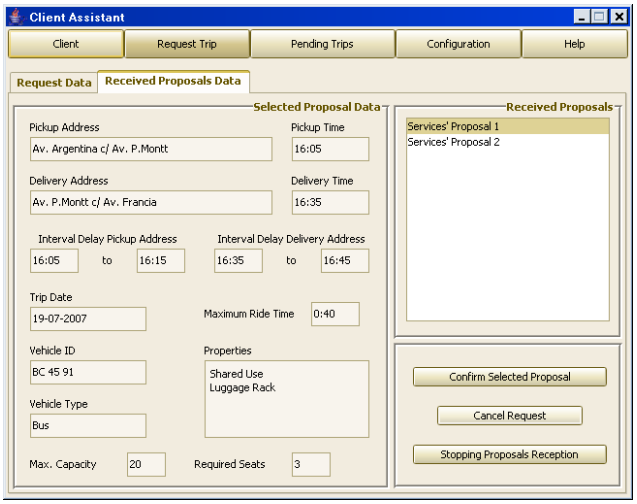


Figure 4: Client agent GUI showing the "Received Proposals Data" tab in the "Request Trip" menu

react when faced to eventualities.

Finally, the Planner agent processes all the customers' requests coming through their Trip-request agents. It initiates a contract-net (CNP) [16] with the Schedule agents and manages all the arrived proposals. It is also in charge of managing events that may affect the trip services already contracted and scheduled. The remaining actors correspond to supporting agents or sub-systems from the service-layer that interact with the agent society, such as the broker, responsible for the initial service matching, and the map, providing times and distances, among others.

4.1 The Customer Side

As stated before, the Client is an interface agent devoted to the Customer-System interaction. In principle, this trip-client assistant may reside on diverse devices (e.g PC, PDA, mobile phone) in order to allow a more flexible and pervasive access to the transportation system. In our prototype, has been developed a Client agent for PC, remaining the versions for more restricted devices as future work. In this sense, it is important to highlight that all the complex processing or decision-making (if delegated by the Customer) has been attached to the Trip-request agent in order to lightweight the Client (the interface agent). In the following Figure 3 a screenshot of the Client agent GUI is shown, detailing the tab that appears when initiating the request of a trip. In the "Request Data" area, on the left, is asked all the information necessary to detail a transport service request under the demand-responsive considered scenario. This regards the date, the pickup and delivery points (addresses), the corresponding times and other specific information such as the required seats and diverse vehicle characteristics.

On the right hand, the available transport services are deployed, showing for each selected service the covered area in terms of street intersections. The services' list can be imported from the system (on line) or from a local file. At the bottom, the Customer can send the trip request and save the services' list.

The following Figure 4 corresponds to a screenshot of the Client agent GUI detailing the "Received Proposals Data" tab which appears as an answer after sending the request for a trip. In this form are displayed all the transportation alternatives found to be capable of performing the service. The list of alternatives is on the right-hand box and by selecting on each of them the left area (Selected Proposal Data) displays the details of such proposal. The data involved concerns the address and requested time for pickup and delivery. In addition, a time window is specified for the pickup and for the delivery in order to make more flexible the service and tackle possible differences with the original schedule.

Other relevant data provided regards the vehicle ID and type, the required seats, together with diverse specific properties, such as the capacity, bicycle rack, shared/individual use, among others. It is important to mention that all the concepts involved in the specification of services make part of a Service Ontology specific for this transportation domain (for further details on the ontology please refer to [6]).

The PASSI methodology used for the modeling considers a *Task Specification* step. In this activity the scope is to focus on each agent's behavior, decomposing it into tasks which usually capture some functionality that conforms a logical unit of work. Therefore for each agent an activity diagram is developed, containing what that agent is capable of along the diverse roles it performs. In general terms, an agent will be requiring one task for handling each incoming and outgoing message.

In the following Figure 5 a portion of the Task Specification Model for the Client Agent is depicted. The diagram shows six tasks that constitute the main Client agent capabilities devoted to the process of requesting a transportation service. The *SendQueryAvailableService* task handles the request from the Customer to search for available services and triggers the *ManageClientQuery* task of the Trip-Request agent which is in charge of requesting the Broker for pos-

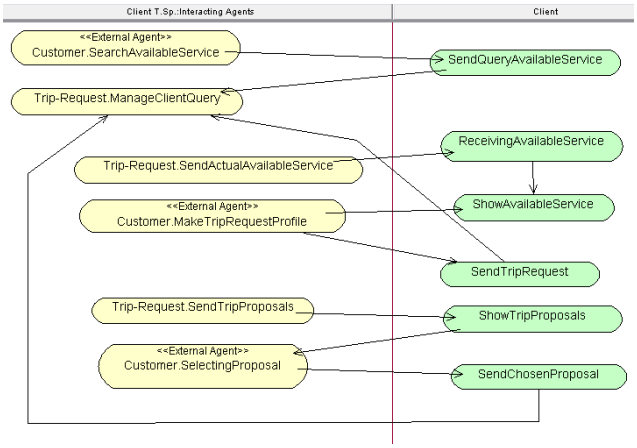


Figure 5: Part of the Task Specification Model for the Client Agent, showing the flow of tasks involved in the trip request processing

Figure 6: Vehicle agent GUI showing the main screen with the itinerary

sible transportation services available. These are returned by the *SendActualAvailableService* task of the Trip-request and is received by the *ReceivingAvailableService* task of the Client which processes and decodes the ACL message and forwards the services' list to the *ShowAvailableService* task responsible for displaying the list in the proper form as already shown in Figure 4 right-hand box.

The Customer, when making a trip Request Profile (see Figure 3), can browse on the available services (after loading them) in the right-hand area calling to the *ShowAvailableService* task or can send the request (by pressing the button) after filling the left-hand information, calling the *SendTripRequest* task. This Client's task is responsible for sending the *Request Profile* to the Trip-Request, being handled by its *ManageClientQuery* task, which on its turn will forward the request to the Planner.

The Trip-request agent will receive from the Planner the trip proposals coming from the different Schedule agents of the vehicles and its *SendTripProposals* task will send them to the Client. On its turn, the Client will receive and handle the proposals through its *ShowTripProposals* task, also responsible for displaying them on the appropriate form area as already shown on Figure 4.

In this way, the Customer will be able to select the best alternative according to his preferences and will click the "Confirm Selected Proposal" button on the GUI (see Figure 4 first button lower-right corner). This action will call the *SendChosenProposal* task of the Client responsible of forwarding the selected proposal to the Trip-request agent, which on its turn will forward it to the Planner, who is in charge of communicating the proposals' acceptance/rejection to the diverse Schedule agents that made bids.

4.2 The Vehicle Side

As mentioned earlier, the Vehicle agent constitutes an interface agent for the Driver - Transportation System interaction. From the Agents'Identification Diagram of Figure 2, it is possible to see that the Vehicle is responsible for allowing the communication of incoming events to the Customer and of vehicle events to the transportation system.

The following Figure 6 shows a screenshot of the Vehicle agent GUI, detailing the actual vehicle itinerary with expected times. On a first view, we can realize that the layout is minimalistic with simple shapes as buttons. This is because the Vehicle agent is intended to be on-board the transportation vehicle (car, van, maxi-taxi, etc.). Hence, the interfaces were developed to be used in touch screens.

On the left side appears the timetable, providing the expected times (e.g. 10:00, 10:30, 11:00 and so on) of the places to visit (either pickup or delivery points). These can be scrolled up or down with the square buttons in the lower part.

On the right hand the interface is divided in three. A header on the top, showing the present date and time plus a square led that blinks when a change to the itinerary is carried out by the system and needs to be communicated to the driver. On the middle-right are shown the details of the entry selected on the left hand (the 10:00 in this particular case). It provides the destination address to reach, the time limit for departure in that place (10:05) for not arriving late to the next destination (at 10:30 in this case) and the number of passengers that go up and down in this stop. Additionally, it is provided the best route in order to arrive from the actual position of the vehicle to the required destination.

Finally on the footer part, the interface deploys three touch buttons; the first allows to confirm passengers presence at the stop, the second to inform a delay or anticipation with respect to the schedule and the third one to turn back to the main menu.

The Task Specification Model for the Vehicle agent is depicted in Figure 7. This activity diagram contains five tasks that specify the labour carried out by the agent. The task *ObtainEventData* manages the notification of events received from the Driver when touching the *Inform Event* button

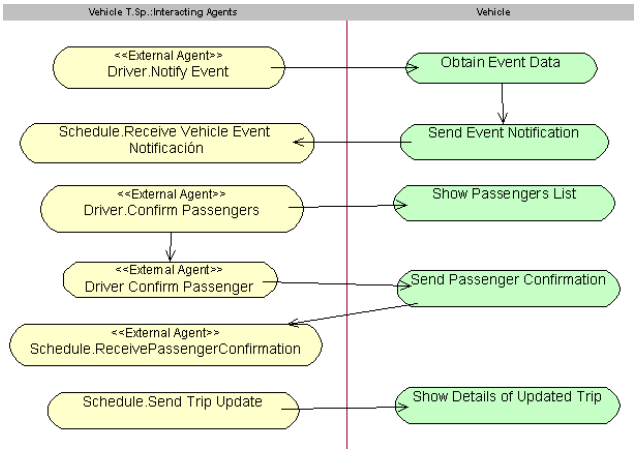


Figure 7: Task Specification model for the Vehicle Agent

of the Vehicle GUI (on Figure 6). The task handles the GUI processing of the notification at low-level, the notification details and forwards them to the *SendEventNotification* task. This task is responsible for translating the event notification and related data into an ACL message in alignment with the transportation domain ontology and finishes by sending to the Schedule agent the ACL message with the given event notification. The *Inform Event* button displays another screen that allows to notify a delay due to a detour, a traffic jam, or a vehicle breakdown.

On each stop the driver must confirm to the transportation system the presence or absence of clients (*Confirm Passengers* button on Figure 6). For this, the driver's passenger confirm action is managed by the *ShowPassengersList* task which is responsible for displaying other tab with a detailed list of the inbound and outbound passengers. After the list is displayed, in the case of inbound customers the driver can confirm the presence or absence on each particular case. This action is managed by the *SendPassengerConfirmation* task which takes the responsibility of taking the client details and sending the Schedule agent an ACL Message with the notification.

Finally, the schedule through its *SendTripUpdate* task informs the Vehicle agent about changes in the itinerary. These messages are handled by the *ShowDetailsOfUpdatedTrip* task of the Vehicle agent. It is in charge of notifying the driver about a change by blinking the upper-right square led on the screen (on Figure 6) together with updating the timetable in order to reflect the changes.

4.3 The Transport Enterprise Side

The Planner is a key agent in the system's architecture. It processes all the clients' requests coming through their Trip-request agents. Therefore it manages all the proposals for a given trip request coming from the diverse Schedule agents representing each available vehicle. It is also in charge of managing inbound and outbound events comming from vehicles and customers. Such events regard the monitoring of vehicles and the modification or cancellation of a trip request.

Upon diferences in the planning (due to breakdowns, traffic-jam, etc) the Schedule agent re-plans. In the case of having an infeasible trip request (mainly due to the time-window restrictions), it informs the Planner agent about the situation. The Planner makes a call for trip-proposals to try reallocating the request in other available vehicle. In any case, the result is informed to the corresponding Trip-request agent, which depending on its degree of autonomy will process the alternatives and take a decision or will inform the client about the change. This change may imply a different vehicle processing the trip only or also a delay or an anticipation of the pickup and delivery times defined previously.

Besides the Planner Agent, there is a whole set of service agents collaborating to give support to the different required functions, such as the matching of request to vehicles, the geographical data access, the accountability of the transactions and the service payment among others. From them, the most critical ones from the planning and control point of view are the Broker and the Map agents.

The Broker's main role is to know which transportation services are available and their characteristics. In addition is able to analyze those service characteristics upon planner request. It provides a publish/subscribe infrastructure that allows vehicles to enter or leave the system freely and allows clients to query the system for available transport services.

It is important to mention that the service profile gives a static description, that is, the description does not take into account the actual state of the vehicle while working. Some characteristics declared in the profile depend on the state (schedule) of the vehicle and hence, are not updated. For example, the service profile can specify that the vehicle has 4 places for wheel chair use. But this information needs to be checked, as it is possible to have all of them used during a route interval. As this information is dynamic, because it depends on the actual vehicle schedule, a further check needs to be performed afterwards in the planning layer, by the schedule agent when making a bid.

The Map agent represents the geographic area being considered where it can be a zone, a city or a part of it. The Map provides the enterprise with a series of information regarding the actual zone being covered such as localization of addresses and stops, street names and distances between localizations, among others. The map agent can be connected to a Geographic information system (GIS) to provide such information.

In our system the Map stores a graph with nodes representing the geographic area under coverage. The distance (km) and time (min) required to go from one node to each other is registered. The measure as distance is likely invariant, while the time measure representation can vary greatly along the day, specially on rush hours. Therefore the map agent can receive updated data from a traffic agent or other external sources of traffic information, allowing the Map to notify the Planner about changes on estimated travel times.

5. THE SCHEDULING PROBLEM

As stated before, during the planning process schedule agents make proposals of trip insertions which are managed by the

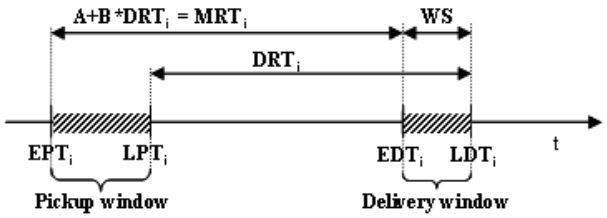


Figure 8: Time-Windows model for clients pickup and delivery intervals

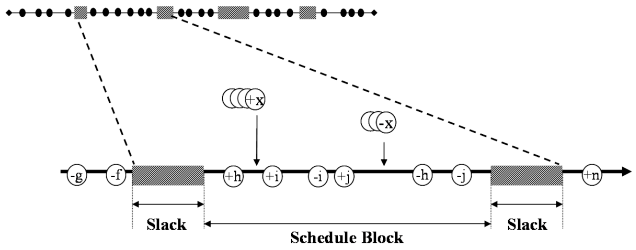


Figure 9: Work-schedule used by vehicles, consisting in sequences of schedule blocks and slacks

planner. Therefore, each of these agents contains a scheduling heuristic to search in the state space for suitable alternatives. The underlying optimization problem and solution heuristic is explained in the following.

From the Operational Research (OR) perspective, the flexible transportation of passengers has traditionally been mentioned in literature under the name of Dial-a-Ride Problem (DARP). In the DARP problem, users formulate requests for transportation from a specific origin (pickup point) to a specific destination (drop-off/delivery point). Transportation is carried out by vehicles that provide a shared service in the sense that several users may be in the vehicle at the same time. The aim is to design a minimum-cost set of vehicle routes serving all requests. Our present implementation of DARP has included the following modelling assumptions:

Passenger Trip Duration. When dealing with passenger transportation is usual to add a limit to the length of the client's journey aboard a given vehicle. This restriction is often mentioned in literature under the name of Maximum Ride Time (MRT) and is usually proportional to Direct Ride Time (DRT), the time needed for the trip but without any deviations (shortest path). Therefore, the MRT has an specific value for each client.

Time Windows. There are different models for constructing the time windows. In our considered variant with Time Windows (DARPTW), customers specify the arriving time, becaming the Latest Delivery Time (LDT). The Latest Delivery Time (see Figure 8) constitutes the upper bound of the delivery time-windows $[EDT_i, LDT_i]$. The majority of real-life pickup and delivery problems are time restricted in a tight or loose way. In our case this is controlled by the systems' parameters A, B and WS allowing to vary the service Quality level to be provided. The model also considers a pickup time-window, the pair (EPT_i, LPT_i), where $EPT_i = EDT_i - MRTi$ and $LPT_i = LDT_i - DRT_i$. In this a way, a vehicle serving the customer i must reach the pickup and delivery points within the time-windows specified for i.

Multiple Vehicles. If the service will be done by one vehicle, the corresponding problem is called the single-vehicle variant of the problem (1-DARP). If there is a fleet of vehicles available for the service, the problem is known as the multi-vehicle variant of the problem (m-DARP) which corresponds to our case.

Multiple Depots. In a number of environments, not all vehicle routes start from and end on the same depot, especially when dealing with multiple vehicles.

Vehicle Route Duration. This formulation allows for constraints on the lengths and/or durations of vehicle tours. For example, such considerations arise from constraints on the geographic coverage of a given vehicle, its refueling requirements and restrictions on the drivers' duty day (e.g. different shifts or time-blocks, lunch breaks) among others.

Dynamic Model. Static formulations assume that customer demand is known ahead in time (e.g. models assuming "advance reservations"). In contrast, in dynamic models DARP (D-DARP), new customer requests are eligible for immediate consideration, requiring revisions of already established routes and schedules. In addition, dynamicity can include delays or cancellations due to traffic-jams, accidents, vehicle breakdowns or simply a client no-show situation, all of which imply a re-planning of the original routes.

5.1 Greedy Insertion Heurisitc

The scheduling algorithm used by schedule agents is based on an improved and distributed version of the ADARTW algorithm [9], a constructive greedy heuristic. Due to the dynamic nature of the problem being tackled it was necesary to make use of a solver fast enough to provide results within seconds rather than minutes, orienting the choice towards this kind of heuristics.

The algorithm finds all the feasible ways in which a new customer can be inserted into the actual work schedule of a vehicle, choosing the one that offers the maximum additional utility according to a certain objective function. Figure 9 shows a schedule block that serves 3 customers (h, i, j) while evaluating the insertion of a fourth one (customer x). Each of them has their pick-up (+) and delivery (-) stops respectively. The search must include all the schedule blocks contained in the vehicle's work-schedule. In a block with already d stops (2 per customer) there are (d+1)(d+2)/2 possible insertions, considering that the customer's pickup must always precede his delivery and that is not possible to pickup a client in one block and deliver him in another (because of the block's definition).

Once we have found that a possibility of insertion is feasible, it is necessary to define the actual times for those events, that is, the Actual Pick-up Time and the Actual Delivery Time. This problem is often mentioned in literature as the

scheduling problem, as once the sequence of trips (route) has been fixed the following step is to define the exact position where the sequence will be placed in time without violating the time-windows defined for each client.

Commonly, there will be a time interval in which can be inserted, meaning that the sequence can be scheduled more early or late in time within that interval. Several authors program the actual times as soon as possible for reducing the travel and waiting times of the customers, reason why our implementation does it in this way.

5.2 Work-Schedule

The model used for the vehicles' work-schedules considers that along the day a vehicle can be in any of these three states: at a depot, in travel or inactive. When the vehicle is at a depot means that it has not started its service period or has just finished it. When the vehicle is in travel, means that it is actually going to pickup or delivery passengers generating schedule blocks. As Figure 9 shows, a schedule block corresponds to a sequence of pickups and deliveries for serving one or more trip requests. A schedule block always begins with the vehicle starting on its way to pick-up a customer and ends when the last on-board customer is discharged. The third state is when the vehicle is inactive or idle generating a slack time. In this case the vehicle is parked and waiting to serve a next customer and then begin another schedule block.

Therefore, a complete vehicle's work-schedule will have periods of vehicle's utilization (schedule blocks) and inactive periods (slacks times) in which the vehicle is available and waiting.

5.3 Time-Windows Feasibility

The time-windows feasibility processing is tightly coupled to the work-schedule model. Within the checking algorithm, different restrictions need to be checked for a given potential solution. The most important ones are the time windows, the capacity constraints (on number and type) and the bounds on the duration of clients' ride and of vehicle routes.

This represented a challenging aspect of the work, as in general is difficult to find in literature the used mechanism for tackling this point. In Jaw et al. [9] is described only in general terms and most research papers state a change from the previous work but not its specific implementation.

For a block X with w events representing either a pickup or a delivery of passengers, Jaw's work presents the following calculations representing how much the events can be anticipated/posticipated in time.

$$BUP(X_i) = Min(Min(AT(X_i) - ET(X_i)), SLK_0)$$

$$BDOWN(X_i) = Min(LT(X_i) - AT(X_i))$$

$$AUP(X_i) = Min(AT(X_i) - ET(X_i))$$

$$ADOWN(X_i) = Min(Min(LT(X_i) - AT(X_i)), SLK_{w+1})$$

SLK_0 and SLK_{w+1} represent the (possible) slack periods immediately preceding and following the block respectively.

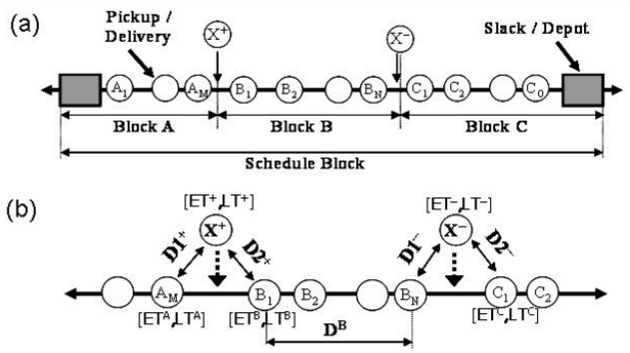


Figure 10: Time-windows feasibility-check procedure

$ET(X_i)$, $AT(X_i)$ and $LT(X_i)$ represent the Early, Actual and Latest Times of the event X_i respectively with $0 < i < w + 1$.

Our developed model is based on the Jaw's calculations on BUP and $ADOWN$ but adds the important idea of intersecting the time windows restrictions along a piece of route, allowing to simplify the processing of the time windows feasibility check and making it possible to evaluate the insertion of the whole client (pickup and delivery) at the same time.

Therefore, in the case of the implemented insertion heuristic the starting point is the schedule block under which to evaluate the insertion of the new client. The Figure 10(a) shows a detailed view when evaluating the insertion of the pickup (X^+) and delivery (X^-) of a client. Between the pickup and the delivery are one or more events separating them and at the beginning (or ending) of the block is a slack or the bus depot. The approach is to divide the schedule block in three sub-blocks A, B and C for the events before the pickup, in between and after the delivery respectively. A special case is when both events are consecutive meaning that the block B includes only the distance from the pickup to the delivery of the new client.

The Figure 10(b) shows the time windows and distances needed for the evaluation and intersection. The interval $[ET_A, LT_A]$ represents the earliest and latest times to which the event A_M can be shifted (anticipated / posticipated) without violating the time window constraints of all the events within its block. A similar thing happens with intervals $[ET_B, LT_B]$ and $[ET_C, LT_C]$ on the events B_1 and C_1 for the blocks B and C respectively. Therefore, is needed to identify the feasible shift up and shift down for each of the three blocks. For the block A are used the BUP and $BDOWN$ of the event A_M as they consider the previous events, while for the block C the AUP and $ADOWN$ of C_1 are needed. For block B is needed the AUP and $ADOWN$ for B_1 but considering only until B_N and not the events on block C as the normal calculations would. Then, for interval $[ET_A, LT_A]$ we have: $ET_A = AT(A_M) - BUP(A_M)$ and $LT_A = AT(A_M) + BDOWN(A_M)$.

A similar thing happens with $[ET_B, LT_B]$ and $[ET_C, LT_C]$. Distance $D1^+$, $D2^+$, $D1^-$ and $D2^-$ correspond to the distances between the nodes indicated by the respective arrows

in the figure. The next step is intersecting the time intervals of the three blocks and the two time windows coming from the new client's pickup and delivery events. This intersection needs to consider the distances separating each of the five intervals. For this reason, a point in the schedule is used as reference and all the intervals are translated to that reference obtaining a single time interval $[ET, LT]$. By using the pickup event (X^+) of the new client as reference point and following Figure 10(b) are obtained:

$$ET = Max(ET_A + D1^+; ET^+; ET_B - D2^+; ET^- - D1^- - D_B - D2^+; ET_C - D2^- - D1^- - D_B - D2^+)$$

$$LT = Min(LT_A + D1^+; LT^+; LT_B - D2^+; LT^- - D1^- - D_B - D2^+; LT_C - D2^- - D1^- - D_B - D2^+)$$

This $[ET, LT]$ interval represents the feasibility area in which to set the new schedule with respect to the reference point. The actual time for the reference point (X^+ in this case) must be set and hence the actual times for the whole schedule block can be calculated as they depend on the fixed distances between one event and another. Defining the optimal place within this interval corresponds to the scheduling problem mentioned before.

6. EXPERIMENT SETUP

As mentioned earlier, the original architecture's planning approach is based in the contract-net (CNP) under self-interested agents. Therefore, Vehicle agents pursued the optimization of the travelling costs (utility function with total slack time and total travel time) and Client agents were oriented towards the maximization of the perceived service quality (utility function with excess travel time and waiting time).

All the tests considered the same geographical net and 20 demand scenarios labeled from U1.txt to U20.txt. Each considered 50 trip requests each, distributed uniformly in a two-hour horizon. For each demand scenario 25 runs were done. Regarding the considered distributed environment, the simulations were carried out over PCs with Intel Pentium 4 of 2 GHz. with 256 MB Ram, connected through a 10/100 Mb. Router.

The following tests focus on the planning capabilities of the architecture. In this sense, the simulations consider an agent devoted to the generation of the Trip-request agents and another devoted to generating the Schedule agents. In addition, a Main agent was in charge of managing all the aspects related to the simulation control, specifically centered on the generation of the agents, request of output data and deletion operations along the diverse runs and scenarios.

The following operational decisions were adopted: 1) the same utility function and scheduling algorithm have been used for all the vehicles, 2) all the clients share the same utility function, 3) the available fleet is of 30 identical vehicles with capacity 20, 4) one depot is used for all the vehicles and 5) in all cases the effectiveness measures (utility variables) were weighted with the same value.

The generation of Trip-request agents (and hence the arrival of trip-requests) to the system follows a Poisson distribution. Then, the time between arrivals distributes Exponential, $E(\lambda)$, with lambda in terms of requests per second. The agents involved in the simulations were the three of the planning layer (Trip-request, Planner and Schedule agents) plus two of the service layer (Map and Broker agents) as Table 1 details.

Table 1: Distribution of agents among hosts over the 3 scenarios

2 Hosts	3 Hosts	5 Hosts	Agents
1	1	1	Map Agent
1	2	2	Trip-Request Agents
2	3	3	Schedule Agents
1	1	4	Planner Agent
1	1	5	Broker Agent

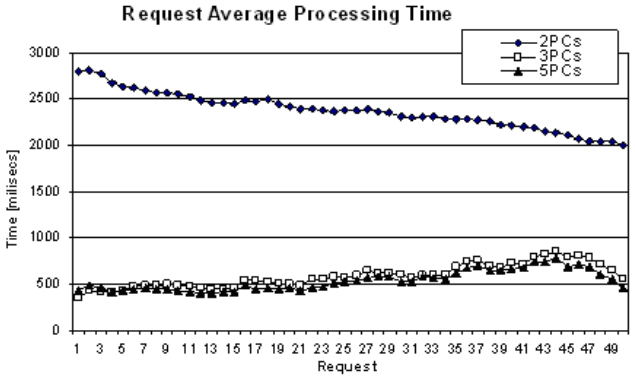


Figure 11: Processing time for trip requests ordered by arrival at Exp(3) under 2, 3 & 5 hosts.

6.1 Results

Figure 11 shows different curves for 2, 3 and 5 hosts' configuration at a $(\lambda) = 3$ arrival rate. A big improvement exists when changing from 2-hosts to 3-hosts configuration, while little improvement is obtained when changing from 3 to 5 hosts. A closer look on how agents were distributed shows that separating Trip-request and Schedule agents from the rest has a big impact on performance, but separating the Planner, Broker and Map agents on diverse host gets only a small improvement in terms of processing time.

A second round considered contrasting the effect of changing the lambda (λ) coefficient over the performance of the planning system when processing a request. In Figure 12 are compared 2 diverse arrival rates; $\lambda = 3$ and $\lambda = 5$ requests per second for the 3-host and 5-host scenarios. The two curves in the lover part correspond to $\lambda = 3$ scenarios while the other two at the top, to $\lambda = 5$.

In contrast, the average quality of the results for all the 20 demand scenarios considered were similar. In fact the number of vehicles used and cost of the solutions provided are not significantly different among the two trip-request rates even when changing the number of hosts.

These results reflect the fact that the system gets much overloaded at a $\lambda = 5$ arrival rate, suggesting a possible

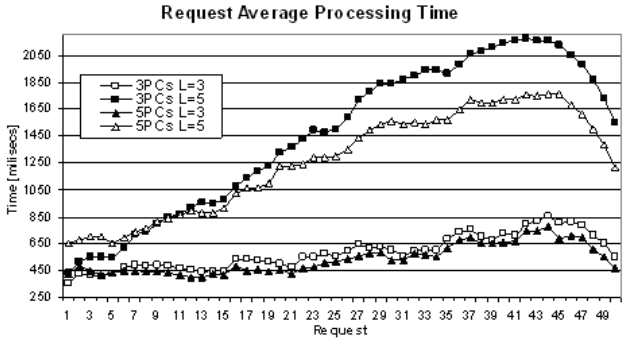


Figure 12: Processing time for trip requests ordered by arrival at Exp(3) and Exp(5) rates, over 3 and 5 hosts.

bottleneck in the architecture. In fact, the Planner Agent constitutes a central point of communication among parties, therefore for bigger-size problems the architecture needs to be scalable.

As a possible solution, the Planner could be replicated on diverse hosts and be organized hierarchically to balance the workload, allowing to improve the overall performance under such scenarios. Of course this remains a matter of further work.

7. CONCLUSIONS

An agent-based software application for managing a flexible passenger transportation systems was described. The undelying architecture provides a transparent and flexible way to make interoperate users, vehicles and support-service providers into a single application.

The methodology used allowed an appropriate level of specification along its diverse phases. The focus was centered on the specification of the main actors involved with the system: Customers, Drivers (Vehicles) and the Transportation Enterprise, providing a concrete idea of interface agents' GUIs while covering the optimization problem involved and the implemented scheduling heuristic.

A project next step involves a field test with a local group of taxis already operating under a shared mode but actually organized in frequency-based route lines. Further work considers to carry out scalability tests while replicating the Planner agent.

8. ACKNOWLEDGEMENTS

Partially funded by the Pontifical Catholic University of Valparaíso (www.pucv.cl), project No. 209.746/2007.

9. REFERENCES

[1] Bellifemine, F. et al: JADE - A FIPA Compliant Agent Framework. C SELT Internal Technical Report, 1999.

[2] Borndörfer, R., Grötschel, M., Klostermeier, F., Küttner, C.: Telebus Berlin: Vehicle Scheduling in a Dial-a-Ride System. Tech. Report SC 97-23, Berlin, 1997.

[3] Burrafato, P., and Cossentino, M.: Designing a multiagent solution for a bookstore with the passi methodology. In 4th Int. Bi-Conference Workshop on AgentOriented Information Systems (AOIS-2002).

[4] Cordeau, J.F., Laporte, G.: A Tabu Search Heuristic for the Static Multi-Vehicle Dial-a-Ride Problem. Transportation Research B, Vol. 37B, 2003, pp. 579-594.

[5] Cubillos, C., Rodríguez, N., Crawford, B.: A Study on Genetic Algorithms for the DARP Problem. Mira, J., Alvarez, J.R. (eds.) IWINAC 2007, Part I. Springer LNCS, Vol. 4527,2007. pp. 498-507.

[6] Cubillos, C., Gaete, S.: Design of an Agent-Based System for Passenger Transportation using PASSI. Mira, J., Alvarez, J.R. (eds.) IWINAC 2007, Part II. Springer LNCS, Vol. 4528,2007. pp. 531-540.

[7] Ferreira, E. D., Subrahmanian E.: Intelligent Agens in Decentralized Traffic Control. IEEE ITS Conference Proceedings, August 2001, USA, pp. 705 -709.

[8] Healy, P., Moll, R.: A New Extension of Local Search Applied to the Dial-a-Ride Problem. European Journal of Operational Research, Vol. 83, 1995, pp. 83-104.

[9] Jaw, J. et al.: A heuristic algorithm for the multiple-vehicle advance request dial-a-ride problem with time windows. Transportation Research. Vol. 20B, No 3, 1986, pp. 243-257.

[10] Jiamin Zhao, Dessouky, M., Bukkapatnam, S.: Distributed Holding Control of Bus Transit Operations. IEEE ITS Conference Proceedings, Oakland - USA, August 2001, pp. 976 - 981.

[11] Kohout, R, Erol, K. Robert C.: In-Time Agent-Based Vehicle Routing with a Stochastic Improvement Heuristic. AAAI/IAAI Int. Conf. Orlando, Florida, 1999, pp. 864-869.

[12] Madsen O.B.G., Ravn H.F., Rygaard J.M.: A Heuristic Algorithm for a Dial-a-Ride Routing and Scheduling Problem with Time Windows, Multiple Capacities, and Multiple Objectives. Annals of Operations Research, Vol. 60, 1995, pp. 193-208.

[13] Nanry, W.P., Barnes, J.W.: Solving the Pickup and Delivery Problem with Time Windows using Reactive Tabu Search. Transportation Research B, Vol. 34B, 2000, pp. 107-121.

[14] Ou, H. T.: Urban Traffic Multi-Agent System based on RMM and Bayesian Learning. Proc. American Control Conference 2000, pp. 2782-2783.

[15] Perugini, D., Lambert, D., et al.: A distributed agent approach to global transportation scheduling. IEEE/ WIC Int. Conf. on Intelligent Agent Technology, 2003, pp 18-24.

[16] Smith, R. G. and R. Davis: Distributed Problem Solving: The Contract Net Approach. Proc. 2nd National Conference of the Canadian Society for Computational Studies of Intelligence. 1978.

[17] Toth, P., Vigo, D.: Heuristic Algorithms for the Handicapped Persons Transportation Problem. Transportation Science, Vol. 31, No. 1, February 1997.

[18] Weiss, G.: Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. MIT Press, Massachusetts, USA. 1999.

A study of collaborative influence mechanisms for highway convoy driving

Majid Ali Khan, Damla Turgut and Ladislau Bölöni
School of Electrical Engineering and Computer Science
University of Central Florida
Orlando, FL 32816–2450
Email: khan@bond.cs.ucf.edu,turgut@eecs.ucf.edu,lboloni@eecs.ucf.edu

ABSTRACT

Convoy driving on highways is a desirable behavior which reduces the risk of highway accidents and makes traffic faster and more fluent. Recent technologies, such as intelligent cruise control devices explicitly facilitate convoy driving by providing a fully automated means for following the previous vehicle. Participating in a convoy, however, requires compromises from the vehicles, such as slowing down to the speed of the lead vehicle; thus many drivers choose not to join any convoy. Collaborative convoy driving systems, based on vehicle-to-vehicle communication, promise to deliver means for the vehicles to influence the speed of the convoy, thus improving its utility. We discuss the mechanisms of convoy participation, including the decision to join and leave the convoy, and the mechanisms through which the vehicles can influence the convoy speed. In an experimental study, we compare three influence mechanisms: the "adapt speed to the leader" mechanism used by human drivers and intelligent cruise control systems and two collaborative influence mechanisms which require vehicle to vehicle communication. We show that the collaborative cruise control methods deliver better macroscopic performance measures: more vehicles participating in convoys, higher average speed and lower number of overtakings.

1. INTRODUCTION

The desired speed of a driver on a highway depends on the capabilities of the vehicle, the driver's driving skills, style, current goal and state of mind, as well his or her assessment of the likelihood of a fine if the vehicles exceeds the posted speed limits. If the vehicles on a highway have a wide spread of desired speed, it leads to a behavior with many lane changes, overtaking, accelerations and decelerations. In practical traffic, it increases the likelihood of accidents, and slows down the traffic by creating bottlenecks.

The ideal traffic behavior would be for all vehicles to travel at the posted speed limit, and to maintain this speed constant, for instance, through the use of a cruise control system. It is also desirable that vehicles position themselves at uniform distance from each other, that is, they form *convoys*. Unfortunately, cruise control systems have slight variations, which make vehicles "creep" closer to each other. In these occasions the drivers need to take control and either take over the vehicle in front or adjust the speed lower. The latter might lead to a chain reaction, where the following vehicles need to make similar decisions. This effectively means that the vehicles have the choice to either (a) adjust to the speed of the slowest vehicle in the convoy or (b) leave the convoy and engage in a series of overtakings.

Many recent vehicles are equipped with an "intelligent cruise control system", which measures the distance from the vehicle in front using a laser or radar and adjusts the speed accordingly. This makes the job of the driver easier, as it automatically performs the slow-down operation, and, in the limits of the previously set preferred speed, it can also perform acceleration. However, the problem still remains that the vehicles will adopt the speed of slowest member of the convoy.

One way to improve this architecture is to create a convoy where the vehicles communicate with each other. Note that communication alone does not change the picture, unless vehicles, in some respect, are responsive to the "common good". For instance, if a convoy of 10 vehicles has a first vehicle whose desired speed is 2mph lower than the followers, the followers might "convince" the leader to increase the speed, rather than performing a series of traffic-disruptive overtakings. In traditional traffic, follower vehicles sometimes pressure the first vehicle by driving closer than the comfort distance of the driver. In general, a driver might choose to collaborate with the convoy as long as the departure from the preferred speed is not too large, as the driver himself benefits from the smoother traffic. In is beyond the scope of this paper to discuss the mechanisms through which the interests of the individual vehicles, the convoy and the general public are reconciled[1].

While the formation of convoys is sometimes an explicitly planned operation, most often it is happening in an ad hoc manner between vehicles whose drivers do not know each other, might not have common goals and can communicate only through indirect means. Convoys are formed and terminated dynamically; their life cycle ranges from tens of seconds to several hours. Vehicles can join and leave, and convoys themselves can split and merge. If we consider the vehicles to be intelligent agents, highway convoy driving is a microcosm of problems including communication (both at networking and semantic level), team formation, leader election, negotiation and planning.

In previous work [6] we have proposed an architecture which managed the formation, creation and splitting of convoys through vehicle to vehicle communication. As a note, our hardware implementation was based on Crossbow MICA2 motes communicat-

[1]One way to assure this is through legislation. Considering the safety advantages of convoy driving, it is possible to allow a higher speed limit for convoys: eg. "speed limit 70mph, up to 80mph when in convoy".

.

ing on the 868/916MHz range. Current efforts around the Dedicated Short Range Communications (DSRC) project and the IEEE 802.11p standardization effort makes it likely that future vehicle-to-vehicle communications will happen in the 5.9 GHz band.

The determining feature of the convoy formation is the *influence mechanism*, the way in which the vehicles influence each other's speed. As we had seen, in convoy driving without inter-vehicle communication the only influence mechanisms possible are the adaptation to the speed of the vehicle in front (with or without explicit adaptations to maintain a following distance). If we assume the existence of inter-vehicle communication, other adaptations are possible. Examples are increasing the speed at the request of the vehicle in the back, decreasing the speed at the request of the vehicle in the back, increasing the speed at the request of the vehicle in the front, and so on.

In [6] we have described an influence mechanism which relies on the Social Potential Fields (SPF) model [9] proposed in the field of mobile robots.

In addition to the influence mechanism, the traffic behavior is also affected by the *convoy participation policy* of the vehicle. This policy governs the choice of the vehicles whether to join the convoy (and, implicitly, to obey the influence mechanism) or to leave the current convoy and either drive alone or join another convoy. For the purpose of this paper we will assume a very simple policy based on *threshold with friction* where the desirability of the convoy is determined by the difference in the speed of the convoy and the desired speed of the vehicle. This policy also adds a cost for the joining and leaving a convoy to prevent frequent defections.

Our experiments with this architecture in [6] have concentrated on the local behavior of several vehicles. As intelligent cruise controls are deployed in more and more vehicles and the possibility of wide scale deployment of collaborative cruise control draws near through the standardization of the vehicle-to-vehicle communication protocols, we are interested in investigating how the deployment of such systems affect the general traffic. We are mainly interested in integrative measures such as the mean velocity of vehicles, the number, size and size distribution of convoys and the number of vehicle overtakings.

The remainder of this paper is organized as follows. We survey related work in Section 2. The convoy formation mechanism and influence mechanisms considered are described in Section 3. We then use these mechanisms in a simulation study involving a large number of vehicles, study the emergent traffic behavior and measure the integrative properties in Section 4. We conclude in Section 5.

2. RELATED WORK

The study of vehicular traffic has attracted the interest of researchers for several decades. One of the schools of thought treats traffic in analogy to various physical phenomena. A thorough overview of proposed models is provided by Chowdhury et al. [3]. One approach is to treat traffic flow in analogy with the hydrodynamic theory of fluids [1]. In this case traffic is seen as a one dimensional compressible fluid; the characteristics of the individual vehicles are not considered, only their density on the road. An alternative approach is the kinetic theory, where traffic is treated as a gas of interacting particles, with each particle representing a vehicle. As molecules in the gas have random movements described by the Boltzmann equation, on its own, this model can not describe the purposeful movements of vehicles. One approach is the Paveri-Fontana model [8] which assumes that each vehicle, in contrast to molecules in the gas, has a desired velocity towards which the actual velocity converges in the absence of other vehicles.

Of a particular relevance to our to our approach are the "car-following theories" of the traffic flow. In these models the traffic is seen as a set of objects interacting under a set of forces analogous to the Newtonian mechanics. Various proposed models make the force acting on the vehicle dependent either on the parameters of the preceding vehicle, or several of the preceding vehicles. Note that the different influence mechanisms in convoy formation (to be discussed later in this paper) can be seen as specific instances of these classes of models.

Another influential approach of traffic modeling uses the language of cellular automata, a representative example being the Nagel-Schreckenberg model [10].

What can the agent community bring to this respected body of research? First of all, the concept that the drivers on the highway are humans with autonomous decision making capabilities. While humans might drive long stretches of road in ways predictable from their environmental conditions, they also frequently exercise their decision making capacity in joining a convoy, accelerating to catch a green light, overtaking to escape an erratic driver and so on. The vehicle-to-vehicle communication systems currently in development will likely change the driving dynamics and their effects needs to be modeled by treating the vehicles as agents.

Although the technical means of implementing vehicle-to-vehicle communication are only beginning to become available, there is already a significant literature in the using agents in the control and modeling of vehicles. Dresner and Stone [4] propose an intersection control mechanism where agent-based reservations replace the traffic lights. They prove the superiority of the approach through simulation and show that the reservation method closely approximates an overpass (which is the optimal, although costly solution for intersection management).

Laumonier et al. [7] work towards a cooperative adaptive cruise control system. The authors propose a reinforcement learning technique for the control of the throttle to maintain the desired inter-vehicle gap.

Girard et al. [5] propose a hierarchical implementation of a control architecture for cooperative cruise control (CACC). Some of the interesting features of their approach includes the ability to switch between various modes of operation depending whether the nearby vehicles are also equipped with CACC-capable devices. In addition, this system also implements cooperative forward collision warning (CFCW), through which the following vehicles receive information about the sudden braking of the vehicle in front.

3. CONVOY FORMATION MECHANISMS

Our interest in convoy formation mechanisms are two-fold. On one hand, we wish to model the driving behavior existing on current highways, as a result of manual driving, traditional cruise control systems and a small minority of vehicles equipped with intelligent cruise control. On the other hand, we are interested in designing new algorithms for the cooperative cruise control systems of near future. Note that for the foreseeable future, vehicles with different level of autonomy will share the same road. In fact, even if a vehicle has an intelligent/collaborative cruise control system, the driver might choose not to turn in on. For the sake of uniform treatment, in the following discussions we will use the term "vehicle" to cover both vehicles under the control of human driver and agents.

There are three different aspects of the participation of a vehicle in a convoy.

- The **decision to join or leave the convoy.** The vehicle can join any convoy in its physical proximity, or it can decide

to drive outside of any convoy. For the sake of a uniform treatment, we will consider the later as the vehicle forming its own convoy.

- The **influence of the convoy on the vehicle**. Once the vehicle has joined a convoy, its driving is influenced by the presence of the other vehicles in the convoy. Most importantly, its speed needs to be synchronized with the speed of the other vehicles. Small, temporary adjustments in speed can be used to achieve the desired following distance / time gap between the vehicles.

- The **influence of the vehicle on the convoy**. In the simplest example, the leading vehicle determines the speed of the convoy, while the other vehicles do not have any influence. As an example of visual communication, a vehicle in the rear might be able to "pressure" the vehicle in the front to increase the speed. In the vehicles are connected through a vehicle-to-vehicle communication system, they will be able to reach a negotiated agreement about the speed of the convoy, following distance, order of the vehicles and other factors.

3.1 Convoy joining policy

In the following we introduce an algorithm for modeling the policy of the agents for convoy joining. This is both an algorithm for practical implementation [6], as well as a model of the human behavior in convoy joining.

The policy we are proposing is based on the measuring of the utility of the different convoys. Whenever a driver needs to make a decision (whether to join a convoy, leave a convoy, or move from one convoy to another) it will evaluate the utility of the convoy and pick the choice with a higher utility. As the utility of a convoy varies in time, the vehicle needs to perform periodic evaluations of the utility of the current convoy. Different vehicles can have different utility functions even if they are part of the same convoy. Vehicles in a convoy, however, need to agree on the same rules for evaluating influences, otherwise the integrity of the convoy can not be maintained.

We assume that the utility of a convoy depends only on the speed of the convoy and the parameters of the vehicle[2]. We assume that a vehicle V_i has a current speed P_i and desired speed D_i. The vehicle also has an upper speed limit H_i (determined by physical or legal factors, or simply by preference) and a lowest accepted speed L_i (normally determined by preference but also by fuel economy considerations).

It is desirable to have a utility function return 0 for convoys whose joining is not feasible for the vehicle and preferably high values for convoys which are close to its desired speed. A simple expression for the utility of the convoy with speed S_i for a vehicle V_j which satisfies this requirement is the following:

$$U = \begin{cases} 1 - \frac{|D_j - S_i|}{D_j} - \lambda \cdot \frac{|P_j - S_i|}{P_j} & \text{if } L_j \leq S_i \leq H_j \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Note that any offered speed that lies outside the lower and upper speed limits has zero utility. Otherwise, the utility of an offered speed is affected by two factors. The *compromise factor* $\frac{|D_j - S_i|}{D_j}$ determines the amount of compromise that the vehicle needs to make to become part of the convoy. It increases with the difference between the desired and the offered speed of the vehicle. Thus, an offered speed that is either higher or lower than the desired speed, will cause the utility of the offer to become lower. The *join cost factor* $\lambda \cdot \frac{|P_j - S_i|}{P_j}$ is the cost of joining convoy C_i, and it is zero if V_j is either currently a member of the convoy or if the offered speed matches the current speed of the vehicle. This factor reflects the need to accelerate or decelerate to join a convoy. In addition, this factor allows us to introduce "friction" in the behavior of the vehicles. By making it expensive for a vehicle to leave a convoy, we can reduce the number of defections and stabilize the convoys. Experimentally, we found $\lambda = 0.1$ to be an adequate value.

[2]A human driver might consider various other factors. A driver might be reluctant to follow a driver whose behavior appears to be erratic, or a large truck obscuring visibility. Drivers might be offended by the bumper stickers on the previous car.

3.2 Influences among the members of the convoy

Let us now consider the next component of the convoy formation, the influences among the members of the convoy. We will consider three influence strategies.

Influence Strategy ASL (Adjusting to the speed of the leader): This is the traditional case of convoys formed by human drivers, or vehicles with intelligent cruise control systems. The advantage of this approach is that joining the convoy does not require negotiation. Furthermore, vehicles leaving or joining the convoy will not change the convoy speed. This means that the utility of the convoy remains the same for a vehicle throughout the lifetime of the convoy, increasing the stability of the convoy. The only reason for a vehicle to reconsider its convoy joining decision is if the convoy passes next to another convoy with a higher utility for the particular vehicle.

The disadvantage, however, is that the speed of the convoy is dictated by its slowest vehicle.

Influence Strategy AVG (Average desired speed):

In this influence strategy the protocol calls for the participating members to compute the average of their desired speeds, and then all members adjust to that speed. Naturally, this requires communication. The speed needs to be recalculated every time a vehicle joins or leaves the convoy. This phenomena is mitigated somewhat by the fact that the vehicles which join will likely have a desired speed close to the current speed of the convoy. The change in the speed of the convoy, in addition to the inconvenience of accelerating or decelerating, also poses the potential problem that by changing the utility of the convoy, it can reach a point where it is not worth for a given vehicle to remain in the convoy. If the vehicle decides to leave the convoy, this would lead to yet another speed adjustment, which, on its turn, might lead to further vehicles leaving. This way, a convoy spontaneously splits into a slower and a faster convoy.

Influence Strategy SPF (Social potential fields): Social potential fields [9] are a distributed behavior control scheme based on the idea of applying artificial forces among agents to keep them in group formation. In a social potential field, we have an artificial force between each pair of agents which can be described as the sum of an attractive and repulsive component, both being inverse polynomial with the distance. The movement of the vehicle is determined by the sum of the forces acting on the vehicle. The formula we used for the force between two vehicles with the inter-vehicle distance r:

$$F(r) = \frac{-c_1}{r^{a_1}} + \frac{c_2}{r^{a_2}} \text{ where } c_1, c_2 \geq 0, a_1 > a_2 > 0 \quad (2)$$

where a_1, a_2, c_1 and c_2 are user-defined constants. We assume that the forces are active only between the vehicles which are part of the

Table 1: Example scenario configuration

Simulation parameter	Value		
Highway length	1 km		
Number of vehicles	5		
Communication range	50m		
Vehicle configuration	ID	Position	Speed(m/s)
	1	800	20
	2	600	28
	3	400	25
	4	200	32
	5	0	35

same convoy and in communication range of each other. Although more complex than the previous approaches, the influence strategy based on social potential fields has the following advantages:

- The convoy speed is dependent on the force parameters and can be adjusted using the parameters c_1, c_2, a_1 and a_2.
- The influence mechanism is able to regulate the inter-vehicle distance in the convoy.
- The influence mechanism does not suggest abrupt changes in the speed of the vehicles.

As the forces are dependent on the distance, the SPF influence strategy requires the knowledge of inter-vehicle distances.

3.3 An example

To illustrate the various phenomena at work, let us consider a small scenario which implements the SPF influence strategy and the proposed convoy joining policy. This example includes only 5 vehicles V1..V5 over a timespan of 10 seconds. While this simulation was performed with the same implementation used in Section 4, in this example we handcrafted the initial state of the scenario to illustrate as many interesting events as possible over a short timespan. The parameters of this scenario are listed in Table 1.

We recorded the evolution of the speed and position of each vehicle. To achieve a better visualization of the configuration of the convoy, our position graphs represent the *relative position* of the vehicles in relation to the last vehicle. The reason for this visualization approach is the fact that the relative movements of the vehicles are small compared with their common longitudinal movement, which would tend to dominate the absolute position plot.

Figure 1 shows the results of the scenario run, using the *SPF* influence algorithm. The top graph represents relative movement (with the origin of the relative coordinate system attached to V5), while the bottom graph represents the speed of the vehicles. The time scales are aligned to facilitate the observation of the correlation between speed changes and vehicle position. The thin lines represent the position of the vehicles if they choose not to join any convoy. This requires the vehicles to overtake the preceding vehicle even if the desired speed difference is small. In our case we have several overtakings (shown by intersecting thin lines). The bold lines represent the evolution of the vehicle locations with the assumption that the convoy formation mechanism is working.

Let us outline the series of events in the scenario:

- The vehicles start at time 0 with 200m distance between them. No convoy is formed as none of the vehicles are in each other's proximity. In absence of any convoy formation mechanism the speed of the vehicles is constant (shown by horizontal lines).
- V1, having a higher speed, approaches V2 from behind. Once they reach into each other's proximity, they agree to form a convoy and they adjust their speed through the SPF mechanism. This is a gradual process through which the speed of V1 is decreasing, while the speed of V2 is increasing. The speed of the vehicles settles at the agreed convoy speed at the moment when they achieve their desired following distance.
- Similarly, V3 approaches V4 from behind. They agree to form a convoy {V3, V4}.
- The convoy {V3, V4} is approached from behind by vehicle V5, which joins the convoy. This requires an increase of speed for V3 and V4 and a decrease of speed for V5.
- Finally, the convoy {V1, V2} is approached from behind by the convoy {V3, V4, V5}. The vehicles agree to merge the convoys, which requires V1 and V2 to increase the speed and V3, V4 and V5 to decrease.

The result is a convoy of 5 vehicles with a uniform speed and uniform inter-vehicle distance.

4. SIMULATION STUDY

In the following, we describe the results of a series of experiments in which we simulated the behavior of vehicles on a stretch of road using various convoy formation approaches. The simulation was implemented in the Java based Yet Another Extensible Simulator (YAES) [2] framework.

We have maintained the same convoy joining decision mechanisms across our experiments, but varied the influence mechanisms. The reason for this choice is that the decision of joining a convoy is, and will likely remain for a long time a decision of the human driver. However, once the convoy joining decision is made, the small adjustments will likely be delegated to the cruise control mechanism (traditional, intelligent or cooperative). At the first glance, it appears that identical convoy joining strategies would create identical sets of convoys (potentially with different speeds). It turns out, however, that this is not true over time. Starting from independent vehicles, indeed, the first set of convoys are formed in an identical way. Later, however, when, for instance vehicle V4 makes the decision whether to join the convoy {V1, V2, V3} the decision is based on the speed of the convoy, which is function of the influence mechanism. As a result, the set of convoys will diverge over time, and the different influence mechanisms may create a completely different macroscopic picture of the highway. As we had seen, current vehicular traffic essentially uses a form of ASL influence (even if the vehicles are equipped with intelligent cruise controls).

The question we are trying to answer is whether any of the more complex, collaboration based mechanisms (such as AVG or SPF) can achieve a better performance. Note that we are not interested in the behavior of the individual vehicles, but in the overall picture of the traffic: is it safer, more fluent, faster?

The parameters of the simulation are listed in Table 2. For these experiments, we considered a 60km long stretch of the highway with the number of vehicles ranging from 100 to 900 modeling various vehicle densities and traffic conditions. The data was collected by observing the traffic conditions for 600 seconds. The simulations were repeated 100 times with random initial conditions and the average values and the 95% confidence intervals were calculated.

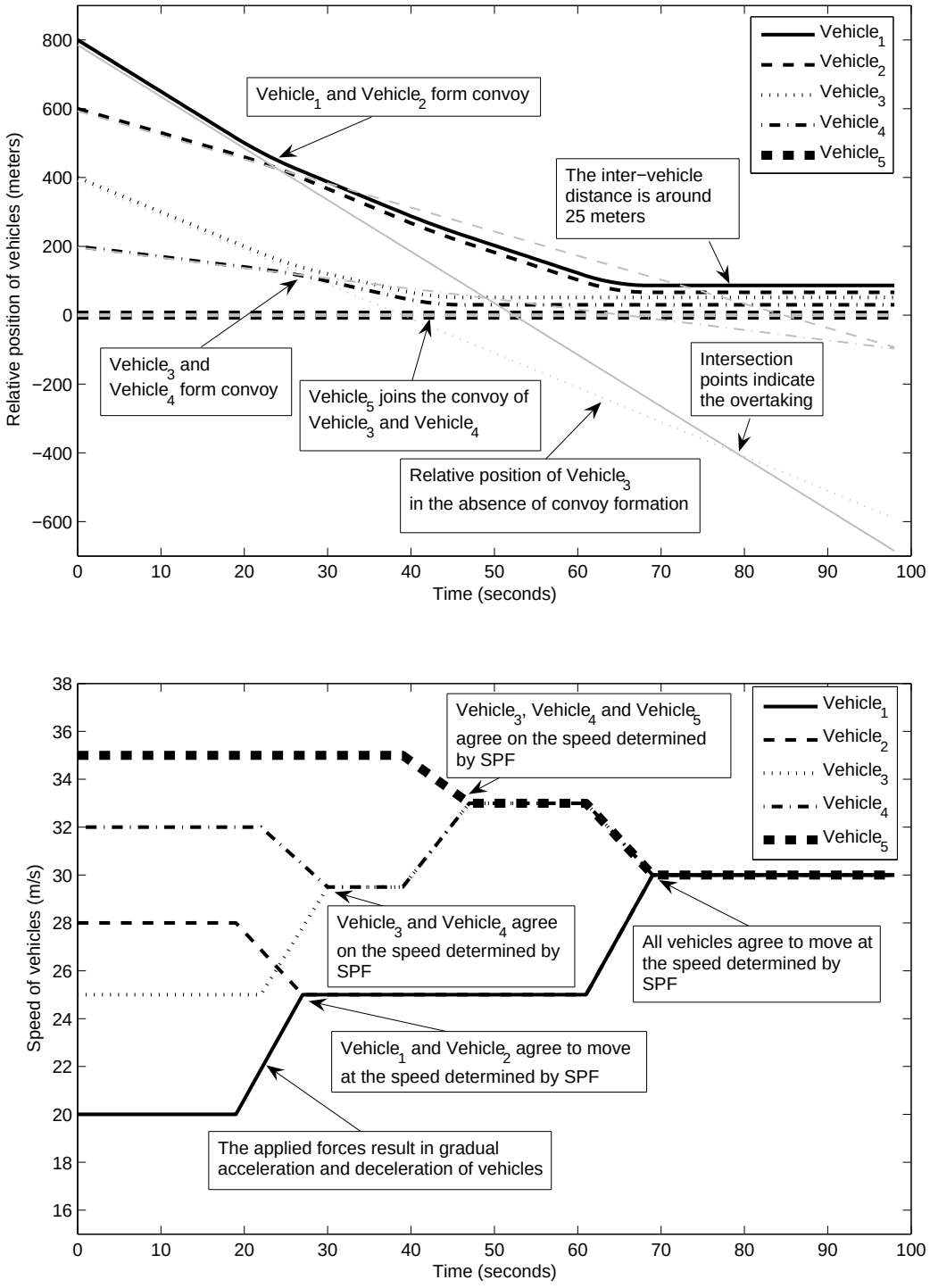


Figure 1: Convoy formation with the *SPF* influence mechanism. Top: relative position of the vehicles with respect to Vehicle-5. Thin lines: without convoy formation, thick lines: with convoy formation. Bottom: the evolution of the speed of the vehicles during convoy formation.

Table 2: Highway configuration used for the simulation

Simulation parameter	Value
Highway length	60 kilometer
Number of vehicles	100-900 vehicles
Vehicle initial speed	Uniformly distributed between 10m/s to 40m/s
Vehicle desired speed	Uniformly distributed between the vehicle's initial speed to 40m/s
Vehicle communication range	50m

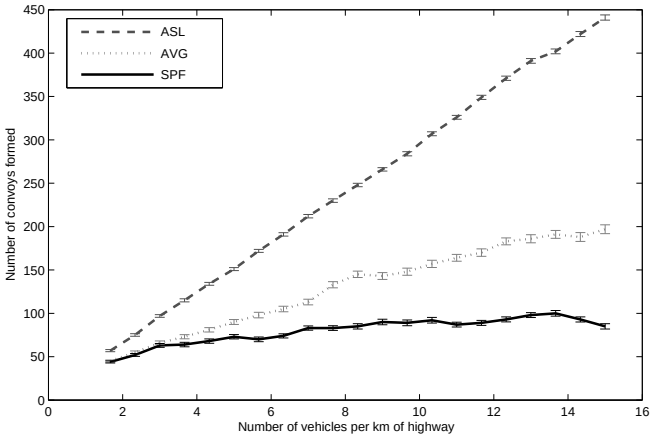


Figure 2: The number of convoys as a function of the density of the vehicles on the highway. The number of convoys formed increase with the density of the vehicles. The *ASL* and *AVG* approaches result in large number of convoys. The *SPF* approach results in the lowest number of convoys.

4.1 Number of convoys formed

Figure 2 shows the number of convoys function of the density of the vehicles. This number includes single-vehicle convoys, thus it is practically the number of independently operating units on the highway. Obviously, for a given traffic situation, the lower this value, the better, as it leads to a more fluent traffic. However, these results can not be interpreted in isolation, as it is also important to consider how well the speed of the convoy reflects the desires of the vehicles. For instance, a large convoy moving at the speed limit is desired, whereas a convoy formed of vehicles stuck behind a slow moving vehicle is not.

We find that the number of convoys vary greatly among the various influence strategies. With the *ASL* and *AVG* approaches, the number of convoys increases approximately linearly with the density, with the *AVG* approach creating about half as many convoys as the *ASL* approach.

The *SPF* approach, however, maintains a roughly constant number of convoys, in fact the number of convoys even show a slight, but noticeable decrease at high densities.

4.2 Distribution of convoy sizes

Figure 3 shows the distribution of the convoy sizes after the elapse of 600 seconds of simulation using 900 vehicles. As the size of the convoys ranges from 1 to 110, for sizes above 10 vehicles we have clustered them in groups of sizes 11-30, 31-50, 51-70, 71-90 and 91-110.

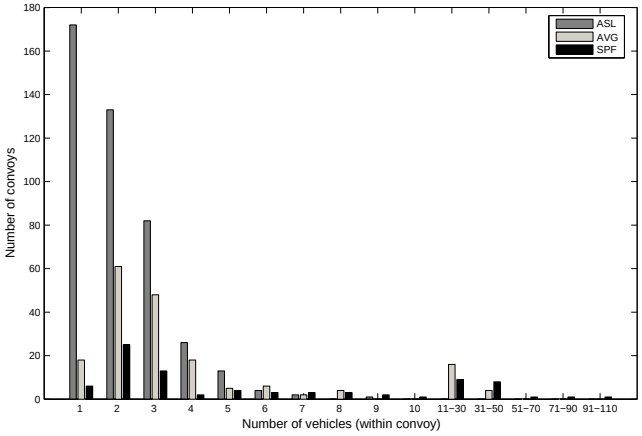


Figure 3: Distribution of the convoy sizes using a simulation involving 900 vehicles.

For the *ASL* influence mechanism, the most frequent case is convoys of size 1 (i.e. vehicles which are not part of any convoy), followed by convoys of 2 and 3 vehicles. From here, the number of convoys continues to drop very quickly. There were no convoys of 10 vehicles or more formed with the *ASL* influence mechanism.

For the *AVG* approach, the largest number of convoys had the size of 2 vehicles, followed by 3, 4 and finally 1 vehicle. The *AVG* approach allowed the occasional formation of larger convoys as well, up to the 31-50 vehicle range.

Finally, the *SPF* approach also shows the largest number of convoys consisting of 2 and 3 vehicles. However, the *SPF* influence mechanism allowed the creation of several very large convoys, up to the 90-110 vehicle range (naturally, as there are only 900 vehicles in the experiment, there can not be a very large number of convoys of this size).

The conclusion of this experiment is that every influence mechanism produces a different distribution of the convoy sizes. The *ASL* mechanism favors small convoys or even individual vehicles (note that our experiments did not model "convoys by necessity" where vehicles get stuck behind a slow moving vehicle). The *AVG* and *SPF* mechanisms prefer larger convoys with 2-4 vehicles, with an occasional larger convoy of up to 50 vehicles for *AVG* and up to 110 vehicles for the *SPF*.

4.3 Distribution of convoy speed

Figure 4 shows the distribution of the convoy speed at the end of 600 seconds of simulation using 900 vehicles. For better visualization, we have clustered the convoys in the speed ranges of 0-4, 5-9, 10-14, 15-19, 20-24, 25-29, 30-34, 35-39 and 40-44 (m/s).

We did not consider aspects of the highway traffic such as speed limits, or the risks of high speed driving, aspects which should not normally be regulated through the convoy mechanism. Under these assumptions, the higher the average speed of the vehicles, the better for the traffic.

With the *ASL* influence mechanism, the highest percentage of convoys are moving at the very slow speed of 10-14 m/s, followed by a smaller and smaller percentage of convoys moving at higher speed. A very minute percentage of convoys move at speed 30 m/s or higher.

With the *AVG* influence mechanism, the highest percentage of

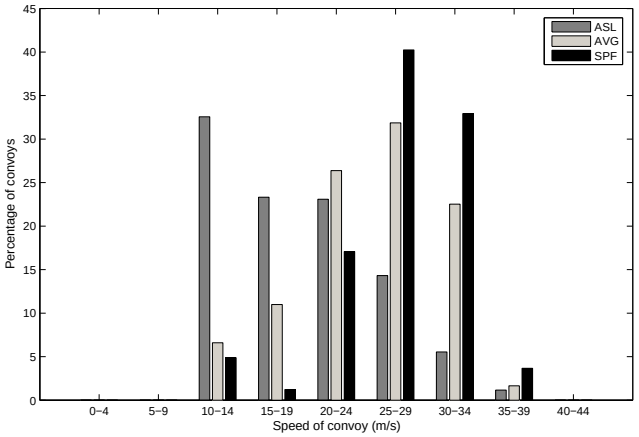


Figure 4: Distribution of the convoy speed using a simulation involving 900 vehicles.

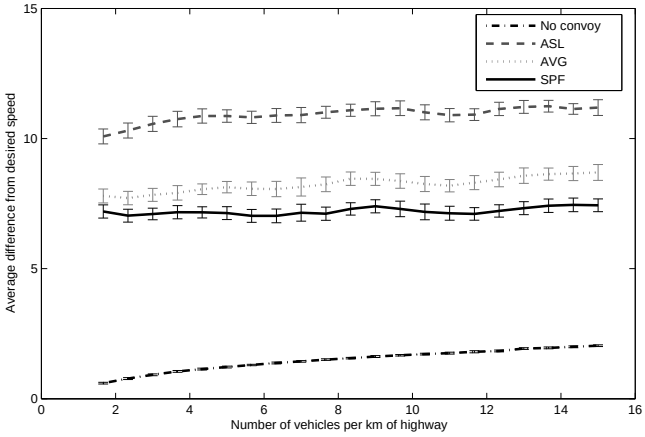


Figure 5: The average difference of the desired speed of the vehicles from their measured speed.

convoys are moving at the speed of 25-29 m/s, followed by the convoys moving at speed 20-24 m/s and then 30-34 m/s. A very small percentage of convoys move at the slow speed of 10-14 m/s or the highest speed of 35-39 m/s.

And finally, with *SPF* influence mechanism, the highest percentage of convoys are moving at the speed of 25-29 m/s, followed by a considerably large percentage of the convoys moving at the speed of 30-34 m/s.

So, the *ASL* mechanism favors convoys moving at very slow speed while both *AVG* and *SPF* approach favor faster moving convoys. The percentage of convoys moving at the speed of 25m/s or higher is largest with the *SPF* mechanism.

4.4 Average difference between measured and desired speed of the vehicles

From the point of view of the individual vehicle, the ideal driving environment is one in which the vehicle is alone on the road. In this situation, a vehicle would simply drive at its desired speed. When sharing the road with other vehicles, the agent might either form convoys, or attempt to achieve its desired speed by overtaking all the slower vehicles, irrespectively of the speed difference. On most roads, the act of overtaking in itself involves a certain amount of delay. Furthermore, a traffic environment where every vehicle is attempting to overtake all slower vehicles becomes highly chaotic and unsafe. On the other hand, convoy driving requires the vehicle to adjust its speed to the convoy, thus renouncing to its desired speed in exchange for the safety and predictability of convoy driving. In general, the lower the difference between the desired speed of the vehicle and the actual speed of the convoy, the better the convoy formation model is.

Figure 5 shows the average difference between the vehicles' measured and desired speed. The data used to plot this graphs was obtained by observing the middle 60 vehicles from a group of 900 vehicles moving on the highway. This was done to avoid the perturbations which occur at the periphery of the simulation environment. For instance, a fast vehicle at the front of the simulation would not have any slow vehicles in front of it, a fact which is an artifact of the simulation setup and it would reduce the accuracy of the measurement. The no-convoy graph was obtained under the assumption that all the vehicles in the traffic are trying to maintain their desired speed by overtaking all slower vehicles. We assumed that every overtaking incurs a small delay. Thus, with a large number of vehicles on the road, even the "no-convoy" approach does not guarantee the vehicle to move with its exact desired speed.

The graph shows that the smallest compromise is obtained by the "no-convoy" approach, followed in order by *SPF*, *AVG* and *ASL*. We can see that the collaborative convoy driving approaches *SPF* and *AVG* require a significantly lower amount of compromise between the actual and desired speed. This is a direct consequence of the fact that for these approaches all the vehicles in the convoy contribute to the choice of speed. This is a significant result because it provides strong motivation for the development of the collaborative convoy driving devices, and it makes it likely that the drivers will actually use them, as they can achieve speeds much closer to their desired speed compared to other approaches.

Convoy formation requires the vehicles to compromise over their desired speed. The *ASL* approach results in a large difference from the desired speed, because the vehicles will agree on the slow speed of the front vehicle. The *AVG* approach is somewhat better, while the *SPF* approach shows the smallest difference. This means that the *SPF* approach allows the vehicles to drive the closest to their desired speed. This is because *SPF* based convoys tend to agree on higher than average speed and the vehicles generally have the desire to move at higher speed. Also the utility function guarantees that vehicles do not join convoys that have large difference from their desired speed.

4.5 Number of overtakings

Figure 6 shows the number of overtakings as a function of the density of the vehicles. In general, the smaller the number of overtakings, the safer the traffic. The data used to plot the graphs was also obtained by observing 60 vehicles in the middle of the highway.

As expected, in the absence of any convoy formation approach, there are large number of overtakings. This number increases with the density of the vehicles. As expected, the number of overtakings are reduced by using the convoy formation approaches. The number of overtakings are the smallest for the *SPF* approach, followed by *AVG* and *ASL*. This can be attributed to the larger convoy sizes resulting from the *SPF* approach. While the number of overtakings increases with vehicle density for all three approaches, the increase is the slowest for SPF, making it the most scalable approach.

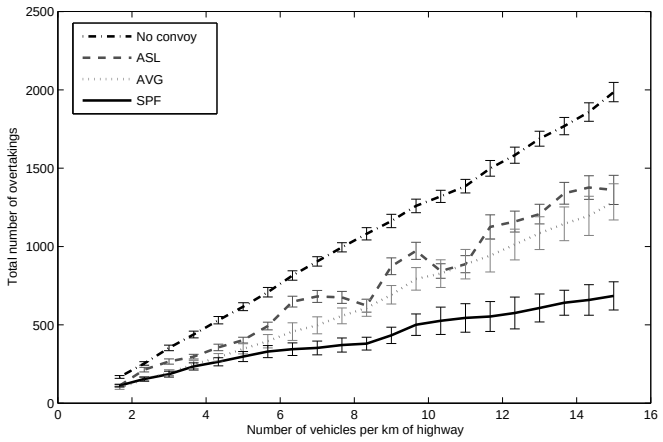


Figure 6: The number of overtakings as a function of the density of the vehicles.

5. CONCLUSIONS

It is a well-known fact that convoy driving has a beneficial effect on the fluency of the traffic, improving safety, and (in average) reducing traveling time. Naturally, convoy driving can be accomplished without the mediation of communicating agents, by adapting to the speed of the previous vehicle (which is the equivalent of the *ASL* strategy). This is the approach taken both by human drivers, as well as intelligent cruise control systems. For more complex strategies, however, it is necessary for vehicles to exchange information with each other. The *AVG* and *SPF* influence strategies we proposed can not be accomplished without inter-vehicle communication.

Our experimental results show that these collaborative strategies have significant benefits by allowing the formation of larger convoys, bringing the average speed of the convoys closer to the desired speed of the participating agents and reducing the number of overtakings.

As we had seen, the technical means for a widespread deployment of the collaborative cruise control systems will become available in the near future. These will likely by dual-control systems, where the decision to join a convoy will be under the control of the human driver, while the speed adjustments necessary to maintain the convoy and the communication necessary to determine the overall convoy speed will be under the control of the agent. As we had seen, even very simple convoy influence mechanisms can yield significant improvements in overall traffic behavior. More complex influence mechanisms, based on advanced negotiation models, global traffic awareness and so on will bring a new set of research challenges.

Acknowledgments

This research was sponsored in part by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-06-2-0041. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation heron.

6. REFERENCES

[1] N. Bellomo and M. Delitala. On the mathematical theory of vehicular traffic flow I: Fluid dynamic and kinetic modelling. *Mathematical Models and Methods in Applied Sciences*, 12(2):1801–1843, 2002.

[2] L. Bölöni and D. Turgut. YAES - a modular simulator for mobile networks. In *Proceedings of the 8-th ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems MSWIM 2005*, pages 169–173, October 2005.

[3] D. Chowdhury, L. Santen, and A. Schadschneider. Statistical physics of vehicular traffic and some related systems. *Physics Reports*, 329(4-6):199–329, 2000.

[4] K. Dresner and P. Stone. Multiagent traffic management: A reservation-based intersection control mechanism. In *In The Third International Joint Conference on Autonomous Agents and Multiagent Systems*, page 530âĂŞ537, July 2004.

[5] A. Girard, J. Sousa, J. Misener, and J. Hedrick. A control architecture for integrated cooperative cruise control and collision warning systems. In *Proceedings of the IEEE Conference on Decision and Control*, 2001.

[6] M. Khan and L. Bölöni. Convoy driving through ad-hoc coalition formation. In *Proceedings of IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), San Francisco, California*, pages 98–105, Los Alamitos, CA 90720-1314, March 2005. IEEE Computer Society Technical Committee on Real-Time Systems, IEEE Computer Society Press.

[7] J. Laumonier, C. Desjardins, and B. Chaib-draa. Cooperative adaptive cruise control: a reinforcement learning approach. In *4th Workshop on Agents in Traffic And Transportation, AAMAS'06*, Hakodate, Hokkaido, Japan, 2006.

[8] S. Paveri-Fontana. On boltzmann-like treatments for traffic flow: a critical review of the basic model and an alternative proposal for dilute traffic analysis. *Transportation research*, 9:225, 1975.

[9] J. Reif and H. Wang. Social potential fields: A distributed behavioral control for autonomous robots. In *Proceedings of International Workshop on Algorithmic Foundations of Robotics (WAFR)*, pages 431–459, 1995.

[10] K. N. und M. Schreckenberg. A cellular automaton model for freeway traffic. *J. Phys. I France*, (2):2221–2229, 1992.

Bottlenecks and Congestion in Evacuation Scenarios: A Microscopic Evacuation Simulation for Large-Scale Disasters

Gregor Lämmel
Transport Systems Planning and Transport Telematics (VSP)
TU Berlin, Salzufer 17-19, Sekr. SG 12, 10587 Berlin, Germany
laemmel@vsp.tu-berlin.de

Marcel Rieser
Transport Systems Planning and Transport Telematics (VSP)
TU Berlin, Salzufer 17-19, Sekr. SG 12, 10587 Berlin, Germany
and
Institute for Transport Planning and Systems (IVT)
ETH Zurich, 8093 Zurich, Switzerland
rieser@vsp.tu-berlin.de

Kai Nagel
Transport Systems Planning and Transport Telematics (VSP)
TU Berlin, Salzufer 17-19, Sekr. SG 12, 10587 Berlin, Germany
nagel@vsp.tu-berlin.de

ABSTRACT

Multi Agent Simulation has increasingly been used for transportation simulation in recent years. With current techniques, it is possible to simulate systems consisting of several million agents. Such Multi Agent Simulations have been applied to transportation simulation for whole cities and even large regions. In this paper we demonstrate how to adapt an existing multi agent transportation simulation framework to large-scale pedestrian evacuation simulation. The underlying flow model simulates the traffic based on a simple queue model where only free speed and bottleneck capacities are taken into account. The queue simulation, albeit simple, captures the most important aspects of evacuations such as the congestion effects of bottlenecks and the time needed to evacuate the endangered area.

During the simulation, each evacuee optimizes his/her personal evacuation route to find the fastest escape route. At this point two different routing solutions are considered: (1) An "empty network" routing solution, where every evacuee follows the path that would be fastest in an empty network. (2) A "Nash equilibrium" approach, where, via iterations every evacuating person attempts to find a route that is optimal for him/herself under the given circumstances. Both approaches can be considered as benchmarks: the first as one where congestion effects are not taken into account in the path choice; the second one as one which might be achieved by appropriate training or guidance while maintaining acceptability in the sense that no person could gain by deviating from this solution. The results from the simulation give an estimate of the time it could take to evacuate the endangered area. We applied the system to a hypothetical scenario, namely a dam-break of the Sihlsee dam near Zurich, which would lead to an inundation of large parts of the city of Zurich within two hours. We show how well both approaches perform with respect to evacuation time and the outflow rate of evacuees.

Keywords

multi agent simulation, large-scale evacuation simulation

1. INTRODUCTION

The evacuation of whole cities or even regions is an important problem, as demonstrated by recent events such as the evacuation of Houston in the case of Hurricane Rita or the evacuation of coastal cities in the case of Tsunamis. A robust and flexible simulation framework for such large-scale disasters helps to predict the evacuation process. Furthermore, it is possible to recognize bottlenecks in advance, so that an elimination of those bottlenecks is possible. This should lead to a better preparedness for an event of evacuation for cities or regions that face a high risk of natural disasters.

2. RELATED WORK

Disaster and evacuation planning has become an important topic in science and politics. In principle there are two different situations: evacuation of buildings, ships and airplanes or the like on the one hand, or evacuation of whole cities or even regions on the other hand. The former involves normally the evacuation of pedestrians, where the latter is rather associated with the evacuation by car.

In the area of pedestrian evacuation simulation, there has been done considerable research in the last 20 years. A good overview about models and software for pedestrian evacuation simulation can be found in the proceedings of the conference "Pedestrian and Evacuation Dynamics" [35, 9, 10]. Corresponding to the two different types of problems, there are two different basic approaches for simulating the traffic flow:

(1) Methods of dynamic traffic assignment (DTA) have been applied to evacuation simulation on the city or regional scale. Some examples are: MITSIM [19], DYNASMART [22] or VISSIM [14]. The DTA approach is based on the analogy between traffic and hydrodynamic characteristics of fluids. That means DTA is a macroscopic approach and reduces the problem of evacuation dynamics to a well known physical problem. On state of the art hardware it is possible to handle even large-scale scenarios with this approach. – However, in DTA it is not straightforward to deal with the inhomogeneity of a population. For this, a microscopic simulation is needed, where all people are simulated as individuals.

(2) Microscopic simulations are often based on Cellular Automata (CA) [28, 29, 16]. In CA models each evacuee is designed as an individual; therefore it is possible to simulate also population structures where people have different speeds or ranges, or more complex behavior. The modeling of complex behavior in evacuation simulation has become important in recent years. People could for example ignore warnings or might not choose the nearest emergency exit, furthermore people tend to follow others (herd behavior) [15, 23]. Agent oriented research groups have modeled such behavior [27, 30]. In general it is expected that complex behavior leads to longer evacuation times, consequently a simulation that ignores such behavior patterns is probably optimistic.

The aim of this approach is to develop a simulation framework for large-scale scenarios, e.g. for large cities with a population of hundreds of thousands. A standard CA-based approach is not applicable here, because the area of those cities could be several hundred square kilometers. In this case a CA-model would consist of more than 10^9 cells, leading to rather long computing times.

In contrast, a DTA approach, as pointed out earlier, is not able to handle complex individual behavior. One possible approach to deal with such large-scale scenarios but to retain persons as individual agents is based up on a modified queuing model [11, 36]. The queuing model simplifies streets to edges and crossings to nodes; the difference to standard queuing theory is that agents (particles) are not dropped but spill back, causing congestion. This graph-oriented model is defined by lengths/widths, free speed and flow capacity of the edges. This simplification leads to a major speedup of the simulation while keeping results realistic. For example, the simulation of the whole (motor) traffic of Switzerland (approx. 5 million trips) takes less then 5 minutes for 24h real time [32]. In this work the adaptation of the existing multi agent transportation simulation framework to large-scale pedestrian evacuation simulation is described.

3. MULTI AGENT SIMULATION

Our simulation is constructed around the notion of agents that make independent decisions about their actions. In this case study, each evacuee is modeled as an individual agent in our simulation. In the simulation the agents try to find the best (in terms of time) escape route, whereby the real world is modeled as a network constructed of nodes (intersections) and links (roadway between intersections). The overall approach consists of three important pieces:

- Each agent independently generates a so-called *plan* which encodes its intended escape route.
- All agents' plans are simultaneously executed in the simulation of the physical system. This is also called the *traffic flow simulation* or *mobility simulation*.
- There is a mechanism that allows agents to *learn*. In our implementation, the system iterates between plans generation and traffic flow simulation (i.e. systematic relaxation [20, 4]). The system remembers several plans per agent, and scores the performance of each plan. Agents normally chose the plan with the highest score, sometimes re-evaluate plans with bad scores, and sometimes obtain new plans. Further details will be given below.

The simulation approach is the same as in many of our previous papers (e.g. [33, 3]) on the same subject. The results of this paper are based on a re-implementation of the MATSim framework in Java [25]. Since not all elements of MATSim are important for an evacuation simulation, the following exposition is a shortened and simplified description of key elements.

A **plan** contains the itinerary of activities the agent wants to perform during the day, plus the intervening trips the agent must take to travel between activities. An agent's plan details the order, type, location, duration and other time constraints of each activity, and the mode, route and expected departure and travel times of each trip. This paper concentrates on "home" and "evacuated" as the only activities, and "walk" as the only mode.

A plan can be modified by the **router module**: The router is implemented as a time-dependent Dijkstra algorithm. It calculates link travel times from the output of the traffic flow simulation. The link travel times are encoded in variable-sized time bins, so they can be used as the time-dependent weights of the links in the network graph.

The **traffic flow simulation** executes all agents' plans simultaneously on the network. In the work presented here, the plans contain the departure time and the exact routes, and agents just follow these prescriptions; learning is implemented via iterations (see below). The traffic flow simulation is implemented as a queue simulation, where each street (link) is represented as a FIFO (first-in first-out) queue with three restrictions [11, 5]. First, each agent has to remain for a certain time on the link, corresponding to the free speed travel time. Second, a link flow capacity is defined which limits the outflow from the link. If, in any given time step, that capacity is used up, no more agents can leave the link. Finally, a link storage capacity is defined which limits the number of agents on the link. If it is filled up, no more agents can enter this link. The traffic flow simulation provides output describing what happened to each individual agent during the execution of its plan.

The outcome of the traffic flow simulation (e.g. congestion) depends on the planning decisions made by the decision-making modules (in this case, the router). However, those modules can base their decisions on the output of the traffic flow simulation (e.g. knowledge of congestion) using **feedback** from the multi-agent simulation structure [20, 4]. This sets up an iteration cycle which runs the traffic flow simulation with specific plans for the agents, then uses the planning modules to update the plans, these changed plans are again fed into the traffic flow simulation, etc., until consistency between modules is reached.

The feedback cycle is controlled by the **agent database**,

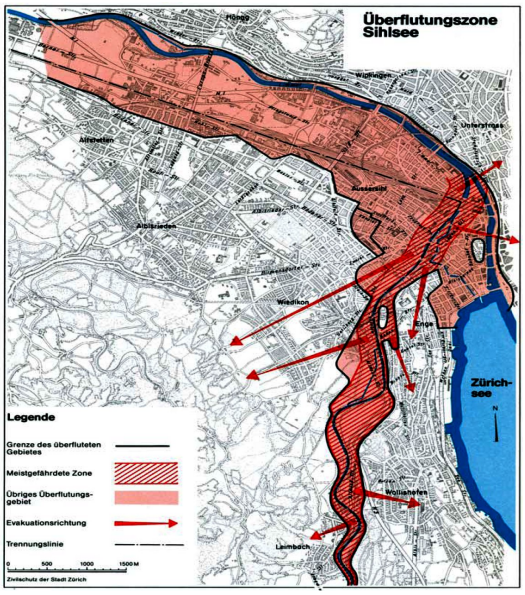


Figure 1: Inundation map provided by the Zurich civil defense office

Figure 2: Empty evacuation network

which also keeps track of multiple plans generated by each agent, allowing agents to reuse those plans at will. The repetition of the iteration cycle coupled with the agent database enables the agents to learn how to improve their plans over many iterations. This circle continues until the system has reached a relaxed state. At this point, there is no quantitative measure of when the system is "relaxed"; we just allow the cycle to continue until the outcome seems stable. The actual number of iterations that are needed depends on the scenario. Normally 100 to 200 iterations are sufficient to reach this "relaxed" state (also see below).

In order to compare plans, it is necessary to assign a quantitative **score** to the performance of each plan. In principle, arbitrary scoring schemes can be used (e.g. prospect theory [2]). In this work it is assumed that the agents are only interested in minimizing their individual evacuation time. For that reason, the utility of a plan is just the negative of the time needed to reach the safe area.

4. SCENARIO

A hypothetical event of a dam-break of the Sihlsee dam was chosen. This would lead to an inundation of parts of the city of Zurich. According to the civil defense office there will be an advance warning time of about 110 minutes until the inundation will reach the city center. The civil defense office also provides an instruction sheet [1] with an inundation map of the area at risk (shown in figure 1).

4.1 Data Basis

There are two main inputs that have to be provided to the simulation framework. At first the simulation needs a network. We extracted the evacuation network by projecting the inundation map from the civil defense office onto the network of Switzerland provided by NAVTEQ[1]. This extraction has been done semi-automatically. First, all boundary nodes were selected manually, and after this all links and nodes inside the so selected area were selected automatically. So the overall effort of pre-processing was manageable. The original NAVTEQ network of Switzerland consists of about 400k nodes and 880k links. After cropping, the resulting network consist of 3037 nodes and 6120 links. It is shown in figure 2.

The other important input to the simulation framework is a so-called "plans file", containing information about people and their plans, including home and work locations. A synthetic population for the area of Zurich was generated by [26] and provided to us. The population was generated using data from the Swiss census for the year 2000 [12] and information about facilities in the city center. Every person in this synthetic population obtains one complete day plan, describing all activities the person performs during a day. The first work location appearing in a plan of each agent was extracted to build the agents' initial locations for the evacuation. In the end, there were 165571 agents with a work activity within the endangered area. This set of agents and locations builds our start setup for the evacuation; this means we will simulate a break of the Sihlsee dam during regular working hours.

4.2 Calibration of the Queuing Model

Since the underlying simulation framework is mainly designed for the simulation of motorized transportation, several adaptations are necessary. At first it is obvious that the evacuees do not care about the traffic direction. So we allowed all links in the street network to be used in both directions. Given the link length, the queuing model is described by three parameters for each link. The parameters are: **flow capacity**, **storage capacity** and **free flow speed**. These parameters had to be calibrated to achieve an appropriate flow dynamic for pedestrians. In literature the flow dynamic of pedestrians is often described by fundamental diagrams [37, 31]. These diagrams show the velocity as a function of

[1]NAVTEQ is a provider of digital maps for in-vehicle navigation systems (see also http://www.navteq.com/)

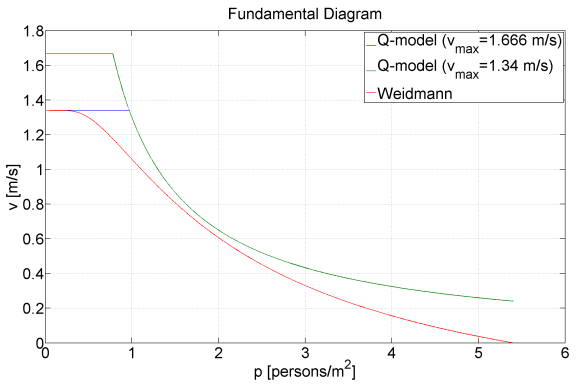


Figure 3: Weidmann's fundamental diagram compared to queuing model

the density of pedestrians. Weidmann pointed out that the relation between density and velocity is adequately captured by the so-called Kladek-formula [37][2]:

$$v_{F,hi}(D) = v_{F,hf} \times [1 - e^{-\gamma \times (\frac{1}{D} - \frac{1}{D_{max}})}]$$

With:

- $v_{F,hi}$ the velocity at a particular density $[m/s]$,
- $v_{F,hf}$ the velocity at free flow $[m/s]$,
- γ a free parameter $[persons/m^2]$,
- D the actual density $[persons/m^2]$ and
- D_{max} the density at which no flow occurs $[persons/m^2]$.

Empirical studies showed the best results with $\gamma = 1.913$ $persons/m^2$, $v_{F,hf} = 1.34$ m/s and $D_{max} = 5.4$ $person/m^2$.

Our queuing model, however, generates a speed-density relationship of the form $v = \min[v_{max}, 1/D]$ [36]. Therefore a complete agreement is not possible. However, as shown in figure 3, the flow dynamic produced by our queue model is not too far away from Weidmann's fundamental diagram. The details of the calibration are explained in the following paragraphs.

As the above mentioned NAVTEQ network is designed for transport simulation, we had to adjust the networks parameters accordingly. In the original network file, there is only information about number of lanes but not the width of the street, so we had to estimate it. According to the handbook Strassenprojektierung [6] the lane width on streets in Switzerland has to be 2.20-3.00 m for automobiles and 3.10-3.90 m for trucks. Taking this information, we set the width of all lanes in the network to 3.50 m. For pedestrian evacuation the flow capacity is assigned in persons per meter per second, but it depends on the actual density of persons. According to Weidmann [37] the maximum flow is about 1.3 $persons/(m \cdot s)$ at a density of 2 $persons/m^2$. The SFPE Handbook of Fire Protection Engineering [8] supports these values. Together with the lane width we got the flow capacity of 4.55 $persons/(lane \cdot s)$.

[2]Newer studies [34] imply other fundamental diagrams then those from Weidmann or Predtetschenski and Milinski. An adaptation of these values could, in consequence, become necessary in future.

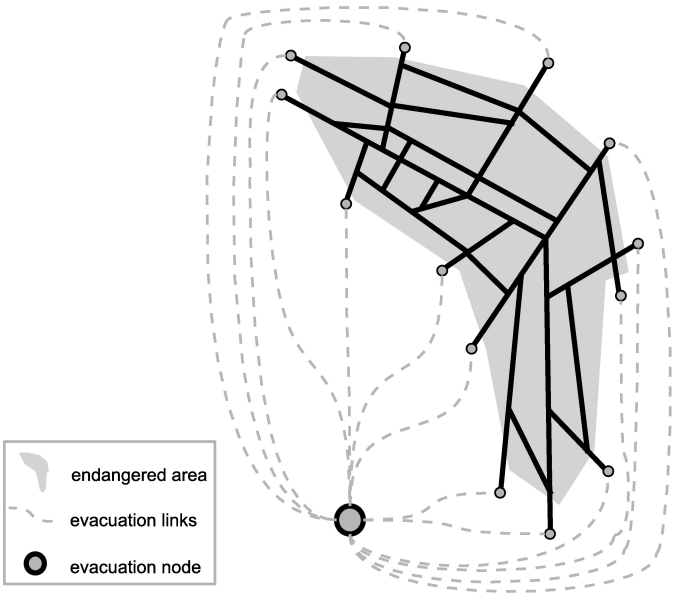


Figure 4: Sketch of the modified evacuation network

Another parameter for the queue simulation is the storage capacity of the links. Not to contradict the flow rate in Weidmann's fundamental diagram we set the storage capacity to 2 $persons/m^2$.

The free flow speed was set to $1.666m/s$. This value is slightly higher then the $1.34m/s$ recommended in literature, but the values presented by Weidmann reflect the pedestrian flow under normal conditions and not in a case of emergency. Before we can apply this approach to "real world" scenarios we have to verify all parameters and check if they are realistic.

Overall, there are 101 links that lead out of the evacuation area. Most of them have one or two lanes. The aggregated capacity of all these "escape links" is 787.15 $persons/s$. However, this capacity is a theoretical value since it is unlikely that the evacuees will find a way to distribute themselves in such a smooth way over the network. Rather it is expected that this outflow rate has a much lower value at the initial iteration, where all evacuees proceed on the assumption that the network is empty and there is free speed on all links. With the optimization of the evacuation procedure the outflow rate is expected to increase as the evacuees will make better use of all roads.

4.3 Initial Routing

Initial plans use the shortest path (according to free speed travel time) out of the evacuation area for all agents. Within the MATSim framework a shortest path router based on Dijkstra's shortest path algorithm [7] has been implemented. This router finds the shortest path in a weighted graph from one node to any other, whereby the actual weights for a link are defined by a time-dependent cost function. Since we want to evacuate the city as fast as possible, the weights represents the (expected) travel time[3].

There is, however, no particular node as the target of the

[3]For the initial evacuation plans the expected travel time is determined by free travel speed.

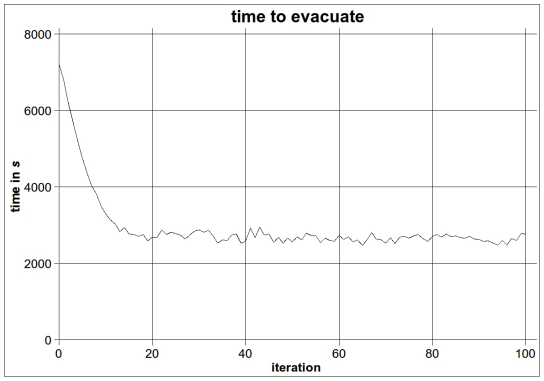


Figure 5: Evacuation time vs. iteration number

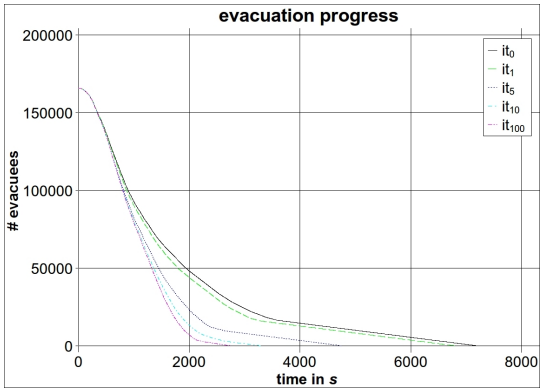


Figure 6: Evacuation progress

shortest path calculation, as the evacuees have more than one safe place to run to. Instead, in the underlying domain every node outside the evacuation area is a possible destination for an agent that is looking for an escape route. To resolve this, the standard approach (e.g. [24]) is to extend the network in the following way: All links which lead out of the evacuation area are connected, using virtual links with infinite flow capacity and zero length, to a special "evacuation node" (see figure 4). Doing so, Dijkstra's algorithm will always find the shortest route from any node inside the evacuation area to this evacuation node.

4.4 Re-Planning and Learning

At the end of each iteration, every agent scores the performed plan. In this study the scoring function is simply the negative of the travel time. This score is then memorized for the plan. After an agent has updated the score of its actual plan, it will be selected with a probability of 10% for re-routing. This replanning probability is a configurable parameter; 10% is a good compromise between slow convergence on the one hand, and over-reaction of the system on the other hand. In the re-routing procedure, the Dijkstra router is again applied to find the fastest escape route for the particular agent. The difference to the initial routing is that the weights for the links are no longer based on free speed travel times but on the experienced travel times from the last iteration. The travel times of all links are recorded and averaged into time bins. More precisely, the link traversal times of all pedestrians *entering* a link during a specific time bin are averaged. Those link travel times are then used when, during the Dijkstra computation, a specific link is entered by the algorithm. More details can be found in [18].

The size of these bins is configurable; for the present study, a size of 15 mins was used. If no traffic for a particular time bin and link occurs, free speed travel time is assumed for this time bin and link.

For agents that have not been chosen for re-planning, the plans with the highest scores (i.e. the plan with the fastest escape route) are selected for the next iteration. Repeating this iteration cycle, the agent behavior will move towards a Nash equilibrium. If the system were deterministic, then a state where every agent uses a plan that is a best response to the last iteration would be a fixed point of the iterative dynamics, and at the same time a Nash Equilibrium since no agent would have an incentive to unilaterally deviate. Since, however, the system is stochastic, this statement does not hold, and instead we look heuristically at projections of the system such as in Fig.5. In all such plots, 100 iterations is more than enough to arrive at a horizontal line, indicating that the iterative dynamics has reached a steady state.

In most (but not all) evacuation situations, the Nash equilibrium leads to a shorter overall evacuation time than when everybody moves to the geographically nearest evacuation point. On the other hand, a Nash equilibrium means that nobody has an incentive to deviate. The Nash equilibrium in an evacuation situation can therefore be considered as a solution that could be reached by appropriate training.

5. RESULTS

The simulation run was performed on a dual core CPU at 2.33 GHz with 2 GB of RAM. The computer runs JAVA jdk1.5_012 on Linux. The evacuation simulation was stopped after 100 re-planning cycles. The average runtime for an iteration was 123 seconds and the overall runtime was 3 hours and 24 minutes. The simulation consumed up to 1393MB of RAM. Besides the evacuation time, the outflow rate of the evacuation area has been recorded, too.

As expected, the evacuation time decreases significantly with the iterations. Especially within the early iterations, it drops very fast. A diagram that represents this process is shown in figure 5. The evacuation takes 7205 seconds at the initial iteration. Beginning with iteration 15 there are only small changes and it fluctuates randomly around 2676 seconds.

These values show only how long the overall evacuation takes but it tells nothing about the evacuation process itself. Therefore we evaluated the evacuation process for iteration 0, 1, 5, 10 and 100 in detail. Figure 6 shows the results.

The initial iteration results in a steep gradient (high outflow) at the beginning but it flattens very fast. As the iterations progress the initial gradient gets even steeper and becomes more linear.

Some statistics of the outflow of evacuees for the discussed iterations are given in table 1. Overall the results are as expected: both the maximum flow and the median flow are increasing with the iterations. Nevertheless, there are some interesting details. One interesting aspect is the low value for the median of the initial or 5th iteration. A possible reason for this phenomenon is that many agents try to perform

Iteration	max	mean	median
0	127	22.98	5
1	129	24.32	5
5	139	34.78	10
10	139	50.26	46
100	148	59.94	68

Table 1: Statistics of the outflow rate ($persons/s$)

the same escape route. If this happens they will line up in a few long queues, which will result in a low constant outflow rate.

The comparison of the snapshots for iteration 0 and 100 in figure 7 supports this hypothesis. Both snapshots were taken after 30 minutes of evacuation. The escape directions are indicated by black arrows. In iteration 0 there are considerably more evacuees at this point then in iteration 100. In the latter, the agents take advantage of six evacuation points (indicated by the red circles). This is much more effective then the behavior in the initial iteration, where only four evacuation points are used. Bottlenecks can also be detected. Figure 8 depicts this issue. In this figure the links are colored dependent on congestion. A green color indicates that the agents travel with free flow speed and as the color moves to red the flow speed decreases. It is not surprising that these congestion instances emerge at bridges, but the snapshot is taken after 100 iterations of learning and that means: there seems to be no better solution for the individual agent then to queue up on these bridges.[4]

6. DISCUSSION

The simulations concentrate on two types of agent behaviors: One where every agent follows the shortest path to the safe area; one where a Nash equilibrium is reached. Both can be considered as benchmarks:

- The first as one where agents are rational about their path choice, but unaware of congestion effects.
- The second as a solution that could be reached by training, assuming that agents follow the training solution also in the real situation.

Clearly, both can only be considered as benchmark solutions. In panic situations, people tend to be irrational and to display herd behavior [15]. Still, if even the Nash equilibrium solution does not leave enough time, then this would be a strong indicator that major measures would need to be taken to rectify the situation.

It should also be stated that Nash equilibrium and system optimum do not need to coincide – i.e. that solutions even better than the Nash equilibrium might be possible. Such solutions would, however, be unstable in the sense that people would have an incentive to deviate. Such solutions seem even more improbable than Nash equilibrium solutions.

Finally, one should mention that MATSim already contains the first hooks towards en-route replanning [17]. This would allow to add situation-based behavior into the simulation.

[4]For those who know the area: Since this preliminary study is based on a vehicular traffic network, it ignores links which can be used by pedestrians only. This could be corrected by using different network data.

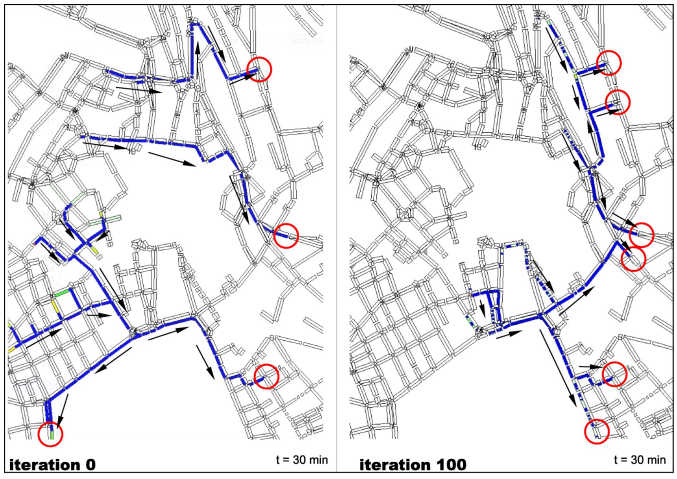


Figure 7: Comparison of two snapshots

Another issue concerns the mode choice: The investigation assumes that all evacuation is done by foot while it might be reasonable to assume that some people use cars or cycles, and they might even leave vehicles in the street to continue on foot if progress by vehicle becomes too slow. For the time being, such issues are not considered. The queue model could, to a certain extent, be parameterized to deal with mixed traffic, as long as all modes move with the same speed. Beyond that, one would arguably need to switch to a true two-dimensional model such as [15] or [21]. Such models could still operate on networks [13].

7. CONCLUSIONS

We introduced a microscopic pedestrian simulation framework for large-scale evacuations. It is implemented as a Multi Agent Simulation, where every agent tries to optimize its individual evacuation plan in an iterative way. The simulation framework is demonstrated through a case study based on a hypothetical dam-break of the Sihlsee dam near Zurich. Despite the underlying behavioral model being quite simple, the simulation gives plausible results regarding the predicted evacuation time and bottlenecks. The runtime performance shows that this approach is well suited for large scale scenarios. With state of the art hardware it is no problem to simulate much larger scenarios with over one million agents. In future work it is planed to apply this framework to an evacuation simulation in the case of a Tsunami warning for the Indonesian city of Padang. The improvement of the behavioral model (e.g. herd behavior [15] modified for large-scale scenarios [13]) could also be a topic of future work.

8. ACKNOWLEDGMENTS

This project was funded in part by the German Ministry for Education and Research (BMBF), under grant number 03G0666E ("last mile").

9. REFERENCES

[1] *Wasseralarm Sihlsee*. Zivilschutz Stadt Zürich (www.stadt-zuerich.ch). handout in German.

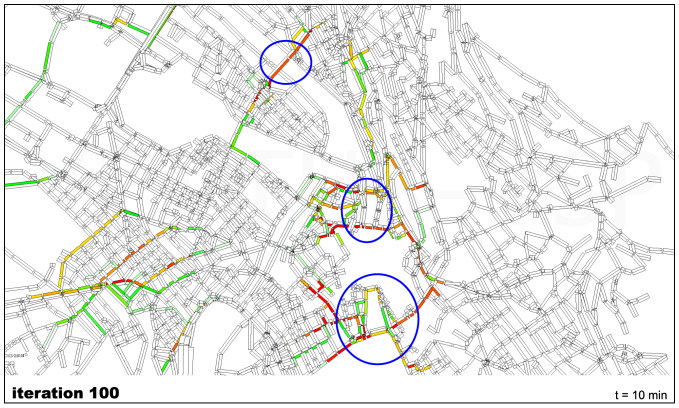


Figure 8: Bottlenecks after 100 iterations

[2] E. Avineri and J. Prashker. Sensitivity to uncertainty: Need for paradigm shift. Paper 03-3744, Transportation Research Board Annual Meeting, Washington, D.C., 2003.

[3] M. Balmer, B. Raney, and K. Nagel. Adjustment of activity timing and duration in an agent-based traffic flow simulation. In H. Timmermans, editor, *Progress in activity-based analysis*, pages 91–114. Elsevier, Oxford, UK, 2005.

[4] J. Bottom. *Consistent anticipatory route guidance*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 2000.

[5] N. Cetin, A. Burri, and K. Nagel. A large-scale agent-based traffic microsimulation based on queue model. In *Proceedings of Swiss Transport Research Conference (STRC)*, Monte Verita, CH, 2003. Earlier version, with inferior performance values: Transportation Research Board Annual Meeting 2003 paper number 03-4272.

[6] K. Dietrich. *Strassenprojektierung*. Schriftenreihe des IVT. Institute for Transport Planning and Systems ETH Zürich, 9 edition, 1998. in German.

[7] E. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269 – 271, 1959.

[8] P. DiNenno, editor. *SFPE Handbook of Fire Protection Engineering*. National Fire Protection Association, Boston, 2nd edition, 1995.

[9] E. R. Galea, editor. *Pedestrian and Evacuation Dynamics*. Proceedings of the 2nd international conference, London, 2003. CMS Press, University of Greenwich, UK, 2003.

[10] P. Gattermann, N. Waldau, and M. Schreckenberg, editors. *Pedestrian and Evacuation Dynamics*. Proceedings of the 3rd international conference, Vienna. Springer, Berlin, 2006.

[11] C. Gawron. An iterative algorithm to determine the dynamic user equilibrium in a traffic simulation model. In D. Wolf and M. Schreckenberg, editors, *Traffic and granular flow'97*, pages 469–474. Springer, Berlin, 1998.

[12] U. Germann. *Abschlussbericht zur Volkszählung 2000*, volume 1 of *Statistik der Schweiz*. Bundesamt für Statistik, Neuchâtel, 2005. in German.

[13] C. Gloor, P. Stucki, and K. Nagel. Hybrid techniques for pedestrian simulations. In *Cellular automata, Proceedings*, number 3305 in Lecture Notes in Computer Science, pages 581–590. Springer, 2004.

[14] L. Han and F. Yuan. Evacuation modeling and operations using dynamic traffic assignment and most desirable destination approaches. Paper 05-2401, Transportation Research Board Annual Meeting, Washington, D.C., 2005.

[15] D. Helbing, I. Farkas, and T. Vicsek. Simulating dynamical features of escape panic. *Nature*, 407:487–490, 2000.

[16] S. Hoogendoorn, P. Bovy, and W. Daamen. Microscopic pedestrian wayfinding and dynamic modelling. In Schreckenberg and Sharma [35], pages 123–154.

[17] J. Illenberger, G. Flötteröd, and K. Nagel. Enhancing matsim with capabilities of within-day re-planning. In *IEEE Intelligent Transportation Systems Conference*, Seattle, WA, 2007.

[18] R. R. Jacob, M. V. Marathe, and K. Nagel. A computational study of routing algorithms for realistic transportation networks. *ACM Journal of Experimental Algorithms*, 4(1999es, Article No. 6), 1999.

[19] M. Jha, K. Moore, and B. Pashaie. Emergency evacuation planning with microscopic traffic simulation. Paper 04-2414, Transportation Research Board Annual Meeting, Washington, D.C., 2004.

[20] D. E. Kaufman, K. E. Wunderlich, and R. L. Smith. An iterative routing/assignment method for anticipatory real-time route guidance. Technical Report IVHS Technical Report 91-02, University of Michigan Department of Industrial and Operations Engineering, Ann Arbor MI 48109, May 1991.

[21] H. Klüpfel. The simulation of crowd dynamics at very large events – Calibration, empirical data, and validation. In Gattermann et al. [10].

[22] E. Kwon and S. Pitt. Evaluation of emergency evacuation strategies for downtown event traffic using a dynamic network model. Paper 05-2164, Transportation Research Board Annual Meeting, Washington, D.C., 2005.

[23] C. Lieberman, D. Kurowski, M. Avila, L. Ricci, N. Thomas, C. Collyer, and B. Aguirre. Conceptual framework for simulating the pedestrian evacuation behavior from buildings. Paper 05-2297, Transportation Research Board Annual Meeting, Washington, D.C., 2005.

[24] Q. Lu, B. George, and S. Shekhar. *Capacity Constrained Routing Algorithms for Evacuation Planning: A Summary of Results*, volume 3633 of *Lecture Notes in Computer Science*, pages 291–307. Springer Berlin / Heidelberg, 2005.

[25] MATSIM www page. MultiAgent Transport SIMulation, accessed 2008.

[26] K. Meister, M. Rieser, F. Ciari, A. Horni, M. Balmer, and K. Axhausen. Anwendung eines agentenbasierten Modells der Verkehrsnachfrage auf die Schweiz. In *paper presented at Heureka '08*, Stuttgart, Germany, March 2008.

[27] Y. Murakami, T. Ishida, T. Kawasoe, and R. Hishiyama. Scenario description for multi-agent

simulation. In *Autonomous agents and multiagent systems (AAMAS'03)*, Melbourne, Australia, July 2003.

[28] K. Nagel and M. Schreckenberg. A cellular automaton model for freeway traffic. *Journal de Physique I France*, 2:2221–2229, 1992.

[29] K. Nishinari, A. Kirchner, A. Nazami, and A. Schadschneider. Extended floor field CA model for evacuation dynamics. In *Special Issue on Cellular Automata of IEICE Transactions on Information and Systems*, volume E84-D, January 2001.

[30] X. Pan, C. Han, K. Dauber, and K. Law. A multi-agent based framework for the simulation of human and social behavior during emergency evacuations. *AI and Society*, 22(2):113–132, 2007.

[31] W. Predtetschenski and A. Milinski. *Planning for Foot Traffic in Buildings*. Amerind Publishing Co. Pvt. Ltd., New Delhi, 1978.

[32] B. Raney, N. Cetin, A. Völlmy, M. Vrtic, K. Axhausen, and K. Nagel. An agent-based microsimulation model of Swiss travel: First results. *Networks and Spatial Economics*, 3(1):23–41, 2003.

[33] B. Raney and K. Nagel. An improved framework for large-scale multi-agent simulations of travel behavior. In P. Rietveld, B. Jourquin, and K. Westin, editors, *Towards better performing European Transportation Systems*. Routledge, London, 2006. Similar version TRB preprint number 05-1846.

[34] A. Schadschneider, W. Klingsch, H. Klüpfel, T. Kretz, C. Rogsch, and A. Seyfried. Evacuation dynamics: Empirical results, modelling and applications. In B. Meyers, editor, *Encyclopedia of Complexity and System Science*. Springer, Berlin, to appear.

[35] M. Schreckenberg and S. D. Sharma, editors. *Pedestrian and Evacation Dynamics*. Proceedings of the 1st international conference, Duisburg, 2001. Springer, 2001.

[36] P. M. Simon, J. Esser, and K. Nagel. Simple queueing model applied to the city of Portland. *International Journal of Modern Physics C*, 10(5):941–960, 1999.

[37] U. Weidmann. *Transporttechnik der Fussgänger*, volume 90 of *Schriftenreihe des IVT*. Institute for Transport Planning and Systems ETH Zürich, 2 edition, 1993. in German.

Agent Performance in Vehicle Routing when the Only Thing Certain is Uncertainty

Tamas Mahr
Delft Technical University and Almende BV
Rotterdam, The Netherlands
T.Mahr@tudelft.nl

Jordan Srour
RSM Erasmus University
Rotterdam, The Netherlands
JSrour@rsm.nl

Mathijs de Weerdt
Delft Technical University
Delft, The Netherlands
M.M.deWeerdt@tudelft.nl

Rob Zuidwijk
RSM Erasmus University
Rotterdam, The Netherlands
RZuidwijk@rsm.nl

ABSTRACT

While intermodal transport has the potential to introduce efficiency to the transport network, this transport environment also suffers from a lot of uncertainty at the interface of modes. For example, trucks moving containers to and from a port terminal are often uncertain as to when exactly their container will be released from the ship, from the stack, or from customs. This leads to much difficulty and inefficiency in planning a profitable routing for multiple containers in one day.

In this paper, we examine agent-based solutions as a mechanism to handle job arrival uncertainty in the context of a drayage case at the Port of Rotterdam. We compare our agent-based solution approach to a well known on-line optimization approach and study the comparative performance of both systems across four scenarios of varying job arrival uncertainty. We conclude that when less than 50% of all jobs are known at the start of the day then an agent-based approach performs competitively with an on-line optimization approach.

1. INTRODUCTION

Scheduling the routes of trucks to pick-up and deliver containers is a complex problem. In general such Vehicle Routing Problems (VRPs) [19] are known to be NP-complete, and therefore inherently hard and time consuming to solve to optimality. Fortunately, these problems have a structure that can facilitate efficient derivation of feasible (if not optimal) solutions. Specifically, the routes of different trucks are more or less independent. Such "locality" in a problem is a first sign that an agent-based approach may be viable.

Modeling and solving a VRP by coordinating a set of agents can bring a number of advantages over more established approaches in the field of operations research even when using state-of-the-art mixed integer solvers such as CPLEX [7]. Agent advantages include the possibility for distributed computation, the ability to deal with proprietary data from multiple companies, the possibility to react quickly on local knowledge [5], and the capacity for mixed-initiative planning [1].

In particular, agents have been shown to perform well in uncertain domains. That is, in domains where the problem is continually evolving [5]. In the VRP, for example, a very basic form of uncertainty is that of job arrivals over time. To the best of our knowledge, however, the effect of even this basic level of uncertainty on the performance of agent-based planning in a realistic logistics problem has never been shown.

We think it is safe to assume, based on its long history, that current practice in operations research (OR) outperforms agent-based approaches in settings where all information is known in advance (static settings). However, in situations with a lot of uncertainty, agent-based approaches are expected to outperform these traditional methods [8].

In this paper we investigate whether a distributed agent-based planning approach indeed suffers less from job arrival uncertainty than a centralized optimization-based approach. Our main contribution is to determine at which level of job arrival uncertainty agent-based planning outperforms on-line operations research methods. These results can help transportation companies decide when to adopt an agent-based approach, and when to use an on-line optimization tool, depending on the level of uncertainty job arrivals exhibit in their daily business.

In Section 2 we provide a survey of current work on agent-based approaches to logistics problems. In Section 3 we then introduce the case of a transportation company near the port of Rotterdam. Based on this literature review and the specific nature of our case study VRP, we propose a state-of-the-art agent-based approach where orders are auctioned among trucks in such a way that each order is assigned to the truck that can most efficiently transport the container. Moreover, these trucks continuously negotiate among each other to exchange orders as the routing situation evolves. This agent-based approach is the topic of Section 4. In the following section, Section 5, we describe the centralized on-line optimization approach used in comparison to our distributed agent-based system. The structure of our test problems and the computational results are the topics of Section 6. In our final section we discuss the consequences of our results, summarize our advice to transportation companies, and give a direction for future work.

2. LITERATURE SURVEY

In their frequently cited 1995 paper, Fischer et al. argued that multi-agent models fit the transportation domain particularly well [5]. Their main reasons were that (i) the domain is inherently distributed (trucks, customers, companies etc.); (ii) a distributed agent architecture can cope with multiple dynamic events; (iii) commercial companies may be reluctant to provide proprietary data needed for global optimization and agents can use local information; and (iv) inter-company cooperation can be more easily facilitated by agents. To illustrate the idea, the authors also provided a detailed agent architecture for transportation problems that evolve over time thereby exhibiting uncertainty over time. This architecture makes a distinction between a higher and a lower architectural level. At the higher level, company agents negotiate over transportation requests to eliminate ill-fitting orders. On the lower level, truck agents (clustered per company) participate in simulated market places, where they bid on offered transportation orders. Truck agents use simple insertion heuristics to calculate their costs and use those costs to bid on auctions implementing an extended contract net protocol [17]. Although the heuristics that agents use to make decisions are rather crude, the authors suggested that in dynamic problems (problems with high uncertainty), such methods survive better than sophisticated optimization methods.

Fischer et al.'s bi-level approach recognizes that one shortcoming of a fully distributed system is that agents only have access to local information [5]. The need to balance between the omniscience of a centralized model and the agility of a distributed model, was similarly recognized by Mes et al. [12]. They also introduce a higher level of agents, but with a different role than the high-level agents of Fischer et al. Mes et al.'s two high-level agents (the planner and the customer agent) gather information from and provide information to agents assigned beneath them. The role of the higher level agents is to centralize information essential for the lower level agents to make the right decisions.

Some researchers have gone even further in proposing centralized agent-based models. These researchers focused on centralizing the problem information to be able of make better distributed decisions. In one of the few models that is actually applied in a commercial company, Dorer and Calisty cluster trucks geographically, using one agent per cluster [4]. This way, one agent plans for multiple trucks. They use insertion heuristics to initially assign orders to trucks, and then use cyclic transfers [18] to enhance the solution. In an even more centralized model, Leong and Liu use a fully centralized optimizer to initialize the agents [11]. The agents' role is then to change the plans as events are revealed. The authors analyze the performance of their model on a selection of Solomon benchmark sets, and show that it performs competitively.

As noted previously, however, the move towards centralization can hinder the ability of the agents to react quickly on local information. Given the uncertain environment of our problem, we are interested in the competitiveness of a system with fully distributed agents. One example of a fully distributed agent approach in the transportation domain is that of Brückert et al. They proposed a more detailed (holonic) agent model [1]. They distinguished truck, driver, chassis, and container agents that have to form groups (called holons) to serve orders. Already formed holons use the same techniques to allocate tasks as Fischer et al., but the higher agent level is omitted, since they model only a single company case. The main focus of their research is computer-human cooperative planning, and they do not test their model extensively against other models.

Generally, the decision to use a distributed approach is based on the expectation (included already in the reasons of Fischer et al.) that distributed models handle uncertainty better. The agent architecture in these fully distributed models is completely flat, the models avoid centralizing information, and agents can use only local information when making decisions. Having lost the power of using (partial) global information, distributed agents need other ways to enhance their performance.

In the model of Fischer et al., as well as in the models of many of their followers, agents use simple approximation techniques to make decisions. In the related domain of production planning Persson et al. embed optimization in the agents to improve local decisions [13]. They show that optimizing agents outperform the approximating agents, but they also show that central optimization still outperforms the optimizing, but distributed, agents.

While Persson et al. concentrated on making optimal decisions within agents, there is still a need to coordinate between the distributed agents. For example, in the transport problem context, when orders are assigned to trucks sequentially, at every assignment the truck with the cheapest insertion gets the order. Later, however, it might turn out that it would be cheaper to assign the same order together with newly arrived orders to another truck. From the truck point of view it means that trucks that bid early and win assignments might not be able to bid later on more beneficial (better fitting) orders. This problem is called 'the eager bidder problem' [16], and several researchers proposed alternative techniques to solve it. Kohout and Erol introduce an enhancement process that works between agents [9]. The process mimics a well known enhancement technique called 'swapping' or two-exchange [2]. Kohout and Erol implement this swapping process in a fully distributed way, and show that it yields significant improvement.

Perugini et al. extend Fischer's contract-net protocol to allow trucks to place multiple possibly-conflicting bids for partial routes [14]. These bids are not binding, trucks are requested to commit to them only when one of the bids is accepted by an order agent. Since auctions are not necessarily cleared before other auctions are started, agents have a chance to "change their mind" if the situation changes. This extension helps to overcome the eager bidder problem to some extent and thereby produces better results. Another possible way to tackle the same problem is to use leveled commitment contracts introduced by Sandholm and Lesser [15]. Leveled commitment contracts represent agreements between agents that can be withdrawn. If a truck agent finds a new order that fits better, it can decommit an already committed order and take the new one. Hoen and La Poutré employ truck agents that bid for new orders considering decommitting already assigned ones [6]. They show that decommitment yields more optimal plans in a single-company cooperative case.

Returning to Fischer's reasoning, however, the primary reason for using distributed agent models is that they are usually expected to outperform central optimization models in problem instances with high levels of uncertainty. Tak-

ing this for granted, researchers usually show that their distributed algorithm is better than the distributed algorithms of others. Experiments studying the behavior of distributed methods over varying levels of uncertainty in comparison to centralized optimization methods are generally absent from the literature.

If advanced swapping and decommitment techniques are used, can fully distributed agents perform competitively with (or better than) centralized optimization in highly uncertain settings? Can the time gained in doing local operations compensate for the loss of not considering crucial global information? In our opinion these questions have not been fully answered. In this paper, we construct a distributed agent model using the most promising techniques as identified in the agent literature and compare this approach via experiments on a real data set to a state-of-the-art centralized on-line optimization approach. The lack of appropriate comparisons between agent-based approaches and existing techniques for transportation and logistics problems possibly indicates a belief on the part of agent researchers that agent-based systems outperform traditional methods [3]. Our goal is to add credibility to this belief by studying a state-of-the-art agent-based system in comparison to a state-of-the-art centralized optimization approach for a real-world dynamic transportation problem. In the following section we define in detail the exact VRP that we use to study both the distributed agent-based and centralized optimization-based approaches.

3. VEHICLE ROUTING PROBLEM

Many of the agent-based approaches for vehicle routing problems are tested on generated data-sets. These data-sets are usually constructed to test specific features of the agent system - often focusing on the extreme ends of the performance spectrum. We, however, want to understand the potential of agent solutions in the highly uncertain real world. To that end we are fortunate to have access to operational data from a mid-sized Dutch logistics service provider (LSP) engaged in the road transport of sea containers. While the LSP that we study is active in several sectors, we focus only on the container division which has a fleet of around 40 trucks, handling an average of 65 customer orders each day.

The process of executing an order starts with receiving an order, generally one day before execution is required. While the orders are often called in one day early, the company does not generally use this information in planning routes or establishing schedules. This is due to the unreliable nature of the order information and the resulting uncertainty encountered during execution. An order is a customer request to the LSP for pickup and transport of a specific container from a container terminal (in the case of an import container) to the customer, with delivery within a certain time window. Arriving at the customer's requested location, the container is then unloaded, and the empty container is brought back to a container terminal or empty depot. This concludes the order, and the truck is ready for its next order. The process is reversed for export containers. What adds uncertainty to this process is that not all containers are available at the time indicated in the received order: either they have not physically left the ship at the expected time or they are delayed for administrative reasons, e.g. an unsettled payment or customs clearing. The LSP can only transport containers that have been released, and are allowed to leave the container terminal. For this reason it is hard to optimize the system in a traditional sense, since not all information is known beforehand, and will only become available at some point in time during the day.

The planning and control of operations is currently performed manually by a team of three human planners, who take care of order intake, arrange the proper amount of trucks based on the expected workload, and assign current orders to trucks. Given the primarily manual method of operations, the addition of a computerized decision support system may greatly enhance the profitability and scalability of the LSP's operations.

To formalize the structure of this case study problem we make several formal assumptions:

- Each demand is available for scheduling at the time it is announced. The announcement of a demand includes all information on: the pick-up location (zipcode), the customer location (zipcode), return drop-off location (zipcode), and the required time windows for arrival at each of these three locations.
- Loading and unloading at the terminals and customer takes time. Picking up a container requires 60 minutes; servicing the container at the customer requires 60 minutes; and returning a container to the final terminal takes 30 minutes.
- All travel times are measured according to data on the Benelux road network.
- No time window violations are allowed; if a job is going to violate time windows then it is rejected at a penalty.
- The penalty for rejecting a job is equal to the loaded time of the job. Given the problem structure defined here, loaded time serves as a proxy for revenue.
- Given the demand structure, the truckload nature of the problem, and the fact that the truck must remain with the container at the customer location, we bundle the pick-up, drop-off, and return activities into one job. The loaded time of a job is then the time spanning the arrival at the pick-up terminal through the completion of service at the return terminal - including all loading and unloading times.
- All trucks in the fleet are equivalent.

Given this context, the objective of this vehicle routing problem is to derive a schedule in real-time that serves as many jobs as possible at the least cost. Cost is defined here in terms of time, as the time spent traveling empty (i.e. non-revenue generating travel) to serve all jobs in addition to the loaded time penalty affiliated with rejecting jobs. By adding a penalty for rejecting jobs equal to the loaded distance (in terms of time) of each job, the obvious cost-minimizing solution of rejecting all jobs is avoided. In this regard, it is important to note that in our setting the loaded distance of an order is approximately four times as great as the empty distance incurred in serving that job.

4. AGENT-BASED APPROACH

Based on the agent-based modeling literature and the assumptions related to our problem as introduced in Section

3, our goal is to design, using selected techniques from the literature, a distributed agent model that can outperform a centralized optimization approach. Since we are primarily interested in distributed agent models, we use an uncompromisingly flat architecture: no agents can concentrate information from a multitude of other agents. The global idea of our agent-based planning system is to apply an advanced insertion heuristic in a distributed setting and combine this with two heuristics for making (local) improvements: substitution of orders, and random attempts for re-allocation of orders. The only two kinds of agents that participate in this planning system are *truck* agents and *order* (or *container*) agents.

Our order agents represent container orders. The particularity of container orders is identical to the real-world case of the previous section in that they are described by the three stops required: a pick up at a sea-terminal, a delivery at the customer's, and a drop-off return at a possibly different sea-terminal. With each of the three stops there is a time window and a service time associated, which are obeyed by the trucks. Truck agents represent trucks with a single chassis, which means that they can transport only one order at a time. They make plans in order to transport as many containers as they can.

Order agents hold auctions in order of their arrival, and truck agents bid in these auctions. This results in partially parallel sequential auctions. Trucks may bid on multiple orders at the same time; these bids are not binding. If a truck happens to win more than one order, it takes only the first one. All the other orders it won parallel to the first one are rejected, which results in the rejected order agents starting a new auction. Truck agents ultimately accept only one winning bid on parallel auctions as all bids submitted in parallel are highly dependent on the order of previously won and accepted bids. In this way, in the end, the orders are auctioned sequentially, even if they happen to arrive at the same time.

To clear an auction, order agents choose the best bid as winner, and respond positively to the winner and negatively to the others. For this we chose a one-shot auction (and more specifically, a Vickrey auction [20]) for its computational efficiency, as in the model of Hoen and La Poutré [6]. If the winner confirms the deal, a contract is made. These contracts are semi-binding, so truck agents might break it in order to achieve a better allocation.

At the heart of the agent model are the decisions truck agents make. The most important decision they have to make is the bid they submit for a given order. Every truck agent submits a bid that reflects its cost associated with transporting the given order. This cost is a quantity in the time domain. To calculate it, a truck considers *inserting* the new order into its plan, or alternatively *substituting* one of the already contracted orders by the new one.

To calculate the cost of insertion, the truck agent tries to insert the new order in-between every two adjacent orders in their plan (see Figure 1), plus at the beginning and the end. At every position, it calculates the amount of extra empty time it needs to drive if this order is inserted there. Suppose that an agent considers the position between container i and j, and calculates that the empty time the truck needs to travel to pick up j after returning i is d_{ij}. Here we use d_{ij} to represent the distance (in time) between the two jobs i and j, and d_{ii} to denote the loaded distance of job i. The

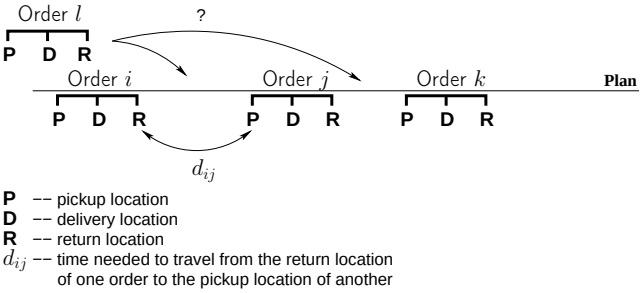


Figure 1: Insertion

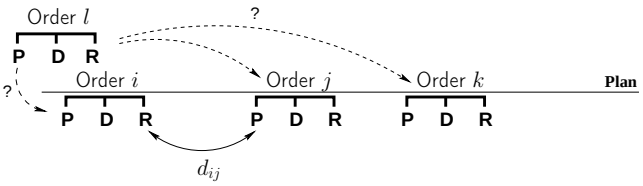


Figure 2: Substitution

amount of extra empty time the truck would need to drive for container l then equals $ins^l_{ij} = d_{il} + d_{lj} - d_{ij}$.

In addition to insertion, a truck agent also considers substitution (analogous to what others call decommitment). To calculate the cost of substituting one of the already contracted orders by the new one, it sums up the cost components. The first component is the insertion cost of the new order at the place of the substituted order, the second component is the lost profit on the substituted order, and the third component is a penalty term. For example, we compute the cost of substituting order j with order l ($subs^l_j$) in Figure 4. Here $subs^l_j = ins^l_{ik} + profit_j + d_{jj}$. The insertion term ins^l_{ik} is the same as defined above. The value of $profit_j$ is the difference of the price received for order j and its insertion cost: $profit_j = price_j - ins^j_{ik}$. This term represents the market position of the substituted order in the bid. If the competition for order j is fierce, the profit on j would be low (since the second-best bid was hardly higher than the winning bid). This results in a low substitution cost, therefore such orders are more likely to be substituted. An order that is well suited for a specific truck is likely to produce a high profit for that truck, therefore it will have a high substitution cost. The last term in this expression, the amount of loaded time of order j, serves as a penalty on substituting that job. Using such a penalty discourages the substitution of long orders that may be harder to fit somewhere else. Additionally, the orders that are finally rejected (those that do not manage to make a contract with any truck agents) will be shorter, which will result in a better total cost. Algorithm 1 describes how new orders are dealt with.

In addition to bidding on auctions for new orders, truck agents have another way to enhance the overall solution. At random time intervals, every truck randomly selects an order in its plan and releases it. Trucks never select the order they are currently serving and also not one, for which the execution is about to begin (the pick-up time of the container is less than 10 seconds away – this small time

Algorithm 1 Insertion and substitution of orders

1. Compute the extra costs for every possible insertion and every possible substitution
2. Order the merged list of insertions and substitutions in increasing order of these costs
3. Iterate over this list
   (a) If the new order's time windows are violated, continue with the next alternative.
   (b) If a time window of an order after the new one is violated, continue with the next alternative.
   (c) Else the cheapest feasible position is found. Return this position.

buffer is selected to provide as much opportunity for route improvement as possible). Note, the same time limit is also applied to the insertion and substitution decisions explained earlier. An order agent that is released (just as those order agents that are substituted) initiates a new auction to find another place. In most cases, these auctions result in the very same allocation as before the release. Nevertheless, sometimes they do manage to find a better place and make a contract with another truck.

Whenever an order agent finalizes a contract with a truck agent, it sends a message to all other order agents to notify them about the changed plan of the given truck. This is important for order agents that do not have a contract yet. Any change in the trucks' plans may be their chance to find their place in a truck. Those order agents will start an auction in response to the notification message in the hope of finally making a contract.

To summarize the agent-based approach, let us list the main techniques that characterize it:

- Orders are allocated to trucks via second-price auctions sequentially, at the time they become known to the agent system.
- Truck agents consider insertion and substitution of new orders in their plan. Substituted orders are released from the truck. Released order agents hold a new auction to find another place. If a truck cannot deliver an order within the time windows, it rejects it.
- Truck agents randomly release contracted order agents. Randomly released order agents also hold a new auction to find a place.
- Order agents notify each other whenever they change the plan of a truck (make a contract). Rejected orders (without a contract) thereby get a chance to hold a new auction and find a truck.

To evaluate this approach, we implemented a real-time truck simulator that we connected to the agent system. Every truck agent assumes responsibility for a simulated truck. In the coupled agent-truck-simulator system, agents send plans to trucks for execution. Simulated trucks drive along the road network of the Benelux as the plans prescribe. They periodically report their position as well as their activities to the agents. This way truck agents can follow the execution of the plans and make decisions with the knowledge of what is happening in the (simulated) world.

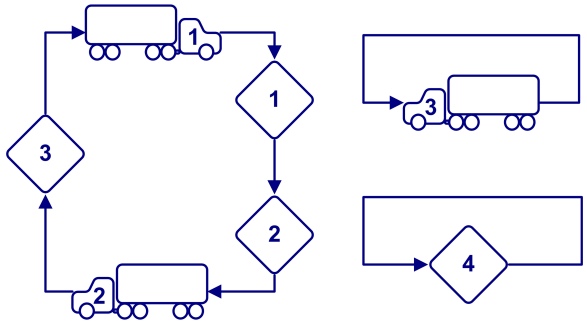


Figure 3: Cycles in the MIP solution structure.

Finally, we have a third element in the system, whose role is to monitor both the agents and the simulator, thereby gathering all information necessary to evaluate the performance of the agents, and to calculate the total cost of the routing. Just as described in Section 3, the ultimate objective of the agents is to minimize the total cost of the routing which is specified in terms of the time trucks travel empty plus the loaded-travel-time penalty associated with rejecting a container. The next section describes the on-line optimization approach that is used in comparison to the agent-based approach, based on this total cost.

5. ON-LINE OPTIMIZATION APPROACH

To estimate the value of the agent-based solution approach (described in Section 4), we study it in comparison to an optimization-based solution approach, reflective of those currently embedded in commercially available vehicle routing decision support software (DSS). We therefore examine two optimization based solution approaches: (i) a mixed-integer program for solving the static *a priori* case in order to provide a baseline benchmark, and (ii) an on-line optimization approach, comparable to the agent approach, and designed to represent current vehicle routing DSS.

At the core of both the static *a priori* solution and the on-line optimization is a mixed integer program (MIP) for a truck-load vehicle routing problem with time windows, which is given to CPLEX [7]. This MIP is based on the formulation put forth by Yang et al. [21]. The complete description of our modifications to Yang et al.'s MIP is the focus of this section. Before introducing the notation and mathematical formulation for this problem, we begin with a small example to illustrate exactly how Yang et al.'s MIP works to exploit the structure of this truckload pick-up and delivery problem with time windows.

Imagine a scenario with three trucks and four jobs. The model of Yang et al. is constructed such that it will find a set of least cost cycles describing the order in which each truck should serve the jobs. For example, as depicted in Figure 3, the outcome may be a tour from truck 1 to job 1, then job 2, then truck 2, then job 3, then back to truck 1. This would indicate that truck 1 serves job 1 and 2, while truck 2 serves job 3. The cycle including only truck 3 indicates that truck 3 remains idle. Similarly, the cycle including only job 4 indicates that job 4 is rejected.

Given the assumptions in Section 3, we designate the fol-

lowing notation for the given information.

- K the total number of vehicles available in the fleet.
- N the total number of known demands.
- d_{ij} as introduced in 4, the travel time required to go from demand i's return terminal to the pick-up terminal of demand j. Note, if $i = j$ then the travel time d_{ii} represents the loaded distance of job i.
- d^k_{0i} the travel time required to move from the location where truck k started to the pick-up terminal of demand i.
- d^k_{iH} the travel time from the return terminal of demand i to the home terminal of vehicle k.
- v^k the time vehicle k becomes available.
- l_i the loaded time required of job i (time from pick up at originating terminal to completion of service at the return terminal). Note, $l_i = d_{ii}$.
- τ_i^- earliest possible arrival at demand i's pick-up terminal.
- τ_i^+ latest possible arrival at demand i's pick-up terminal.
- M a large number set to be $2 \cdot \max_{i,j}\{d_{ij}\}$.

Note: τ_i^- and τ_i^+ are calculated to ensure that all subsequent time windows (at the customer location and return terminal) are respected. Given the problem of interest, we specify the following two variables.

- x_{uv} a binary variable indicating whether arc (u, v) is used in the final routing; $u, v = 1, \ldots, K + N$.
- δ_i a continuous variable designating the time of arrival at the pick-up terminal of demand i.

Using the notation described above, we formulate a MIP that explicitly permits job rejections, based on the loaded distance of a job.

$$\min \quad \sum_{k=1}^{K}\sum_{i=1}^{N} d^k_{0i}x_{k,K+i} + \sum_{i=1}^{N}\sum_{j=1}^{N} d_{ij}x_{K+i,K+j} + \sum_{i=1}^{N}\sum_{k=1}^{K} d^k_{iH}x_{K+i,k} \tag{1}$$

such that

$$\sum_{v=1}^{K+N} x_{uv} = 1 \qquad \forall u = 1, \ldots, K+N \tag{2}$$

$$\sum_{v=1}^{K+N} x_{vu} = 1 \qquad \forall u = 1, \ldots, K+N \tag{3}$$

$$\delta_i - \sum_{k=1}^{K}(d^k_{0i} + v^k)x_{k,K+i} \geq 0 \qquad \forall i = 1, \ldots, N \tag{4}$$

$$\delta_j - \delta_i - Mx_{K+i,K+j} + (l_i + d_{ij})x_{K+i,K+i} \geq l_i + d_{ij} - M \qquad \forall i, j = 1, \ldots, N \tag{5}$$

$$\tau_i^- \leq \delta_i \leq \tau_i^+ \qquad \forall i = 1, \ldots, N \tag{6}$$

$$\delta_i \in \mathbb{R}^+ \qquad \forall i = 1, \ldots, N \tag{7}$$

$$x_{uv} \in \{0, 1\} \qquad \forall u, v = 1, \ldots, K+N \tag{8}$$

In words, the objective (1) of this model is to minimize the total amount of time spent traveling without a profit generating load. This objective is subject to the following seven constraints:

(2) Each demand and vehicle node must have one and only one arc entering.

(3) Each demand and vehicle node must have one and only one arc leaving.

(4) If demand i is the first demand assigned to vehicle k, then the start time of demand i (δ_i) must be later than the available time of vehicle k plus the time required to travel from the available location of vehicle k to the pick up location of demand i.

(5) If demand i follows demand j then the start time of demand j must be later than the start time of demand i plus the time required to serve demand i plus the time required to travel between demand i and demand j; if however, demand i is rejected, then the pick up time for job i is unconstrained.

(6) The arrival time at the pick up terminal of demand i must be within the specified time windows.

(7) δ_i is a positive real number.

(8) x_{uv} is binary.

Mathematically this model specification serves to find the least-cost (in terms of time) set of cycles that includes all nodes given in the set $\{1, \ldots, K, K+1, \ldots, K+N\}$. We define $x_{uv}, (u, v = 1, \ldots, K+N)$ to indicate whether arc (u, v) is selected in one of the cycles. These tours require interpretation in terms of vehicle routing. This is done by noting that node $k, (1 \leq k \leq K)$ represents the vehicle k and node $K + i, (1 \leq i \leq N)$ corresponds to demand i. Thus, each tour that is formed may be seen as a sequential assignment of demands to vehicles respecting time window constraints.

The model described above is used to provide the optimal (yet realistically unattainable) lower bound for each day of data in each scenario. We denote this approach as the static *a priori* case. In this case, we obtain the route and schedule as if all the jobs are known and we have hours to find the optimal solution. Thus, not only is this lower bound realistically unattainable due to a relaxation on the amount of information available, but also due to a relaxation on the amount of time available to CPLEX for obtaining the optimal solution. In this way, because the problem instances are relatively small (note, using this MIP structure CPLEX can handle a maximum of about 100 jobs and about 50 Trucks, yet our instances are only 34 trucks and 65 jobs) we are able to uncover the optimal solution for all 26 problem instances across all four uncertainty scenarios.

In order to provide a fair comparison with the agent-based approach, the MIP is then manipulated for use in on-line operations. In our on-line approach, this MIP is invoked at 30 second intervals. At each interval, the full and current state of the world is captured, and then encoded in the MIP. This "snapshot" of the world includes information of all jobs that are available and in need of scheduling, as well as the forecasted next available location and time of all trucks. The MIP is then solved via a call to CPLEX. The decision to use 30 second intervals was driven by the desire to be comparable to the agent-based approach while still providing CPLEX enough time to find a feasible solution for each snapshot problem. The solution given by CPLEX is parsed and any jobs that are within two intervals (i.e. 60 seconds) of being late (i.e. missing the time specified by δ_i in the latest plan) are permanently assigned if travel is not commenced in the next interval. Any jobs that were designated for rejection in the solution are rejected only if they are within two intervals of violating a time window;

otherwise they are considered available for scheduling in a subsequent interval. The procedure continues in this fashion until the end of the working day at which point all jobs have been served or rejected.

The test problems and the results from the static *a priori* benchmark, the on-line optimization, and the agent-based solution approach as applied to these test problems are the topic of the next section.

6. COMPUTATIONAL EXPERIMENTS

In this section we report the computational results on the performance of the agent-based approach in comparison to the optimization-based approach. The first subsection (6.1) describes how the test problems were generated and in subsection 6.2 we present the results of these tests.

6.1 Test Problems

The data we used for our experiments was based on data provided to us by the LSP described in Section 3. In all, we were given the execution data from January 2002 to October 2005 as well as the data from January 2006 through March 2006. We could not, however, simply use this data in its raw form. We first had to make multiple corrections to the customer address fields as many addresses referred to postal boxes and not to the physical terminal locations. After cleaning the address fields, we then extracted a random sample of jobs from the original data-set in order to generate a set of 26 days with 65 orders per day. The company from which these data are taken serves between 50 and 80 jobs per day, thus 65 jobs per day represents the average daily job load.

Just as discussed before, each order consists of a pickup location, customer location, and return location. To standardize the data for our experimental purposes we specified time windows at all locations as follows: for the terminals (the pickup and return locations) the time windows span a full twelve hour work day from 6am to 6pm and the time windows at the customer locations are always 2 hours. The start of the 65 customer time windows occurs throughout the working day in accordance with the data provided by the LSP, which roughly follows a uniform distribution. Given the variation in customer locations, the workload per day varies similarly. On average each job requires approximately 4.2 hours of loaded distance. When the routing is optimal in the case that all jobs are known at the start of the day the average empty time per job is approximately 25 minutes.

Given our interest in determining how the agent solution performs on this pick-up and delivery problem with time windows and order arrival uncertainty, we further rendered our 26 days of data into four separate scenarios with varying levels of order arrival uncertainty. This was done by altering the arrival times of the orders, i.e., the time at which the order data is revealed to the LSP. We generated these points in times over the day using a uniform distribution. We used such a uniform distribution as the original data did not show a fit with other distributions. The four different scenarios reflecting different levels of order arrival uncertainty were:

Scenario A: All orders (100%) are known at the start of the working day, 6AM.

Scenario B: About half of the orders (50%, selected randomly from the 65 jobs) are known at the start of the working day, 6AM. The other half of the orders arrive two hours before the start of the customer location time window (i.e., four hours before the end of the customer location time window, leaving slightly less than two hours on average before the latest departure time from the pickup location).

Scenario C: Only seven of the jobs (10%, selected randomly from the 65 jobs) are known at the start of the working day, 6AM. The remaining 58 jobs arrive two hours before the start of the customer location time window.

Scenario D: None of the jobs (0%) are known at the start of the working day. All 65 jobs arrive two hours before the start of the customer location time window.

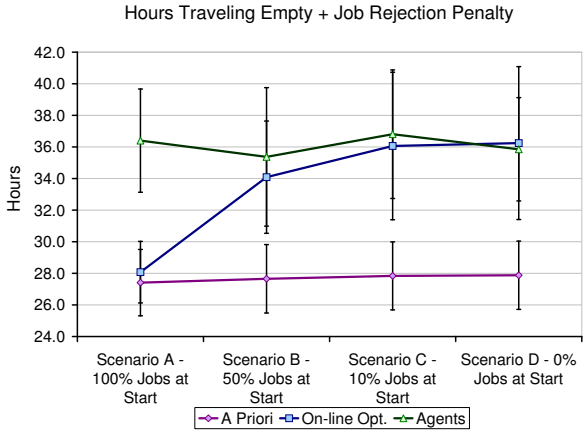


Figure 4: Mean, over 26 days, of the total cost for the three approaches across the four scenarios. Bars indicate ± standard deviation.

If we classify these scenarios in terms of the effective degree of dynamism for vehicle routing problems with time windows as developed by Larsen et al. in 2002 [10] then values of dynamism for Scenarios A, B, C, and D are .5, .7, .8, and .9, respectively. Noting that this form of measuring uncertainty may range from 0 to 1 with 1 being the most uncertain, then we may say that our test problems range from partially uncertain to mostly uncertain.

6.2 Computational Results

All three solution approaches were applied to each of the 26 days of data in the four scenarios. The mean cost over the 26 days of these experiments may be seen in Figure 4. From this graphical depiction, the on-line optimization procedure clearly outperforms the agents only in Scenario A. In fact, in Scenario A in which all information is known at the start of the day, the on-line optimization performs at a level quite close to the realistically unattainable benchmark optimal. The on-line optimization does not, however, achieve optimal in Scenario A as the snapshot problem in the first 30 second interval represents the full problem size. A size for which finding the optimal solution in thirty seconds is quite difficult. In all cases, CPLEX does, however, provide a feasible solution which can then be improved in future intervals. In the remaining three scenarios, however,

Table 1: Mean ± standard error over 26 days for on-line optimization and the agent-based approach on the total cost for scenarios A, B, C, and D.

	A	B	C	D
On-line Opt.	28.07 ± .38	34.09 ± .70	36.06 ± .92	36.24 ± .95
Agents	36.4 ± .64	35.37 ± .86	36.81 ± .80	35.85 ± .64

Table 2: Results of the t-test on the null hypothesis that the means of the total cost of the two datasets are equal (with .05 significance).

	A	B	C	D
Calculated t-value	11.16	1.16	.61	.34
Tabulated t-value	2.01	2.01	2.01	2.01
Result	Reject	Fail to Reject	Fail to Reject	Fail to Reject

the agents perform at a level competitive to the on-line optimization.

To fully understand the competitive nature of the agents in the dynamic settings of Scenarios B, C, and D a t-test was performed to determine if the average total cost of the routing solutions were statistically equivalent. The results of these tests may be seen in Tables 1 and 2. From these results we may conclude that for the reality-based datasets used in this study, agent-based solution approaches perform competitively with the on-line optimization when at least half of the jobs is unknown at the start of the day.

While the study of total cost and associated t-test results are promising for the agent approach, we must also look at the portion of this total cost due to the job rejection penalty and the portion of the cost due to empty travel time. Figure 5 depicts the penalty of rejected jobs on the left axis and the number of jobs rejected on the right axis. Note, we do not include the *a priori* optimal in this figure as no jobs were rejected using this approach. While the on-line optimization demonstrates a clear trend in the number of rejections (the more dynamic the setting the more jobs are rejected at a higher penalty), the agent approach does not demonstrate any trend. In comparing Figure 6 and Figure 5, it is clear that this irregular job rejection trend of the agent approach is having a significant impact on the trend in the total cost of the agent approach (see Figure 4).

Figure 6 depicts the average number of hours spent traveling empty in the routing solution provided by each approach in the four scenarios. From this figure, all three approaches show a general trend toward an increased level of empty travel with an increased level of uncertainty. Interestingly, however, the agent approach shows far more stability in this regard. In this sense we may conclude that despite the agents' poor performance in our less uncertain settings, they are, however, less susceptible than on-line optimization to the effects of high uncertainty. Yet, in the end, both systems perform comparatively well in the most uncertain setting.

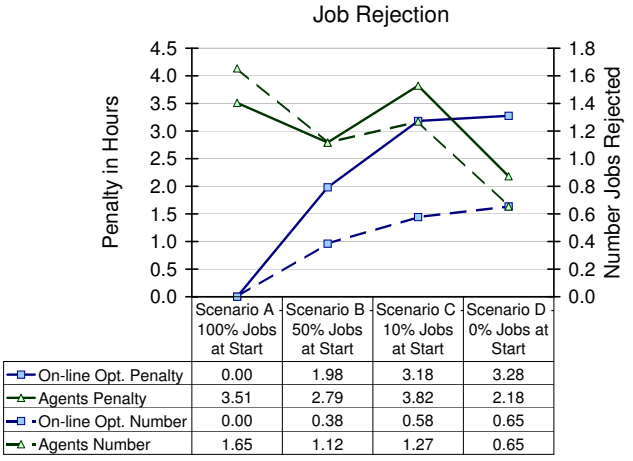


Figure 5: Job rejection in terms of penalty and number of jobs rejected for the on-line optimization and agent approaches across four scenarios.

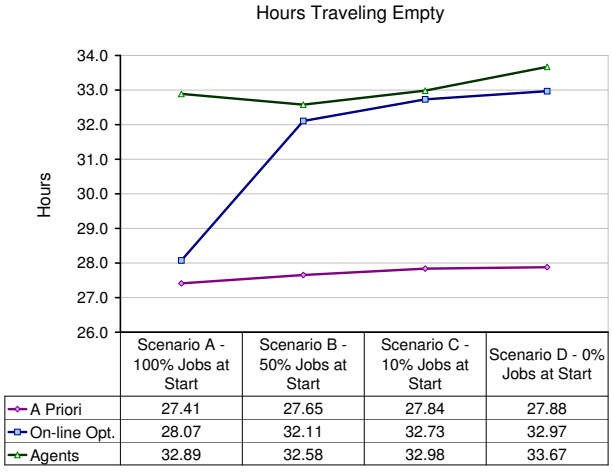


Figure 6: Time spent traveling empty for the three approaches across the four scenarios.

7. DISCUSSION

In this paper, we studied an on-line truckload vehicle routing problem arising from a real-world case study. We proposed a state-of-the-art agent-based solution approach and compared that approach to a well known on-line optimization approach. The computational results, from 26 days of data spanning four different scenarios representing various levels of job arrival uncertainty, indicate that the agent-

based approach is highly competitive in cases where less than 50% of the jobs are known in advance.

Given these results, agents should be considered as a viable decision support mechanism for transportation planners that must cope with uncertain job arrivals. If, however, the job arrival environment is relatively static, that is more than half of the jobs are known at the start of the day, then optimization should remain the tool of choice. Admittedly, this recommendation carries the following caveat. The agents do suffer a certain level of instability as reflected in the lack of a trend in job rejections relative to the level of uncertainty. The reason is that while job rejection is explicitly handled in the optimization model, it is implicit in the agent model. When an agent rejects an order, it has no way of knowing whether other agents will reject it too. In general, it is therefore more difficult to implement a global notion such as the number of rejected orders in an agent approach. In practice, a transportation provider must be very explicit about routing priorities. If a consistent or predictable level of job rejections is important then on-line optimization is a better choice.

One of the reasons that the agent-based solution performs consistently in terms of empty distance traveled is because of the sequential auction method used to handle jobs that arrive simultaneously. Thus, in Scenario A, in which the uncertainty is low, the agents must run many auctions at the start of the day; on-line optimization on the other hand may exploit all of this information at once to obtain a near optimal solution. In Scenario D, on the other hand, the agents approach the auctions in very much the same way as in Scenario A except that they are spread more evenly over time. In contrast the on-line optimization is forced to adopt job assignments that may preclude the assignment of jobs arriving late in the day.

In short, agent-based systems perform well in settings where less than half of all jobs are known in advance. Agents do, however, present issues concerning tractability in terms of rejected jobs. The number and penalty of rejected jobs is particularly variable with no clear trend across the four scenarios. Finally, in steep contrast to the online optimization, the agents used in this study are not well suited to exploit large batches of job arrivals; agents tend to perform better when a small number of jobs arrive evenly spaced through out the planning horizon.

Noting from these cases the impact of clumped job arrivals on the two approaches brings us to our first extension of this work. We recommend that both systems be tested across several problem sizes and a variety of uncertain job arrival patterns to truly understand the effect of clumped job arrivals.

Turning now to the theme of uncertainty, job arrival uncertainty as studied here represents only one narrow definition of uncertainty. A simple extension to this definition by including variable numbers of jobs across the days (i.e. each day would have a different number of jobs taken from the range 50 to 80) will provide additional insight on the strengths and weaknesses of agents in handling uncertainty. Furthermore examining other sources of uncertainty in the transportation domain, such as loading, unloading, and travel time variability, will not only add realism to the study, but will also yield a more robust view on the benefits and drawbacks of an agent approach as compared to centralized approaches.

Another extension of this work is the introduction of optimization into the agent approach. In this way, the agents may be able to capitalize on the benefit of optimization in less uncertain situations and the benefit of local heuristics in more uncertain situations. We conclude by stating that agent-based approaches may have even greater benefits when we consider modeling other forms of uncertainty such as travel time uncertainty, loading and unloading time uncertainty, and so forth. The field for agent-based approaches to the VRP is wide open, but should also be carefully explored to ensure that the practical everyday needs of real-world transport planners are met.

Acknowledgments

The authors would like to thank Hans Moonen for his help in retrieving and constructing the data set used for the experiments in this paper. Furthermore, this work is partially supported by the Technology Foundation STW, applied science division of NWO, and the Ministry of Economic Affairs of the Netherlands. Finally, the authors gratefully acknowledge the comments provided by three anonymous reviewers.

8. REFERENCES

[1] H.-J. Bürckert, K. Fischer, and G. Vierke. Holonic transport scheduling with Teletruck. *Applied Artificial Intelligence*, 14(7):697–725, 2000.

[2] J.-F. Cordeau, G. Desaulniers, J. Desrosiers, M. M. Solomon, and F. Soumis. VRP with time windows. In *The vehicle routing problem*, pages 157–193. Society for Industrial and Applied Mathematics, 2001.

[3] P. Davidsson, L. Henesey, L. Ramstedt, J. Törnquist, and F. Wernstedt. An analysis of agent-based aproaches to transport logistics. *Transportation Research Part C*, 13(4):255–271, 2005.

[4] K. Dorer and M. Calisti. An adaptive solution to dynamic transport optimization. In *Proc. of 4th Int. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2005)*, pages 45–51, New York, NY, USA, 2005. ACM Press.

[5] K. Fischer, J. P. Muller, M. Pischel, and D. Schier. A model for cooperative transportation scheduling. In *Proc. of the 1st Int. Conf. on Multiagent Systems*, pages 109–116, Menlo park, California, 1995. AAAI Press / MIT Press.

[6] P. J. Hoen and J. A. L. Poutré. A decommitment strategy in a competitive multi-agent transportation setting. In *Proc. of 2nd Int. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2003)*, pages 1010–1011, New York, NY, USA, 2003. ACM Press.

[7] C. Incorporated. Using the CPLEX Callable Library and CPLEX Mixed Integer Library, 1992.

[8] N. Jennings and S. Bussmann. Agent-based control systems: Why are they suited to engineering complex systems? *Control Systems Magazine, IEEE*, 23(3):61–73, 2003.

[9] R. Kohout and K. Erol. In-time agent-based vehicle routing with a stochastic improvement heuristic. In *AAAI '99/IAAI '99: Proc. of the 16th national conference on Artificial Intelligence and the 11th on Innovative Applications of Artificial Intelligence*, pages

864–869, Menlo Park, CA, USA, 1999. American Association for Artificial Intelligence.

[10] A. Larsen, O. Madsen, and M. Solomon. Partially dynamic vehicle routing-models and algorithms. *Journal of the Operational Research Society*, 53:637–646, 2002.

[11] H. W. Leong and M. Liu. A multi-agent algorithm for vehicle routing problem with time window. In *Proc. of the ACM Symposium on Applied Computing (SAC 2006)*, pages 106–111, New York, NY, USA, 2006. ACM Press.

[12] M. Mes, M. van der Heijden, and A. van Harten. Comparison of agent-based scheduling to look-ahead heuristics for real-time transportation problems. *European Journal of Operational Research*, 181(1):59–75, 2007.

[13] J. A. Persson, P. Davidsson, S. J. Johansson, and F. Wernstedt. Combining agent-based approaches and classical optimization techniques. In *Proc. of the European workshop on Multi-Agent Systems (EUMAS 2005)*, pages 260–269, 2005.

[14] D. Perugini, D. Lambert, L. Sterling, and A. Pearce. A distributed agent approach to global transportation scheduling. In *IEEE/WIC Int. Conf. on Intelligent Agent Technology (IAT 2003)*, pages 18–24, 2003.

[15] T. W. Sandholm and V. R. Lesser. Leveled commitment contracts and strategic breach. *Games and Economic Behaviour*, 35:212–270, 2001.

[16] M. Schillo, C. Kray, and K. Fischer. The eager bidder problem: a fundamental problem of DAI and selected solutions. In *Proc. of 1st Int. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2002)*, pages 599–606, New York, NY, USA, 2002. ACM Press.

[17] R. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12):1104–1113, 1980.

[18] P. Thompson and H. Psaraftis. Cyclic transfer algorithms for multivehicle routing and scheduling problems. *Operations research*, 41(5), 1993.

[19] P. Toth and D. Vigo, editors. *The Vehicle Routing Problem*. SIAM Monographs on Discrete Mathematics and its Applications. Society for Industrial and Applied Mathematic, 2002.

[20] W. Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961.

[21] J. Yang, P. Jaillet, and H. Mahmassani. On-line algorithms for truck fleet assignment and scheduling under real-time information. *Transportation Research Record*, 1667:107–113, 1999.

Exploring the potential of multiagent learning for autonomous intersection management

Matteo Vasirani
Artificial Intelligence Group
University Rey Juan Carlos
Madrid, Spain
matteo.vasirani@urjc.es

Sascha Ossowski
Artificial Intelligence Group
University Rey Juan Carlos
Madrid, Spain
sascha.ossowski@urjc.es

ABSTRACT

The problem of advanced intersection management is being discovered as a promising application field for multiagent technology. In this context, drivers interact autonomously with a coordination facility that controls the traffic flow through an intersection, with the aim of avoiding collisions and minimizing delays. This is particularly interesting for the case of autonomous vehicles that are controlled entirely by agents, a scenario that will become possible in the near future.

In this paper, we seize the opportunities for multiagent learning offered by such a scenario, by introducing a coordination mechanism where teams of agents decentrally coordinate their velocities when approaching the intersection. We show that this approach enables the agents to improve the intersection efficiency, by reducing the average travel time and so alleviating traffic congestions.

1. INTRODUCTION

Traffic congestion is a costly problem of cities in all developed countries. Many human-centered instruments and solutions (e.g. message signs, temporary lane closings, maximum velocity changes), are deployed in highways and roads in order to speed up the traffic flow. Nevertheless, in line with the recent advances of telematic infrastructures, the problem of road traffic management is being discovered as a promising application field for multiagent technology [1]. Multiagent systems (MAS) are the ideal candidates for the implementation of road traffic management systems, due to the intrinsically distributed nature of traffic-related problems.

In this context, the problem of advanced intersection management, where drivers interact autonomously with a coordination facility that controls the traffic flow through an intersection so as to avoid collisions while minimizing delays, is receiving more and more attention.

In [2] is presented a reservation-based system in which vehicles request an intersection manager to reserve the necessary time slots during which they may pass through the intersection. This work opens many possibilities for multiagent learning, with the goal of improving the efficiency of intersections.

In this paper, we present a coordination mechanism based on Probability Collectives (PC) [10]. With such an approach, teams of agents decentrally coordinate their velocities during their approximation to the intersection, with the aim of reducing the average travel time by making better, non-conflicting, reservations.

The paper is structured as follows: in section 2 we present the intersection management problem, as proposed by Dresner and Stone [2, 3]; in section 3 we introduce our proposal for making the agents learn to coordinate their actions in order to improve the intersection efficiency; in section 4 we show the results of the experiments and we discuss the possibility of improving them. Finally in section 5 conclusions and future work is outlined.

2. RESERVATION-BASED INTERSECTION MANAGEMENT

The reservation-based system proposed in [2] consists of two different kind of agents: intersection managers and driver agents. An intersection manager is responsible for managing the vehicles that want to pass through the intersection, by assigning the necessary time slots; a driver agent is responsible for controlling the vehicles to which it is assigned.

Each driver agent, when approaching the intersection, "calls ahead" the intersection manager and requests a reservation of space and time in the intersection. Such a request contains the necessary information to simulate the vehicle journey through the intersection, such as the vehicle properties (vehicle ID, vehicle size,...) as well as some properties of the proposed reservation (arrival time, arrival velocity, type of turn, arrival lane, arrival road segment,...).

The intersection manager then determines whether or not a request is feasible, by checking the confirmed reservations that are stored in its database. If the request is confirmed by the intersection manager, the driver agent stores the reservation details and tries to meet them. Otherwise, it slows down and makes another request at a later time.

The reservation system offers many opportunities for improving the efficiency of intersection, by incorporating learning mechanisms in the agents of such scenario [4]. For example, since the intersection manager serves the requests in a "first-come-first-served" fashion, it is possible to relax this constraint and allow the intersection manager to respond to requests at a later time. In this way the intersection manager can evaluate more competing requests at the same time and make a more well-informed decision.

While the learning opportunities for the intersection manager are of the form of *single agent learning*, the very *multi-*

agent learning opportunities reside in the driver agents. In the current implementation, driver agents must estimate the arrival time at the intersection, the arrival velocity, the arrival lane ... without communication nor coordination with the other driver agents; each agent makes its request on the basis of its actual velocity, and, if the request is rejected, the driver slows down and tries again. On the other hand, by letting the agents form teams and coordinate their actions, we provide them with more information that they use to make decisions.

2.1 Intersection model

We started from the model of intersection management proposed by Dresner and Stone [2, 3]. A driver agent cannot cross the intersection without a confirmed reservation. For this reason, we assume that when it reaches a minimum distance to the intersection, it is obliged to start making reservation requests. If a driver agents arrives at the intersection without a confirmed reservation, it stops and keeps trying to find a time slot when it may pass. The fullfilment or not of this norm is not guaranteed by any authority. Simply it is assumed that the driver agents are rational and that it is not convenient for them to risk a crash by crossing the intersection without a confirmed reservation.

If a request is confirmed by the intersection manager, the driver agent continuosly checks if it can meet the request conditions. If it realizes that is not able to meet them, due to the traffic conditions, it cancels its reservation by sending a message to the intersection manager, and prepares a new reservation request. Again, doing so is in its interest because a driver agent can only have one reservation, and a confirmed reservation that is not possible to meet is useless to hold.

2.2 Agent model

Our environment imposes a series of constraints on the agent behaviour. We assume that each driver agent has a preferred velocity which tries to keep along the entire journey. We also assume that when a vehicle appears in a lane of a road segment, it cannot change it during the approach to the intersection[1]. In this way, if a front vehicle proceeds at a lower velocity, the following vehicle is obliged to slow down. Furthermore, as demonstrated in [3], it is not convenient that the driver agents could turn from any lane, so in our model turning right (respectively left) is only possible from the rightmost (respectively leftmost) lane of a road segment.

The actions that a driver agent can autonomously take are related to the velocity at which it crosses the intersection. In particular, an agent could set its velocity to a value in the (discretized) interval $[1, preferredVelocity]$.

So, for the generic driver agent a_i, the variable x_i that defines its action is

$$x_i = \langle vehicleID, direction, lane, \\ turn, arrivalTimeAtIntersection, \\ arrivalVelocityAtIntersection \rangle$$

The field $arrivalTimeAtIntersection$ is implicitly set by the specific $arrivalVelocityAtIntersection$, while the fields $vehicleID$, $direction$, $lane$ and $turn$ are constant parameters.

[1]This is a feature that we plan to remove from the model in the future.

We assume that there are no misunderstandings regarding the ontology that describes the geometric configuration of the intersection, e.g. the lane 3 along the *North* direction corresponds to the same physical lane for every vehicle.

3. LEARNING TO COORDINATE

3.1 Global objective

To improve the efficiency of the intersection, we take the perspective of a system designer, whose goal is minimizing the travel time of the vehicles. The travel time for the generic driver agent a_i depends not only on its velocity while crossing the intersection, but also on the conflicts that may occur among different competing requests.

Let $\mathcal{C}$ be a set of driver agents, $\mathcal{C} = \{a_1, a_2, \ldots, a_n\}$. Each agent can take an action of the form defined in section 2.2. So, the vector $x = \langle x_1, x_2, \ldots, x_n \rangle$ defines the joint action of this set of agents. A possible function [2] that rates "how good" is a joint action, from the system designer perspective, is

$$G(x) = (1 + P(x)) \cdot D(x) \quad (1)$$

where $P(x)$ is the number of collisions resulting from the full joint action x, and $D(x)$ is the time spent by the agents to cross the intersection. We remark that a generic joint action x contains all the necessary information to simulate the agent journeys through the intersection, in the same way it is done by the intersection manager, so that is is also possible to calculate the number of conflicts among them as well as the travel time.

3.2 Agent private utility

The multiagent learning challenge here is making agents learn to act in an environment that is not merely a black-box that produces a reward for every action taken by the agent, but it is actually composed of other learning agents, i.e. the reward an agent receives for its actions depends also on the actions of other agents. So there is a strict relation between the private utility function of a single agent and the global objective of the system.

A recent advance in this direction is that proposed by the COllective INtelligence (COIN) [7, 8, 9] framework. The aim of COIN is studying the properties that a utility function of a learning agent situated in a multiagent environment must meet. COIN introduced the concepts of *factoredness* and *learnability* of an agent private utility function. A private utility function g_i is meant to be *factored* if it is aligned with the global utility G, i.e. if the private utility increases, the global utility does the same. Furthermore, it has to be easily *learnable*, i.e. it must enable the agent to distinguish its contribution to the global utility from that of the other agents. For example, the *Team Games Utility*, $TGU_i(x) = G(x)$, is trivially *aligned*, but is poorly *learnable*. If for example agent a_i takes an action that actually improves the global utility, while all the other agents take actions that worsen the global utility, agent a_i wrongly believes that its action was bad.

[2]It is possible to formulate other objective functions that take in consideration different relationship between collisions and time, as well as including other aspects, such as congestion or lane changes.

Better results have been obtained [9] with the *Difference Utility* (DU), defined as follows:

$$DU_i(x) = G(x) - G(CL_i(x)) \quad (2)$$

where x if the joint action of the collective, $G(x)$ is the global utility derived from such joint action, and $G(CL_i(x))$ is the "virtual" joint action formed by replacing with a constant factor c all the components of x affected by agent a_i. If this constant is $\overrightarrow{0}$, i.e. the null action, the DU is equivalent to the global utility minus the global utility that would have arisen if the agent a_i had been removed from the system.

Such an utility function is *aligned* with the global utility; in fact, since the second term in equation 2 does not depend on the action taken by agent a_i, any action that improves $DU_i(x)$ also improves the global utility $G(x)$. Furthermore, it is more *learnable* than TGU because, by removing agent a_i from the dynamics of the system, it provides a clearer signal to agent a_i.

In the case of intersection management, the driver agent computes the $DU_i(x)$ as follows:

$$DU_i(x) = (1 + P(x)) \cdot D(x) - (1 + P(CL_i(x))) \cdot D(CL_i(x)))$$

where $CL_i(x) = \langle x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n \rangle$

3.3 Probability Collectives (PC)

Once the agents in a collective have been provided with "well-designed" private utility functions, many methods are available for supporting the agent decision making, such as reinforcement learning [5]. In this paper we draw upon a novel method called Probability Collectives (PC) [10], which has been developed within the COIN [7, 8, 9] framework, for the agent decision making. PC replaces the search for the most valuable action across the space of *actions* with the search across the space of *probability distributions* over those actions. In other words, PC aims at learning the agent decision strategies that maximize the global objective.

Formally, let $\mathcal{C} = \{a_1, a_2, \ldots, a_n\}$ be a collective of n agents. Each agent a_i can take an action by setting its action variable x_i, which can take on finite number of values from the set X_i. So these $|X_i|$ possible values constitute the action space of the agent a_i. The variable of the joint set of n agents describing the collective action is $x = \{x_1, x_2, \ldots, x_n\} \in X$, with $X = X_1 \times X_2 \times ... \times X_n$.

Given that each agent has a probability distribution (i.e. mixed strategy in game theory sense) over its possible actions, $q_i(x_i)$, the goal of PC is to induce a product distribution $q = \prod_i q_i(x_i)$ that is highly peaked around the x that maximize the objective function of the problem, and then obtaining the optimized solution x by sampling q.

The main result of PC is that the best estimation of the distribution q_i that generates the highest expected utility values is the minimizer[3] of the Maxent Lagrangian (one for each agent):

$$\mathcal{L}_i(q_i) = E_q[g_i(x)] - T \cdot S(q_i) \quad (3)$$

where q_i is the agent probability distribution over the agent a_i actions, x_i; $g_i(x)$ is the agent a_i private utility function (e.g. the *Difference Utility* defined in equation 2), which maps a joint action into the real numbers; the term $E_q[g_i(x)]$ is the expected utility value for agent a_i, subjected to its action and the actions of all the agents other than a_i; $S(q_i)$ is the Shannon entropy associated with the distribution q_i, $S(q_i) = -\sum_{x_i} q_i(x_i) \ln[q_i(x_i)]$; T is an inverse Lagrangian multiplier, which can be treated as a "temperature": high temperature implies high uncertainty, i.e. *exploration*, while low temperature implies low uncertainty, i.e *exploitation*.

Since the Maxent Lagrangian is a real valued function of a real valued vector, it is possible to use gradient descent or Newton methods for its minimization. Using Newton methods, the following update rule is obtained:

$$q_i^{t+1} = q_i^t - \alpha q_i^t \times \{\frac{E_q[g_i|x_i] - E_q[g_i]}{T} + S(q_i^t) + \ln[q_i^t]\} \quad (4)$$

where $E_q[g_i]$ is the expected utility, $E_q[g_i|x_i]$ is the expected utility associated with each of the agent a_i's possible actions, and α is the update step. Equation 4 shows how the agents should modify their distributions in order to jointly implement a step in the steepest descent direction of the Maxent Lagrangian.

Since at any time step t, an agent might not know the other agents' distributions, in this case it wouldn't be able to evaluate any expected value of g_i, because they depend on the full probability distribution q. Those expectation values can be estimated by repeated Monte Carlo sampling of the distribution q to produce a set of $(x; g_i(x))$ pairs. Each agent a_i then uses these pairs to estimate the values $E_q[g_i|x_i]$, for example by uniform averaging of the g_i values in the samples associated with each possible action.

3.4 PC for intersection management

PC is a broad framework for the analysis, control and optimization of distributed systems that offers new approaches to problems. Nevertheless, in order to be actually instantiated in a particular domain, several design decisions must be made.

Since the entire framework is based on the Monte Carlo-based estimation of the product distribution that maximizes the global objective, it is necessary to have a communication structure that enables to build the set of sampled joint actions. For example in [6] such a set is constructed using a token-ring message passing architecture. In this work, we opted for letting the agents asynchronously request the other agents in the collective to sample their distributions. Then each agent constructs locally its set of sampled joint actions and uses them to update its distribution with equation 4. We assume that the agents truthfully sample their distributions without manipulation, even if investigating how an agent can exploit the coordination mechanism for its purposes deserves a further analysis.

Another design decision is the setting of the initial temperature T and the initial probability distribution q_i. The initial temperature usually depends on the particular domain, because its order of magnitude is strictly related with the expected utility values (see equation 4). In our experiments we set the initial temperature to 1. On the other hand, the initial probability distribution q_i is usually initialized with the maximum entropy distribution, i.e. the uniform distribution over the action space X_i. In this way we don't make any assumptions about the desiderability of a particular action and all the actions are equiprobable.

[3]Without loss of generality, the global utility function is considered as a "cost" to be minimized, by simply flipping the sign of the utility value

Usually, the Lagrangian minimization proceeds as follows: for a given temperature T, the agents jointly implement a step in the steepest descent direction of the Maxent Lagrangian using equation 4. Then the temperature is slightly reduced, and the process continues, until a minimum temperature is reached. The annealing schedule we implemented was geometrically reducing the temperature T as long as a driver agent approaches the point after which it is obliged to send a request to the intersection manager, as described in section 2.1. When a driver agent arrives at that point, it evaluates the action with the highest probability, sets its velocity accordingly (see section 2.2) and makes a reservation request with the given velocity.

Algorithm 1 sketches the algorithmic structure of an agent program that implements PC for the intersection management problem. The algorithm starts initializing the temperature T and the probability distribution q_i (line 01 and 02). The main loop controls the annealing schedule of the temperature T (line 09), until the driver agents reaches the minimum distance to the intersection (line 03).

The minimization of $\mathcal{L}_i$ for a fixed temperature is accomplished by repeatedly determining all the conditional expected values $E_q[g_i|x_i]$ (line 06) and then using these values to update the distribution (line 07). Such values are obtained by requesting samples to the agents in the collective (line 04) and storing them when they are received (line 10), in order to have an estimation of the entire distribution q.

At the end of the algorithm, agent a_i selects its "best" action by sampling the distribution q_i or directly selecting the action with the highest probability, and then store the request that will be sent to the intersection manager.

From this point on, the agent starts to behave like in the reservation-based scenario described in section 2 (for more details, see [2, 3]). It sends reservation requests to the intersection manager, until it receives a confirmation or a refuse message. In the first case, the driver agent stores the reservation details and tries to meet them. Otherwise, it decreases its velocity and makes another request in the next step.

A driver agent is not allowed to cross the intersection with an out-of-date reservation or without reservation at all. A confirmed reservation goes out-of-date if the agent is not able to meet its details, i.e. the agent cannot be at the intersection at the time specified in the reservation, due to the traffic conditions. In this case, the driver agent can cancel the reservation with the intersection manager and make a new one, whose constraints it is able to meet.

If an agent arrives at the intersection with no confirmed and valid reservation, it is obliged to stop at the intersection. At this point, the driver agent is only allowed to propose reservations for the time slots in the near future.

4. EXPERIMENTAL RESULTS

In this section we present the results of the experiments conducted with the simulator of a *4-ways-3-lanes* intersection (see figure 1). The metric we used to evaluate the efficiency of the intersection was the average travel time of a set of vehicles. During the simulation, a total of 100 vehicles are generated using a Poisson distribution $f(k, \lambda) = \lambda^k e^{-\lambda}/k!$, where λ is the number of expected occurences (i.e. vehicles) in a given interval. In all the experiments, the λ parameter is kept fixed, while we progressively reduce the interval, simulating in this way different (increasing) traffic densities

Algorithm 1 PC for intersection management

```
01: T ← 1
02: q_i ← uniformDistribution
03: while minimum distance not reached do
04:    requestMCsamples
05:    if m not empty then
06:       ce ← evalConditionalExpectations(m)
07:       q_i ← updateQ(ce)
08:    end if
09:    T ← updateT
10:    m ← storeIncomingMCSamples
11: end while
12:
13: x̂_i ← mostProbableAction
14: velocity ← x̂_i.arrivalVelocityAtIntersection
15: store request R = ⟨vehicleID, direction, lane,
    turn, arrivalTimeAtIntersection, velocity⟩
```

Each spawned vehicle has a preferred velocity, whose value is generated randomly using a gaussian distribution with $\mu = 3$ and $\sigma^2 = 1$, and the maximum allowed velocity was set to 10.

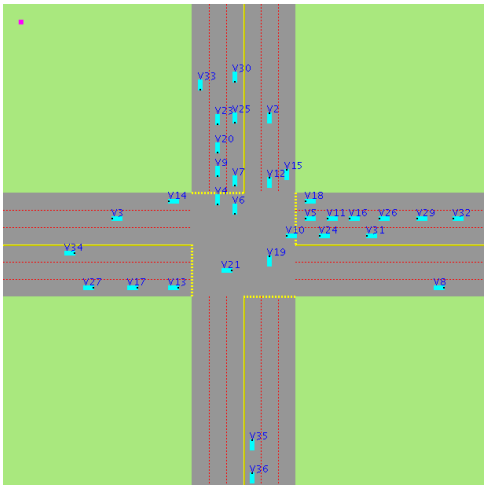


Figure 1: Simulator snapshot

One challenge in the implementation of the coordination mechanism was coping with the extreme dynamic and asynchronous nature of the system, as well as with the constraints imposed by the real-time.

Furthermore, while in multiagent reinforcement learning it is assumed that in every learning episode the set of agents remains the same, in this case this assumption does not hold, because the set of learning agents is created dynamically. Once a driver agent appears in the managed area, its ID is stored by the road infrastructure. Then the road infrastrucuture periodically communicates the set of collected IDs to the agents, in order to create collective of coordinating agents.

In figure 2 we plotted the average travel time of 20 experiments for two different configurations. In one configuration, each driver agent communicates exclusively with the intersection manager by making reservation requests solely on the basis of its knowledge; in the other configuration, the

driver agents implement the coordination mechanism before starting making reservation requests.

If the traffic density is low, the average travel time of the two configurations is approximatively the same. This is reasonable, since with low traffic density few reservation requests are rejected, so no previous coordination is needed. Similarly, with high traffic density the average travel time tends to be the same for the two configurations. Again this is reasonable, because the intersection tends to be saturated by vehicles stopped at the intersection, waiting for its reservation request to be confirmed.

On the other hand, it is noticeable a traffic density interval where the coordination between driver agents benefits the intersection system by reducing the average travel time up to approximatively the 7% (see figure 3).

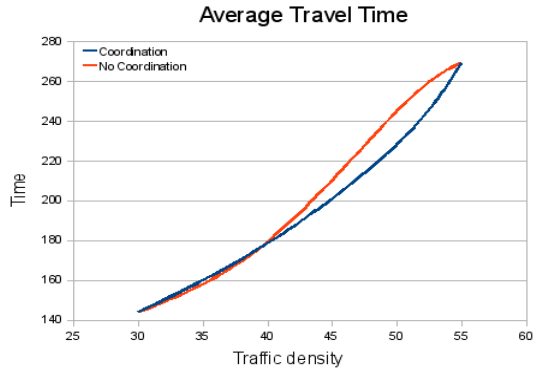


Figure 2: Average travel time

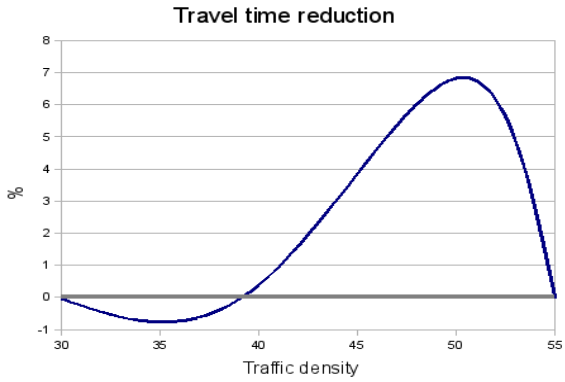


Figure 3: Average travel time reduction

Notwithstanding, there is space for further possible improvements of the agents learning capabilities. Firstly, the agent action space to act in the environment is quite reduced, since it can only set the velocity at which it intends to cross the intersection. For example, if there is a confirmed reservation of a very slow vehicle, which occupies the intersection for many time slots, it is reasonable to think that there is no way for an approaching agent to make a request that will not be refused, no matter the velocity it proposes. So, a possible improvement could derive from giving the agents the possibility of changing its lane.

Furthermore, with the agent model described in section 3, a collective of agent searches the product distribution q to maximize a global utility function $G(x)$. This is a function of the joint action x, and do not take in consideration external factors (i.e. noise). In the domain of the intersection management, for a given $x = \langle x_1, x_2, \ldots, x_n \rangle$, an agent is only able to evaluate the number of conflicts that occurs among the x_is and their travel times, by simulating the journey of each agent a_i through the intersection. If for example the intersection is saturated due to a crash, or it has been reserved by very slow vehicles, the collective is not able to react to these events and adjust its collective behaviour, since it does not have such information.

A way to circumvent this problem is modifying the structure of the global utility as a function of a 2-players game between the collective and the external world. At each time step, the collective sets its joint action x, while the world plays y. Such a vector y contains any external information not directly under control of the collective. Then the global objective $G(z)$ is calculated, as a function of the full vector $z = \langle x, y \rangle$. In the domain of the intersection management, the vector y could contain information about the traffic conditions in front of each driver agent, or about the confirmed reservations that the intersection manager has in its database. Nevertheless, such modification could have side effects that may worsen the learning rather than improving it.

Another issue that it is worth mentioning is that the multiagent learning performed by the driver agents comes as a sort of coordination on-the-fly: an agent does not learn from the behaviours of the other agents that it observes, rather such information is explicitly provided by exchanging samples of the agents' distributions. This setting speeds up the learning, although it has an associated cost for the communication overhead. Basically the key point for an agent is evaluating the expected utility of its actions, $E_q[g_i|x_i]$ (see equation 4). Within this formalization, such a value is obtained using an estimation of the joint distribution q, by using the samples provided by all the agents.

If we remove the communication between agents, the only way for an agent to evaluate the expected utility $E_q[g_i|x_i]$ is actually executing different actions in several episodes (i.e. crossing the intersection several times at different velocities), then using the utility values of the different actions it has executed in these episodes to compute $E_q[g_i|x_i]$ and adapt its mixed strategy q_i. Within this formalization, no communication is needed, although the learning process will take much more time.

5. CONCLUSIONS

This paper showed that the intersection management problem offers many opportunities for multiagent learning [4]. In particular, we started from the COIN framework for the definition of agent private utilities, and we applied Probability Collectives to make the agents learn to coordinate their action. The preliminary experiments showed some improvements of the intersection efficiency, with a reduction of the average travel time for a given traffic density interval.

Future works includes evaluating the model under different metrics (e.g. delay, congestion, number of refused reservation...), considering different private utility functions and global objectives, as well as modifying the model as described in section 4. More generally, the road traffic management scenario is open to a plethora of interesting research lines, from the study of "cooperative vs competitive" agent

behaviour, to the impact of "malicious" agents that try to exploit the coordination mechanism.

6. REFERENCES

[1] F. Klügl, A. Bazzan, and S. Ossowski. Applications of Agent Technology in Traffic and Transportation. Birkhäuser, 2005.

[2] K. Dresner, and P. Stone. Multiagent traffic management: A reservation-based intersection control mechanism. *Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 530–537. ACM Press, 2004.

[3] K. Dresner, and P. Stone. Multiagent traffic management: an improved intersection control mechanism. *Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 471–477. ACM Press, 2005.

[4] K. Dresner, and P. Stone. Multiagent Traffic Management: Opportunities for Multiagent Learning. Lecture Notes in Computer Science, pages 129–138, Volume 3898, Springer-Verlag, 2006.

[5] R.S. Sutton, and A.G. Barto. Reinforcement learning: An Introduction. MIT Press, 1998.

[6] A. Waldock, and D. Nicholson. Cooperative Decentralised Data Fusion Using Probability Collectives First International Workshop on Agent Technology for Sensor Networks (ATSN-07), held at AAMAS 2007

[7] Wolpert, D., and Tumer, K.: *An introduction to COllective INtelligence.* Technical Report NASA-ARC-IC-99-63, NASA Ames Research Center, 1999

[8] Wolpert, D., Wheeler, K. R., and Tumer, K.: *General principles of learning-based multi-agent systems.* In O. Etzioni, J. P. Müller, and J. M. Bradshaw, editors, Proceedings of the Third Annual Conference on Autonomous Agents (AGENTS-99), pp 77–83. ACM Press, New York, 1999

[9] Wolpert, D., and Tumer, K.: *Optimal payoff functions for members of collectives.* Advances in Complex Systems, 2001

[10] D. Wolpert. Information theory - the bridge connecting bounded rational game theory and statistical physics. ArXiv Condensed Matter e-prints, 2004.

Mitigating Catastrophic Failure at Intersections of Autonomous Vehicles

Kurt Dresner and Peter Stone
University of Texas at Austin
Department of Computer Sciences
Austin, TX 78712 USA
{kdresner, pstone}@cs.utexas.edu

ABSTRACT

Fully autonomous vehicles promise enormous gains in safety, efficiency, and economy for transportation. However, before such gains can be realized, a plethora of safety and reliability concerns must be addressed. In previous work, we have introduced a system for managing autonomous vehicles at intersections that is capable of handling more vehicles and causing fewer delays than modern-day mechanisms such as traffic lights and stop signs [3]. While the system is safe under normal operating conditions, we have not discussed the possibility or implications of unforeseen mechanical failures. Because the system orchestrates such precarious "close calls" the tolerance for such errors is very low. In this paper, we make four main contributions. First, we introduce safety features of the system designed to deal with these types of failures. Second, we perform a basic failure mode analysis, demonstrating that without these features, the system is unsuitable for deployment due to a propensity for catastrophic failure modes. Third, we give extensive empirical evidence suggesting that not only is this method effective, but that it is so even when normal communications are disrupted. Finally, we provide an analysis of the data indicating that despite the apparent potential for disastrous accidents, autonomous intersection management is likely to improve driver safety considerably.

Keywords

Autonomous Vehicles, Multiagent Systems, Intelligent Transportation Systems

1. INTRODUCTION

Fully autonomous vehicles promise enormous gains in safety, efficiency, and economy for transportation. By taking the responsibility of driving away from humans, autonomous vehicles will completely eliminate driver error from the complicated equation of automobile traffic. By some estimates, driver error can be blamed for as much as 96% of all automobile accidents [11]. Thus, even if each accident were substantially worse, overall autonomous vehicles would represent an improvement in safety over the current situation. With automobile collisions costing the U.S. economy over $230 billion annually, any significant decrease would be a major triumph for artificial intelligence [6].

Traffic intersections are a compelling problem for multiagent systems. Often a source of great frustration for drivers, intersections represent both a sensitive point of failure as well as a major bottleneck in automobile travel. While fully autonomous open-road driving was demonstrated over ten years ago, events such as the DARPA Urban Challenge prove that city driving, including intersections, still pose substantial difficulty to AI and intelligent transportation systems (ITS) researchers.

We have proposed a reservation-based multiagent framework for managing vehicles at intersections, including both human-driven vehicles and fully autonomous vehicles [3]. Instead of using traditional traffic lights, the mechanism allows autonomous vehicles to "call ahead" to arbiter agents stationed at intersections and reserve space-time in the intersection. When a vehicle obtains a reservation, it can proceed through the intersection without stopping. By coordinating the actions of many autonomous vehicles, the system dramatically decreases time spent stopped or slowing down due to intersections. Because the system heavily exploits the precision sensory and control capabilities of computerized drivers, it offers dramatic improvements in efficiency. However, this increased efficiency is quite precarious. The system orchestrates what can only be described as "extremely close calls", with vehicles missing each other by the smallest (albeit adjustable) margins[1]. Figure 1 contains a screenshot depicting a particularly busy intersection.

While the system is safe in the face of communication failures, we have not addressed the possibility or effects of mechanical failures or unlikely "freak" accidents. In a world without vehicle malfunctions, this would be little cause for concern. However, one can easily imagine an otherwise ordinary problem, such as a flat tire or a slippery patch of road, quickly becoming a nightmare.

Even though the vast majority of automobile accidents can be blamed on driver error (or in some cases, the limitations of human drivers), if individual incidents are a hundred times more deadly, no reasonably achievable reduction in incident frequency will effect an overall improvement. How-

[1]Our project website includes videos of simulations that demonstrate this phenomenon: `http://www.cs.utexas.edu/~kdresner/aim/`

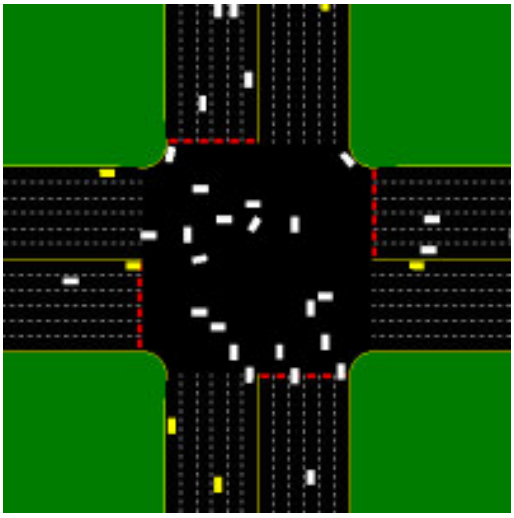


Figure 1: A screenshot showing a busy intersection with a lot of "close calls."

ever, if in the rare event of an accident, the total damage can be kept under control—perhaps at most a few times as many as normal—then, as a whole, riding in automobiles will be a safer experience than it is today.

In this paper, we make four main contributions. First, we introduce safety features of the system designed to deal with these types of failures. Second, we perform a basic failure mode analysis, demonstrating that without these features, the system is unsuitable for deployment due to a propensity for catastrophic failure modes. Third, we give extensive empirical evidence suggesting that not only is this method effective, but that it is robust in the face of poor communications. Finally, we provide an analysis of the data indicating that despite the apparent potential for disastrous accidents, autonomous intersection management is likely to improve driver safety considerably.

The remainder of this paper is organized as follows. Section 2 briefly summarizes our reservation system and earlier results. In Section 3, we elucidate the safety mechanisms in the system to deal with potentially catastrophic mechanical failures and argue for their necessity. Section 4 presents empirical evidence evaluating our addition to the system. We discuss these results and their implications in Section 5 and conclude in Section 6.

2. BACKGROUND INFORMATION

Our multiagent intersection control mechanism comprises the interactions of two classes of agents: *intersection managers* and *driver agents* [3]. Driver agents are computer programs that pilot vehicles, while intersection managers are specialized arbiter agents stationed at each intersection that control access to that intersection. In order to cross the intersection, driver agents must first obtain approval from the intersection manager.

2.1 Assumptions

We make several important assumptions about the capabilities of intersection managers and driver agents. We assume that intersections can be equipped with a wireless communication device with enough strength and bandwidth to communicate with hundreds of driver agents simultaneously. We also assume that the intersection manager has access to sufficient computational resources to process all the messages from these driver agents and respond to them quickly. Because our simulator can execute all the driver agent and intersection manager algorithms in real time, in one process on a desktop computer, we believe this is a realistic assumption. Finally, we assume that vehicles can be similarly outfitted, both in terms of communication and computation, and that these vehicles have access to GPS navigation equipment, detailed electronic maps of their environs, short-wave radar and lidar systems, and any other sensing technology required for them to accurately and reliably determine their location and sense the objects and vehicles around them. These assumptions are all reasonable given current technology.

2.2 Communication Protocol

A major part of the reservation mechanism is the communication protocol that governs all transmissions between agents [2]. In this protocol, when a vehicle approaches an intersection, the driver agent controlling that vehicle "calls ahead" to the intersection manager, requesting permission to cross. This request comes in the form of a REQUEST message. In addition to parameters describing the physical characteristics of the vehicle, such as its size and performance capabilities, a REQUEST message includes the direction the driver agent would like to leave the intersection, as well as estimates of its arrival time and arrival velocity. The intersection manager can then use this information, along with an *intersection control policy* to decide whether or not to grant the reservation. If it chooses to grant the reservation, it responds with a CONFIRM message containing some restrictions the vehicle must obey in order to cross safely. According to the protocol, the intersection manager can also use the restrictions in the CONFIRM message to make a counter-offer. The driver agents acceptance of the confirmation is implicit; as soon as the intersection transmits the CONFIRM message, the vehicle "has" the reservation described therein. If the intersection manager decides not to grant the reservation, it responds with a REJECT message, which can optionally include a reason for the rejection. According to the rules of the protocol, no vehicle may enter the intersection under any circumstances without a reservation.

Once a vehicle has a reservation, its safety is guaranteed in the intersection, provided it crosses the intersection in accordance with that reservation. If the driver agent concludes at any time that it cannot meet the reservation, it can send a CANCEL message to the intersection manager, at which point the vehicle is no longer considered to have a reservation. Additionally, vehicles can attempt to change their reservations, with a CHANGE-REQUEST message. This message is the same as the REQUEST message, except that if the intersection manager responds with a REJECT message, the vehicle maintains its original reservation.

2.3 First Come, First Served

Our framework includes several intersection control policies, including some that emulate current-day mechanisms like stop signs and traffic lights. However, most of the extremely efficient policies are based around the "first come, first served" (FCFS) algorithm. This algorithm divides the intersection into an $n \times n$ grid of *reservation tiles*, where the

parameter n is called the *granularity* of the policy. When it receives a REQUEST, an FCFS policy simulates the trajectory of the vehicle across the intersection using the parameters in the message. Throughout the simulation, the policy determines which of the reservation tiles are occupied by the simulated vehicle, and whether or not any of them are already reserved by another vehicle while the simulated vehicle would occupy them. If no such conflicts are detected throughout the simulation, the appropriate tiles are reserved for the required times, the policy creates the reservation, and the intersection manager sends the relevant information to the requesting agent in a CONFIRM message. Otherwise, the driver agent receives a REJECT message.

While FCFS policies are limited to use by autonomous vehicles only, we have also created a policy called FCFS-LIGHT, which can accommodate human drivers [4]. Briefly, an FCFS-LIGHT policy is similar to standard FCFS, except that it incorporates a *light model*, which both controls a set of physical lights at the intersection, and provides information to the policy about the state of those lights. Areas of the intersection that correspond to conventional paths through the intersection are blocked off from use by autonomous vehicles whenever the light controlling access to that path is green, yellow, or recently red. This creates a *de facto* reservation for any human that might be crossing the intersection based on that green light. While this does allow a sort of "backwards compatibility", it must be noted that by nature, FCFS-LIGHT policies tend to be much less efficient than standard FCFS policies.

2.4 Safety Guarantees

While this paper focuses on some of the ways our mechanism can react to gross mechanical failures, we must point out the ways in which it compensates for smaller, more common errors. As long as all vehicles follow the protocol and all the technology works as expected, no two vehicles should be able to occupy the same space in the intersection at the same time. Only one vehicle can reserve any particular reservation tile at one time, and vehicles can only cross the intersection in accordance with their reservations. Unfortunately, even under normal operating conditions, this is not quite enough. Communication failures including dropped and corrupted messages, as well as small errors in the vehicle's sensors and actuators could all cause problems. The mechanism is designed to be robust against all of these. The driver agent's implicit acceptance of reservation confirmations means that the worst possible consequence of a dropped or corrupted message is additional delay, and not a collision. Buffering in the intersection control policies adds protection against small sensor errors by reserving space for vehicles as if they were larger than they actually are.

3. ADDING A SAFETY NET

A collision in purely autonomous traffic can have any number of causes, including software errors in the driver agent, a physical malfunction in the vehicle, or even meteorological phenomena. In modern-day traffic, such factors are largely ignored for two reasons. First, the exclusively human-populated system, with its generous margins for error, is not as sensitive to small or moderate aberrations. Second, none of these causes are significant with respect to driver error as causes of accidents. In fact, according to a study from the 1980's, vehicle and road issues alone were responsible for fewer than 5% of accidents [11]. However, in the future of infallible autonomous driver agents, it is exactly these issues which will be the prevalent causes of automobile collisions. The safety buffers in our mechanism are adjustable—given some maximum allowable error in vehicle positioning, the buffers can be extended to handle that error—but no reasonable adjustment can account for gross mechanical malfunction like a blowout or failed brakes. Because these types of issues are infrequent, we believe the safety of the intersection control mechanism will be acceptable even if individual occurrences are slightly worse than accidents today. As we will show in Section 4, without the safety measures presented in this section, the system is prone to spectacular failure modes, sometimes involving dozens of vehicles.

3.1 Assumptions

In Section 4, we will show how our modification can reduce the average number of vehicles involved in a crash from dozens to one or two. In order to make this improvement, we must make a few assumptions above and beyond those originally made by Dresner and Stone [3].

3.1.1 Detecting the Problem

First, we assume that the intersection manager is able to detect when something has gone wrong. While this is certainly a non-trivial assumption, it is necessary for any reasonable solution. Simply put, the intersection manager cannot react to something it cannot detect. There are two basic ways by which the intersection manager could detect that a vehicle has encountered some sort of problem: the vehicle can inform the intersection manager, or the intersection manager can detect the vehicle directly. For instance, in the event of a collision, a device similar to that which triggers an airbag can send a signal to the intersection manager. Devices such as this already exist in aircraft to emit distress signals and locator beacons in the event of a crash. The intersection manager itself might notice a less severe problem, such as a vehicle that is not where it is supposed to be, using cameras or sensors at the intersection. However, this method of detection is likely to be much slower to react to a problem. Each has advantages and disadvantages, and a combination of the two would most likely be the safest. The specifics of the implementation are beyond the scope of this paper. What is important is that whenever a vehicle violates its reservation in any way, the intersection manager should become aware as soon as possible. Because our simulations only deal with collisions, we assume that the colliding vehicle sends a signal and the intersection manager becomes aware of the situation immediately.

The communication protocol also includes a DONE message that vehicles transmit when they complete their reservations. One way to reliably sense when a vehicle is in distress would be to notice a missing DONE message. This does have two drawbacks however. First, the DONE message is optional, mainly because there is no incentive for the driver agent to transmit it. Second, the intersection manager may not be able to notice the missing message until some time after the incident has occurred. We hope to investigate this alternative in future work.

3.1.2 Informing the Other Vehicles

The second assumption we make is that there exists a

way for the intersection manager to broadcast the fact that something is wrong to the vehicles. We already assume that the intersection manager can communicate with the vehicles, but this new assumption is a bit different. Under normal operating conditions, individual messages each containing multiple pieces of information are transmitted between agents [2]. In case of an emergency, however, the intersection manager needs only to communicate one bit of information: whether or not something is wrong. This can come in the form of a coded signal (to prevent fakes) repeated continuously on a specific frequency. As with the previous assumption, the specifics of the implementation are not relevant to this work. We assume that the intersection manager can transmit such a signal, and that the vehicles receive it. As we will show in Section 4, even without assuming the vehicles receive the signal, it is still possible to drastically improve the safety properties of the mechanism in the face of mechanical failures.

3.2 Incident Mitigation

When a vehicle deviates significantly from its planned course through the intersection, resulting in physical harm to the vehicle or its presumed occupants, we refer to the situation as an *incident*. Once an incident has occurred, the first priority is to ensure the safety of all persons and vehicles nearby. Because we expect these incidents to be very infrequent occurrences, re-establishing normal operation of the intersection is a lower priority and the optimization of that process is left to future work.

3.2.1 Intersection Manager Response

As soon as our intersection manager is notified of an incident, it ceases granting reservations. All subsequent received requests are rejected without consideration. Due to the nature of the protocol, reservations cannot be revoked by the intersection manager. However, given our assumptions, in such a dire situation the intersection manager can signal to the vehicles that an incident has occurred. This signal is repeatedly broadcast, and is not part of the reservation protocol. Ideally, all vehicles would receive the signal and stop immediately, including those holding approved reservations.

This concept extends naturally to policies that can accommodate humans, such as FCFS-Light. Analogous to refusing further reservation requests, upon detecting an incident, the intersection manager immediately turns all lights red. In a real-world implementation, a more conspicuous visual cue could be provided, but semantically it is only important that the intersection inform the human drivers that they may not enter.

3.2.2 Vehicle Response

The driver agent also has a role to play once an incident has taken place. Normally, when a vehicle is approaching the intersection, it ignores any vehicles sensed in the intersection—what might otherwise appear to be an imminent collision on the open road is almost certainly a precisely coordinated "near-miss" in the intersection. Once the driver agent has received the emergency signal from the intersection manager, it disables this behavior. Thus, if something is wrong, and the vehicle is in the intersection, the driver agent will not blindly drive into another vehicle, if it can help it. If the vehicle is not in the intersection, it will not enter, even if it has a reservation.

A first approach might be to make all driver agents that receive the signal immediately decelerate to a stop. However, this is actually less safe. If all vehicles that receive the signal come to a stop, vehicles that would otherwise have cleared the intersection without colliding may find themselves stuck in the intersection—another object for other vehicles to run into. This is especially true if the crashed vehicle is off on the very edge of the intersection where it is unlikely to be hit. Trying to stop all the other vehicles in the intersection would make the situation much worse.

If a driver agent does detect an impending collision, however, it is allowed to take evasive actions or apply the brakes. Since this is a true multiagent system with self-interested agents, we cannot prevent the driver agents from doing so. Thus, our driver agent only brakes if it believes a collision is imminent.

4. EXPERIMENTAL RESULTS

In this section, we present empirical evaluations of our claims using a custom simulator described in our earlier work [3, 4].

4.1 Experimental Setup

With the great efficiency of the reservation-based system comes an extreme sensitivity to error. While buffering can protect against the more minute discrepancies, it cannot hope to cover gross mechanical malfunctions. To determine just how much of an effect such a malfunction would have, we created a simulation in which individual vehicles could be "crashed", causing them to immediately stop and remain stopped. Whenever a vehicle that is not crashed comes into contact with one that is, it becomes crashed as well. While this does not model the specifics of individual impacts, it does allow us to estimate how a malfunction might lead to collisions.

In order to ensure that we included malfunctions in all different parts of the intersection, we triggered each incident by choosing a random (x, y) coordinate pair inside the intersection, and crashing the first vehicle to cross either the x or y coordinate. This is akin to creating two infinitesimally thin walls, one horizontal and the other vertical, that intersect at (x, y). Figure 2 provides a visual depiction of this process.

After initiating an incident, we ran the simulator for an additional 60 seconds, recording any additional collisions and when they occurred. Using this information, we constructed a *crash log*, which is essentially a histogram of crashed vehicles. For each step of the remaining simulation, the crash log indicates how many vehicles were crashed by that step. By averaging over many such crash logs for each configuration, we were able to construct an "average" crash log, which gives a picture of what a typical incident would produce.

Because our system is compatible with humans, we felt it necessary to also include experiments with the human-compatible intersection control policies [4]. When a significant number of human drivers are present, the FCFS-Light policies do not offer much of a performance benefit over traditional traffic light systems. As such, we limited our experimentation to scenarios in which 5% of the vehicles are controlled by simulated human drivers. These human drivers have a very simple behavior that attempts to maintain a following distance that is proportional to the vehicle's velocity.

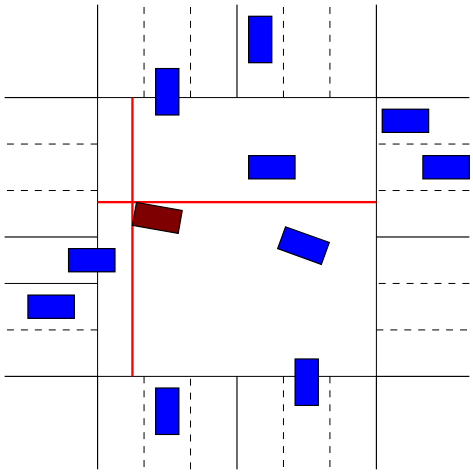


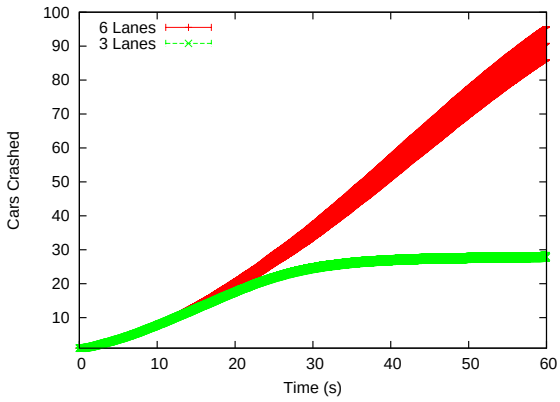
Figure 2: Triggering an incident in the intersection simulator. The dark vehicle turning left is crashed because it has crossed the randomly chosen x coordinate. If a different vehicle had crossed that x coordinate or the randomly chosen y coordinate earlier, it would be crashed instead.

With only 5% human drivers, an FCFS-Light policy can still create a lot of the precarious situations that are the focus of this investigation.

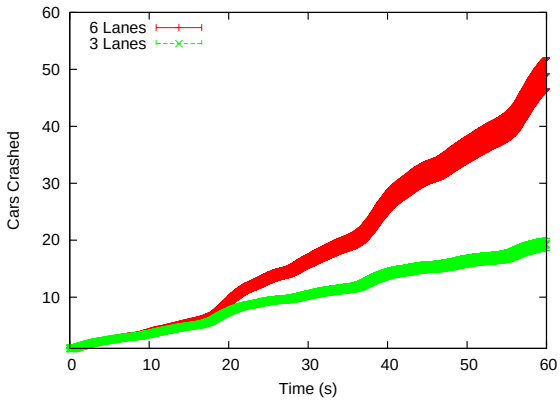
For these experiments, we ran our simulator with scenarios of 3, 4, 5, and 6 lanes in each of the four cardinal directions, although we will discuss results only for the 3- and 6-lane cases (other results were similar, but space is limited). Vehicles are spawned equally likely in all directions, and are generated via a Poisson process which is controlled by the probability that a vehicle will be generated at each step. Vehicles are generated with a set destination—15% of vehicles turn left, 15% turn right, and the remaining 70% go straight. The leftmost lane is always a left turn lane, while the right lane is always a right turn lane. Turning vehicles are always spawned in the correct lane, and non-turning vehicles are not spawned in the turn lanes. In scenarios involving only autonomous vehicles, we set the traffic level at an average of 1.667 vehicles per second per lane in each direction. This equates to 5 total vehicles per second for 3 lanes, and 10 total vehicles per second for 6 lanes. Scenarios with human-driven vehicles had one third the traffic of the fully autonomous scenarios—the intersection cannot be nearly as efficient with human drivers present. We chose these amounts of traffic as they are toward the high end of the spectrum of manageable traffic for the respective variants of the intersection manager. While we wanted traffic to be flowing smoothly, we also wanted the intersection to be full of vehicles to test situations that likely lead to the most destructive possible collisions.

4.2 How Bad Is It?

As we suspected, the average crash log without the safety measures is quite grisly. As explained in Section 3.2.2, driver agents must ignore their sensors while in the intersection, because many of the "close calls" would appear to be impending collisions. Without any way to react the situation going awry, vehicles careen into the intersection, piling up until the entire intersection is filled and crashed vehicles protrude from the intersection. Figure 3 shows that for both 6-lane cases—fully autonomous and 5% human drivers—the rate of collisions does not abate until over 70 vehicles have crashed. Even a full 60 seconds after the incident begins, vehicles are still colliding. In the 3-lane case, the intersection is much smaller, and thus it fills much more rapidly; by 50 seconds, the number of collided vehicles levels off.



(a) All autonomous



(b) 5% humans

Figure 3: Average crash logs (with 95% confidence interval) for 3- and 6-lane intersections, for the system with the safety measures from Section 3 disabled. In 3(a), the intersection manages only autonomous vehicles, while 3(b) includes 5% human drivers.

In both of the scenarios with human drivers, shown in Figure 3(b), the number of vehicles involved in the average incident is noticeably smaller. This outcome is likely the result of two factors. First and foremost, the FCFS-Light policy must make broad allowances to accommodate the human drivers, and thus overall is inherently less dangerous. The characteristic "close calls" from the standard FCFS policy are less common. Second, the simulated human driver agents do not drive "blindly" into the intersection—trusting to the intersection manager—the way the autonomous vehicles do. Also of note in Figure 3(b) is the visible periodicity of the light model portion of the policy. As paths open up for autonomous vehicles due to changes in the lights, they drive unwittingly into the growing mass of crashed cars.

4.3 Number of Collisions

	Fully Autonomous		5% Human	
	3 Lanes	6 Lanes	3 Lanes	6 Lanes
Safety Off	**27.9** ± 1.3	**90.9** ± 4.9	**19.3** ± 1.1	**49.3** ± 2.7
Recv.				
0%	**2.63** ± .13	**3.23** ± .16	**2.23** ± .10	**2.35** ± .13
20%	2.44 ± .13	3.15 ± .17	2.07 ± .10	2.29 ± .13
40%	2.28 ± .12	2.90 ± .16	1.91 ± .10	2.07 ± .12
60%	1.89 ± .10	2.69 ± .15	1.72 ± .09	1.98 ± .11
80%	1.71 ± .08	2.30 ± .13	1.46 ± .07	1.65 ± .09
100%	**1.36** ± .06	**1.77** ± .10	**1.22** ± .05	**1.50** ± .09

Table 1: Average number of vehicles involved in incidents for 3- and 6-lane intersections with Section 3's safety features disabled, and the system intact with various percentages of the vehicles receiving the emergency signal. Even without any vehicles receiving the emergency signal, our modification dramatically decreases the number of crashed vehicles. As more vehicles receive the emergency signal, the amount decreases further.

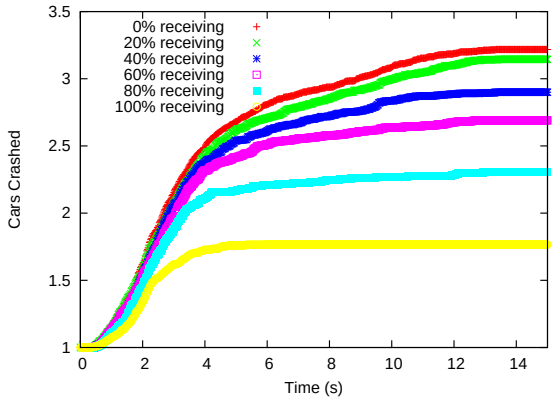
There are two main components to the safety mechanism we described in Section 3. First, the intersection manager stops accepting reservations. Second, the intersection manager emits a signal informing the vehicles that an incident has taken place. There is a possibility that this second part might not always work perfectly; some vehicles might not receive the signal. As part of our experiment, we intentionally disabled some of the vehicles' ability to receive the emergency signal. A parameter in our simulator controls the fraction of vehicles created with this property, and we investigated the effect of varying this parameter on the number of vehicles incidents.

With the safety measures in full effect, the number of vehicles involved in the average incident decreases dramatically. Table 1 shows the numerical results for both the 3- and 6-lane intersections, along with a 95% confidence interval. While we would have liked to include the average crash logs for these runs in Figure 3, they would have been impossible to distinguish from one another. For that reason, we present them in Figure 4.

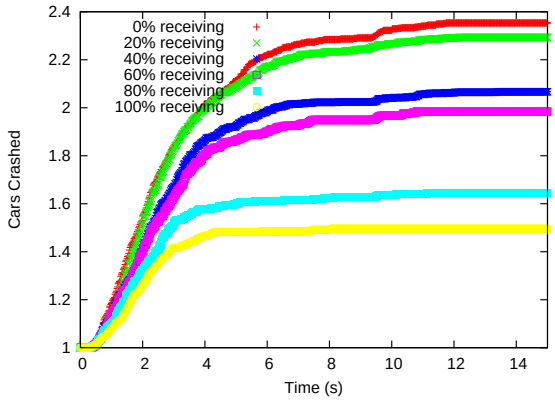
Figure 4 shows the effect of the safety measures on intersections with 6 lanes, with the proportion of receiving vehicles varying from 0% to 100% in increments of 20%. Even with no vehicles responding to the warning signal, the overall number of vehicles involved in the average incident declines by a factor of almost 30 in the fully autonomous scenario, and a factor of over 20 in the scenario with 5% human drivers. As expected, when more vehicles receive the emergency signal, fewer vehicles wind up crashing. The graphs in Figure 4 only show the first 15 seconds of the incident, because in no case did a collision occur more than 15 seconds after the incident started.

4.4 Severity of Collisions

While it is reassuring to know that the number of vehicles involved in the average incident can be kept fairly low, these data do not give the entire picture. For example, compare an incident in which 30 vehicles each lose a hubcap to one in which two vehicles are completely destroyed and all occupants killed. While we do not currently have any plans to model the intricate physics of each individual collision with



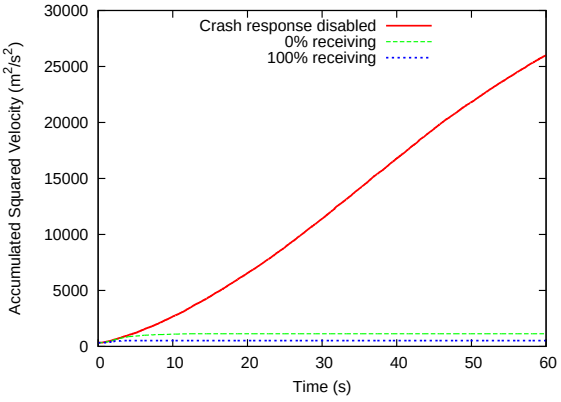
(a) All autonomous



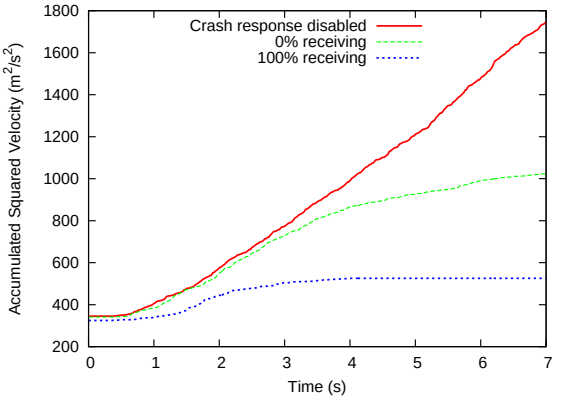
(b) 5% humans

Figure 4: The first 15 seconds of average crash logs for 6-lane intersections with all safety measures in place. As more vehicles react to the signal, safety improves further.

high fidelity, our simulations do give us access to the velocity at which the collisions occur. In the previous example, we might notice that the 30 vehicles all bumped into one another at low velocities, while the two vehicles were traveling at full speed. To quantify this information, we record not only when a collision happens, but the velocity at which it happens. In a collision, the amount of damage done is usually proportional to the amount of kinetic energy that is lost. Because kinetic energy is proportional to the square of velocity, we can use a running total of the squares of these crash velocities to create a rough estimate of the amount of damage caused by the incident. Figure 5 shows an average "damage log" of a 6-lane intersection of autonomous vehicles. Qualitatively similar results were found for the other intersection types, but are not displayed here due to space concerns.



(a) the average incident



(b) zoomed in

Figure 5: Average total squared velocity of crashed vehicles for a 6-lane intersection with only autonomous vehicles. Sending the emergency signal to vehicles not only causes fewer collisions, but also makes the remaining collisions less dangerous.

As Figure 5(a) shows, the improvement by this metric is quite dramatic as well. When no vehicles receive the emergency signal the total accumulated squared velocity decreases by a factor of over 25. When all vehicles receive the signal, it decreases by another factor of 2. Of particular note is the zoomed-in graph in Figure 5(b). Without the emergency signal, the total squared velocity accumulates as if no modification had been made, until the first vehicles stop short of the intersection at around 3 seconds; without a reservation, they may not enter. When the emergency signal is broadcast and all the vehicles receive it, the improvement is almost immediate.

5. DISCUSSION AND RELATED WORK

We believe that these experimental results raise a very important issue. People are often hesitant to put their well-being (physical or otherwise) in the hands of a computer unless they can be convinced that will receive a significant safety benefit in exchange for surrendering precious control. Humans often suffer from the *overconfidence effect*, erroneously believing they are more skillful than others. In a 1981 survey of Swedish drivers, respondents were asked to rate their driving ability in relation to others. A full 80% of those asked placed themselves in the top 30% of drivers [10]. It is this effect that creates the high standard to which computerized systems are held. It is insufficient for such systems to be marginally safer, or safer for the *average* user; they must be the very paragon of safety.

5.1 A Safer System Overall

In our experiments, we showed that the number of vehicles involved in individual incidents can be drastically reduced by virtue of some of the safety properties built into our intersection control mechanism. In fact, when all vehicles received the warning, a large portion of the incidents involved only one vehicle: the one we intentionally crashed. Even in the worst case—6 lanes of traffic and no vehicles receiving the warning signals—an average of only 3.23 vehicles were involved. But how does this compare with current systems? If we conservatively assume that accidents in traffic today involve only one vehicle, this represents a 223% increase per occurrence. Thus, all other things being equal, if the frequency of accidents can be reduced by 70%, the the autonomous intersection management system will be safer overall. A 2002 report for the Federal Highway Administration blamed over 95% of all accidents on driver error [11]. The report blamed 2% of accidents on vehicle failures and another 2% on problems with roads. It is important to note that these numbers are for all driving, not just intersection driving. Accidents in intersections are even more likely to be caused by driver error, sometimes by drivers willfully disobeying the law: running red lights and stop signs or making illegal "U"-turns.

Even if we make overly conservative assumptions—that all driving is as dangerous as intersection driving, and that driver error is no more accountable for intersection crashes than it is in other types of driving—our data suggest that automobile traffic with autonomous driver agents and an intersection control mechanism like ours will reduce collisions in intersections by over 80%. We believe that in reality, the improvement will be staggeringly greater.

The technique presented in this paper is just one method for improving the safety of this system's failure modes. More sophisticated methods involving explicit cooperation amongst vehicles may create an even safer system. The main thrust of our discussion is not that this particular safety mechanism is by any means the best possible. Rather, it is that even with this fairly simple response to accidents, the overall safety of the system can be strengthened well beyond that of current automobile traffic—all without sacrificing the bene-

fit of vastly improved efficiency.

5.2 Related Work

To the best of our knowledge, this paper represents the first study of the impact of an efficient, multiagent intersection control protocol for fully autonomous vehicles on driver safety. However, there is an enormous body of work regarding safety properties of traditional intersections. This includes the general—correlating traffic level and accident frequency [9] and analyses of particular types of intersections [1, 5, 7]—as well as plenty of the esoteric, such as characterizing the role of Alzheimer's Disease in intersection collisions [8]. However, because it concerns only human-operated vehicles, none of this work is particularly applicable to the setting we are concerned with in this paper.

6. CONCLUSION

In this paper, we discussed our previously proposed multiagent intersection control mechanism for autonomous vehicles. We believe the mechanism is promising, but we are not willing to sacrifice too much in the way of safety in the pursuit of efficiency. Our empirical results support our hypothesis that the mechanism can attain its high levels of efficiency without compromising on safety.

This work still leaves some unanswered questions. For example, we have examined only one method of disabling vehicles. In the future, we would like to explore other possibilities such as locking a vehicle's steering, simulating a blowout, sticking the accelerator, or disabling the brakes. For this paper, our aim was to initiate incidents that would test the limits of the intersection control mechanism by disrupting as much of the traffic flow as possible. A truly comprehensive failure mode analysis must include a much wider array of potential hazards. While our very conservative estimates indicate that this intersection control mechanism will be vastly safer than current systems with human drivers, we would like to conduct a more detailed study comparing the two, to quantify the improvement more precisely.

Autonomous vehicles, and the promise of easier and more efficient travel that they offer are a fascinating and exciting development. Before the benefits of this technology can be realized, much more work must be done to ensure that they are as safe as possible for the hundreds of millions of passengers that will use it on a daily basis. This failure mode analysis of our autonomous intersection management mechanism calls attention to the need for keeping an eye toward safety throughout the development of the algorithms and protocols that will control the transportation systems of the future. In this way, we believe we have accomplished a portion of this important work. Further analysis will of course be necessary, first in simulation, and ultimately with real physical vehicles.

Acknowledgments

This research is supported in part by NSF CAREER award IIS-0237699 and by the United States Federal Highway Administration under cooperative agreement DTFH61-07-H-00030. Computational resources were provided in large part by NSF grant EIA-0303609.

7. REFERENCES

[1] J. A. Bonneson and P. T. McCoy. Estimation of safety at two-way stop–controlled intersections on rural highways. *Transportation Research Record*, 1401:83–89, 1993.

[2] K. Dresner and P. Stone. Multiagent traffic management: A protocol for defining intersection control policies. Technical Report UT-AI-TR-04-315, The University of Texas at Austin, Department of Computer Sciences, AI Laboratory, December 2004.

[3] K. Dresner and P. Stone. Multiagent traffic management: An improved intersection control mechanism. In *The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 471–477, Utrecht, The Netherlands, July 2005.

[4] K. Dresner and P. Stone. Sharing the road: Autonomous vehicles meet human drivers. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, pages 1263–68, Hyderabad, India, January 2007.

[5] D. W. Harwood, K. M. Bauer, I. B. Potts, D. J. Torbic, K. R. Richard, E. R. K. Rabbani, E. Hauer, L. Elefteriadou, and M. S. Griffith. Safety effectiveness of intersection left- and right-turn lanes. *Transportation Research Record*, 1840:131–139, 2003.

[6] National Highway Traffic Safety Administration. Economic impact of U.S. motor vehicle crashes reaches $230.6 billion, new NHTSA study shows. NHTSA Press Release 38-02, May 2002. `http://www.nhtsa.dot.gov`.

[7] B. N. Persaud, R. A. Retting, P. E. Gardner, and D. Lord. Safety effect of roundabout conversions in the united states: Empirical bayes observational before-after study. *Transportation Research Record*, 1751:1–8, 2001.

[8] M. Rizzo, D. V. McGehee, J. D. Dawson, and S. N. Anderson. Simulated car crashes at intersections in drivers with Alzheimer disease. *Alzheimer Disease and Associated Disorders*, 15(1):10–20, 2001.

[9] T. Sayed and S. Zein. Traffic conflict standards for intersections. *Transportation Planning and Technology*, 22(4):309–323, August 199.

[10] O. Svenson. Are we all less risky and more skillful than our fellow drivers? *Acta Psychologica*, 47(2):143–148, February 1981.

[11] W. W. Wierwille, R. J. Hanowski, J. M. Hankey, C. A. Kieliszewski, S. E. Lee, A. Medina, A. S. Keisler, and T. A. Dingus. Identification and evaluation of driver errors: Overview and recommendations. Technical Report FHWA-RD-02-003, Virginia Tech Transportation Institute, Blacksburg, Virginia, USA, August 2002. Sponsored by the Federal Highway Administration.

Extending microscopic traffic modelling with the concept of situated agents

Paulo A. F. Ferreira, Edgar F. Esteves, Rosaldo J. F. Rossetti, Eugénio C. Oliveira
Artificial Intelligence and Computer Science Laboratory (LIACC)
Faculty of Engineering, University of Porto (FEUP)
Rua Dr. Roberto Frias, S/N • 4200-465 Porto • Portugal
T: (+351) 225081315 F: (+351) 225081440
{paff, edgar.esteves, rossetti, eco}@fe.up.pt

ABSTRACT
In the traffic simulation field, there is a general agreement that microscopic simulation is becoming more viable, improving the way in which system elements are represented. However, even with more powerful computational resources made available trading off between realism and too much abstraction is an important issue to overcome, as traditional micro-simulation approaches still fail to profit from all benefits that realism could offer to traffic modelling. In this work we bring this discussion forward and propose a multi-agent model of the traffic domain where integration is ascribed to the way the environment is represented and in which agents interoperate in microscopic simulations. While most approaches still deal with drivers and vehicles indistinguishably as a single entity, in this work vehicles are merely moveable objects whereas the driving role is played by an agent fully endowed with different cognitive abilities situated in the environment. We start by discussing on the role of the environment dynamics in supporting a truly emergent behaviour of the system, and then on an extension of the traditional car-following and lane-change models with the concept of situated agents. A physical communication model is proposed to explore the different perception abilities of each single driver agent as the basis for different interactions and overall system behaviour. Some performance issues are also identified and the result is a more flexible structure that allows for a more realistic representation of drivers' behaviour in microscopic simulation models.

Categories and Subject Descriptors
D.2.1 [**Software Engineering**]: Requirements/Specifications – *methodologies (agent-oriented)*

I.2.11 [**Distributed Artificial Intelligence**]: i*ntelligent agents, multi-agent systems.*

General Terms
Algorithms, Design, Human Factors.

Keywords
Situated Agents, Agent-based Traffic Simulation, Microscopic Modelling and Simulation, Car-Following, Lane-Changing

1. INTRODUCTION
The issues concerning traffic and transportation in urban scenarios are so evident that regular users have already realised that most infrastructures are working near their saturation condition mostly due to the more than ever increasing demand. This inevitably implies considerable economic, social, and environmental losses that must be minimised somehow. Some attempts to cope with the potential limitation of road capacity have been put into practice, such as physical modifications to the infrastructure and the improvement of control systems. The former is no longer the best alternative to tackle such a problem. Besides the high cost of implementation, it causes disruptions and damages the environment. In the latter situation, some good advances and successful experiences have contributed to the reduction of problems related to traffic jams. Despite the relatively good improvements they are able to produce, they cannot be considered a lasting solution either. Therefore, current research still seeks alternative means to cope with the traffic and transportation domains.

Using simulation is imperative in planning and realising the correct relation among the parameters of the domain. However, most analyses are carried out on an individual basis as an attempt to reduce the number of variables observed and to simplify the process of finding out their correlations. This brings about the issue of how different standpoints from which the domain is viewed could be coupled in the same model and simulation environment in order to allow for wider analysis perspectives [1][9][16]. This is not a recent concern, though. The basic general framework for a fully transportation theory identifies two different concepts, borrowed from Economics, which encompass all aspects related to demand formulation and supply dynamics within the framework, including multi-modal selection and activities planning [11].

Arguably, realistic models are the first instrument to allow the integration of different analysis perspectives in virtually any application domain. However, modelling is not an easy task and abstraction is often necessary in order to make thinks feasible. The autonomous agent metaphor has been increasingly used in this way and offers a great deal of abstraction while important cognitive and behavioural characteristics of the system entities are preserved. Also, advances in engineering environments for multi-agent systems have fostered the idea of overall system behaviour

that emerges from the interaction of microscopically modelled entities.

In this paper we bring this discussion forward and ascribe to the environment the responsibility for coping with the complexity inherent in the transportation domain, more specifically in the field of traffic modelling and simulation, in order to provide engineers and practitioners with an adequate framework for integrated analyses. Complexity is expected to emerge from the interaction of simpler self-cantered autonomous entities in pursuit of maximizing some individual or collective utility measure. We start by discussing on some potential applications of the concepts of agents and multi-agent systems to such a complex domain, in section 2, and try to bring about the importance of the environment abstraction to agent-based simulation in section 3. A detailed explanation on the interaction mechanism used to support the implementation of situated agents is presented in section 4. In section 5 we conceive the architecture of a system to support practical simulation of traffic scenarios on the basis of the concepts discussed, which is followed by some interaction examples to illustrate the approach proposed, in section 6. Some conclusions are drawn and presented with important considerations for future developments.

2. MAS-T: MULTI-AGENT SYSTEMS APPLIED TO TRANSPORTATION

The abstraction approach of MAS consists of representing a system by multiple entities that exist in a common environment and interact in order to achieve specific goals. These entities that are coined agents, exhibit intelligence, autonomy and some social ability. Some examples of applied MAS in the field of traffic and transportation engineering can be found in the literature [2][5][13]. However, most of the applications are concerned with the control system, even though it is possible to recognise an increasing interest in the driver element. The assumption in the former examples relies on the representation of adaptable control system as a community of controller agents, which co-operate in order to achieve an optimum plan to meet the variable demand [14]. In these cases the movement is represented on the basis of simplified models that, in the great majority, adopt a simple approach of using a reactive structure. Other models where communication mechanisms and drivers with mental attitudes are of importance are found in [3][15][17].

Transportation Engineering is definitely a very broad field of knowledge and contemporarily has evolved so quickly as Intelligent Transportation Systems (ITS) start to make part of everyone's daily life. According to [4], the underlying concept of ITS is to ensure productivity and efficiency by making better use of existing transportation infrastructures.

From what has been discussed above it is reasonable to see this domain as formed of heterogeneous entities, which are geographically and functionally distributed throughout the environment. They pursue individual or collective goals, interact with one another and may transform the environment as well. From observation it is possible to realise three main components in our application domain, namely the moving element, the control system and the road network.

In very basic terms, the moving element is the vehicle that moves from one point to another throughout the network. Disregarding the importance of pedestrians in this first stage of this work, we consider bicycles, motorcycles, automobiles, trucks and buses as examples of moving elements. However, they are actually moving objects steered by their drivers and sometimes occupied by many other passengers that are people with a trip purpose. Also, their decision concerning how the trip will be carried out in most cases seeks to minimize some individual sense of cost. Therefore, we make a clear distinction between travellers and vehicles.

From a transport planning perspective, the inhabitants of urban areas are potential travellers with specific trip needs. Prior to each journey, travellers must make some options basically regarding mode of transport (whether to drive their own cars or to take a public transport service, for instance), the itinerary and a departure time. To the contrary, in the traffic system perspective flow is actually formed of each single vehicle. Nonetheless, vehicles moving throughout the network are steered by their drivers and hence drivers and vehicles are dealt with indistinguishably in virtually the totality of microscopic models [7][8][10]. In the microscopic point of view, it is the driver behaviour that influences traffic flow. Actually drivers manifest an interesting yet implicit social interaction – they compete for the limited resources of the network infrastructure. These different interactions may emerge on an aggregate perspective as these properties will become available in terms of natural stimuli to the inhabitants, who will behave accordingly as they have different perception capabilities and different goals.

3. THE ENVIRONMENT ABSTRACTION

The perspective over **environments** for MAS has been changing in the direction of an increasing importance of this entity. Danny Weyns and colleagues [19] stress out the importance of considering the environment as a first-order abstraction in the engineering processes of developing MAS. Weyns recalls a classical definition of **autonomous agent**: "a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future" [6]. From this definition he states "the importance of the environment as the medium for an agent to live, or the first entity the agent interacts with" [20]. He also recalls the notion of embodiment as "the fact that an autonomous agent has a "body" that delineates it from the environment in which the agent is situated".

Let us take a better look at the definitions above (of autonomous agent and of embodiment). The first states that the agent is not only situated in the environment: it is a component part of that environment. While not contradicting, this is diverse from the second definition which presents the agent and the environment as separate (and separable) entities. We could redefine an agent's body as a subset of the environment. This allows us to clearly define the agent (the agent still has a body) while providing a wider and more complex notion of environment. We will refer as the agent's body as the agent's internal environment. The environment without the agent's body is the agent's external environment.

In [12], authors differentiate a *physical environment* and a *communication environment*. The physical environment models the physical existence of objects and agents, whereas the communication environment includes the structures that support exchange of information (knowledge). These include roles, groups

and communication protocols. They further define *social environment* as a restriction to the set of communication environments. A social environment is "*a communication environment in which the agents interact in a coordinated manner*". Note how the definition somehow restricts the forms of communication that may occur in a MAS. Tummolini and colleagues [18] introduce *Behavioural Implicit Communication*, in which case communication clearly occurs at the physical level (via perception), diverging from Odell's definition [12].

Both views can be unified by extending communication to the *physical environment*. We then classify communication into two main modes: *implicit communication*, occurring in the physical environment and *explicit communication*, occurring in the communication environment and regulated by high-level protocols (out of the scope for this paper). We further classify implicit communication into two distinct forms: *physical communication* (related to the observability of objects and agents) and *behavioural communication* (related to the observability of agent's actions). For the rest of this paper, we will focus on the implicit forms (physical and behavioural).

Physical communication occurs when an agent produces influences over its external environment, these influences produce a state change in that environment, that state change is perceived and interpreted by another agent (could be more than one), and this agent possibly changes his own behaviour in face of the interpretation. As an example of physical communication, consider the following scenario. When a driver wishes to communicate a lane change to the neighbouring agents, it switches the appropriate car light on. This implies **producing an influence** that will most likely result in a **state change** of the vehicle object controlled by the driver. This change will be detected by the agents that "*pursuing their own agenda*", are scanning the environment for visual perception. Some of the agents will **interpret** the state change as an intention of the peer driver and possibly change their own behaviour in face of the peer's intentions.

Behavioral communication works the same way around, with the difference that it occurs when an agent produces influences over its own internal environment. Examples would be a semaphore controller agent switching the signals, or a flagman waving his arms. The action consists of a list of influences over the agent's own body (the *internal environment*), although success or failure may still depend on the *external environment* (i.e., a power failure would prevent the semaphore controller from switching the lights). This is a very important feature of the model. An agent does not fully control its *internal environment*, since it is also a part of the coexisting agent's *external environment*, and so the agent may be "forced" by these external actions, at least up to some extent. Finally, an action may influence both the internal and external environments at the same time. This is transparent in our model, since both forms of communication are leveled by the way agents send their influences and receive the "messages" (via perception).

With these notions of environment and communication in mind, we will elaborate a definition of *physical environment* that stresses on the fact that an agent (and all other agents and objects) is *part of* the environment, instead of being merely inserted in it. We define it as collection of entities and laws. Entities may be objects (inanimate, yet possibly reacting or interactable) and agents (animate and partially *autonomous*). These entities and their interactions are ruled by a set of laws about their own properties and about the environment. All these collections are dynamic (objects may be created, consumed, transformed into other objects; agents may enter or leave the environment, "die" or be "born"). As a draft of a more formal approach we may say that

$$Env(t) = \{Objs(t), Ags(t), Laws(t)\}.$$

An object is characterized by:

- A set of *perceptible features* (*PF*), representing all possible features that may be perceived by agents. A feature may or not be active. We could identify the set of active features in a given time *t* as *PF(t)*. It should also be possible to provide the features with operational (run-time) parameters. As an example, consider the lights of a car. They are always perceptible but the current state of the light may change in each time step (it makes sense that a light is a feature that is always active but it may be "on" or "off" e.g., it is a run-time parameter of the feature).
- A set of *interactable features* (*IF*), representing interfaces that provide the environment access to modify the object state. Agents will not have direct access to the *IFs*. The set of active features in a given time *t* is *IF(t)*.
- A *set of properties* (*SP*), representing part of the internal state of the object. Note that we do not restrict the internal state to *SP*. Instead, we consider *SP* is part of the entities' internal state (which also includes the *PF* and *IF* sets).

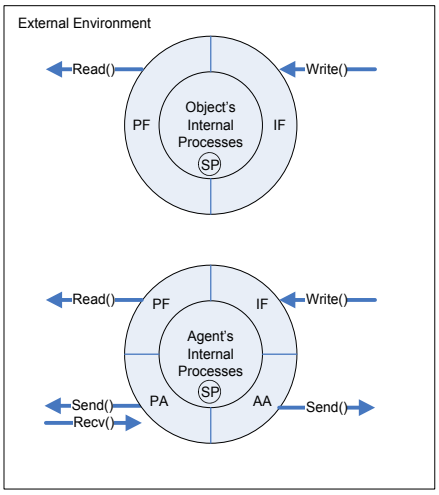


Figure 1 - Primary interfaces of objects and agents with the external environment

To limit the agents' autonomy, reflecting the fact that agents are conditioned by the environment of which they are a part of, and allow for *influences* of the environment over themselves, we define *agent* as a subclass of *object*. The agent may at best have partial control over these influences. This is fundamental to our approach. We long for a highly complete model to accommodate complex environments, allowing agents and agent's actions to be perceived by other agents (agents' actions also have perceptible features) and forced influences from the environment to be performed on the agents. Besides the inherited sets, an agent has:

- A set of *perception abilities* (*PA*), that the agent uses to send messages to the environment expressing the current *foci* of the preceptors and receive messages from the environment with perceptual representations (we will elaborate on this). For performance reasons, only one message is sent/received at each time step, possibly containing several foci/representations.
- A set of *action abilities* (*AA*), that the agent uses to send messages to the environment expressing influences over the agents internal and/or external environments (again, we will restrict agents to send only one message in each time step, though possibly expressing several influences).

Figure 1 illustrates how these sets provide the interface with the external environment of both agents and objects.

4. THE INTERACTION MECHANISM

To connect the basic concepts of our model, we illustrate the relations among the fundamental entities and roles in a class diagram where the roles are specified as "interfaces" (Figure 2).

The basic design of the suggested architecture is to consider that every entity in the traffic system is an object that influences the PF's of agents (which are also objects). To achieve a desired perception of a part of the environment an agent becomes a listener by sending its *foci* to a mediator. All objects within the listener foci become its casters (becoming a caster of a listener means that the listener must perceive the casters' PF). The mediator is responsible for translating the casters' PF according to the current state and laws that rules the world and sending the set of perceptions to the listener accordingly. The interpretation of the set of PF's received by each listener are stored or updated in the knowledge base of the respective agent. We name this type of knowledge as the agent cognitive map. Anticipating performance issues, and relating model elements to real world counterparts, we consider that the traffic environment can be divided into zones, each of which will be assigned a mediator. More on this topic will be discussed later on in this paper.

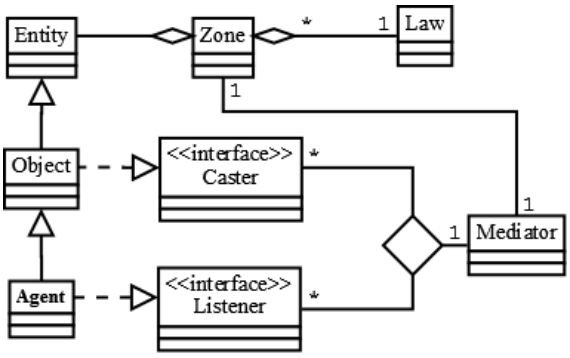


Figure 2 - Class diagram with the fundamental entities and roles

Thus, each mediator contains a representation of all entities inside its zone. So a listener sends its influence (for example a car that accelerates influences the external environment) and its foci to the mediator that updates its zone representation. The mediator contains a representation of every agent structure, allowing the correct interpretation of the agent's set of PF and all its internal states, and updates the necessary information into that structure based on the influence sent by the listener. The influence created by an agent affects the entire surrounding environment and consequently the perception of it. The mediator is also responsible for finding the correct casters for each listener based on the foci sent by each agent in every time step. All objects inside a given foci become casters to that listener. These casters are basically the entities that exist in the mediator, representing agents or objects in its environment zone that are inside the agent *foci*.

The mediator is able to read and write the state of any object including access to the objects' PF's (for example, if an agent is a car and it decides to turn on the lights it will change the characteristics of the front vehicles because they become more illuminated, so a perception feature (illumination) was changed in those vehicles because of an influence made by the listener. Therefore the mediator needs to access those objects internal perception future set, search for the illumination perception future in it and change it to a new value accordingly). If necessary, then the mediator alters the perception features of the casters based on the influence provoked by the listener and after it reads all of the casters perceptions features. With this information it builds the perception of the listener having into account the laws of the environment (for example, if a listener is looking at its front and has a truck and a car as its casters, if the caster car is in front of the truck and the listener car is very near to the truck the listener car cannot receive perception of the caster car, unless the law of the environment rules that trucks are transparent).

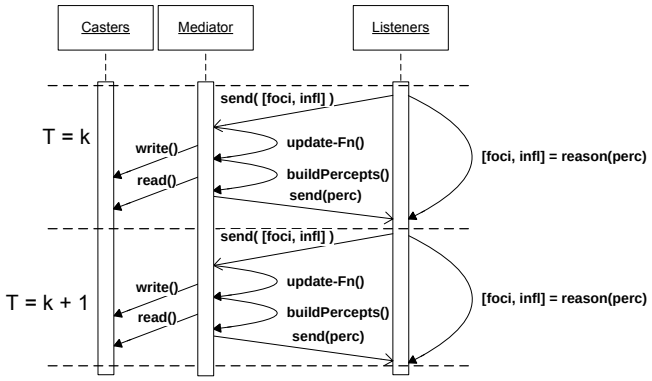


Figure 3 - Sequence diagram, detailing the interactions

With this perceptions received the listener must update its cognitive map. A cognitive map can be understood as a human driver mental perception of the objects surrounding its vehicle (other cars, traffic signs, traffic lights, people, buildings, and so on). An agent cognitive map is dynamically updated according to the perceptual representations received by its PA and by the execution of AA. After finishing updating the cognitive map a time step cycle is terminated. When a new one begins each agent has to decide on the action (influence) it must take, and where to focus its attention. The decisions are made based on the information of the cognitive map updated on the last time step, and on its own desires (desired destiny, desired speed, desired sight, and so forth). Sometimes the information contained on the cognitive map is not enough for an agent to transform a desire into an intention causing the agent to engage in a course of actions (e.g., it desires a left lane change but the left back vehicle perception is too old to risk it without updating it first). In these

cases, an agent can continue its movement and focus its attention to the desired scene in order to obtain the necessary information to fulfil its desires.

The model of the interaction mechanism explained is depicted in Figure 3, in form of a sequence diagram, and the concepts herein presented are illustrated in a more concrete way through an example scenario in section 6.

5. SYSTEM OVERVIEW

According to what has been discussed so far a distributed system is defined to support the implementation of a microscopic simulation engine (MSE). The MSE contains all the world states and objects, and the laws of the world. It also contains the mediators that will translate and send the updated perceptions of objects to the agents that need them. The necessity of a distributed system is a must to guarantee system efficiency and also as a natural way to implement the entities of our application domain.

Basically the world is represented by a set of zones each containing a mediator. Each zone runs independently, having a centralised process that is responsible for the coordination of the world time steps (it guarantees that every zone processes the correct time step, meaning that it is not possible to have different zones processing simultaneously at different time steps). This synchronous process is also responsible for reading the topology of the networks, dividing them into zones, receiving the registration of mediators, and assigning them an appropriate zone. A simulation cannot be started until each zone has been assigned a mediator. It also allows registration of graphical interface modules providing them, in the registration, with the address of each mediator. Such a structure also allows for the simulation to run with no graphical support, which can contribute to speed up simulation studies.

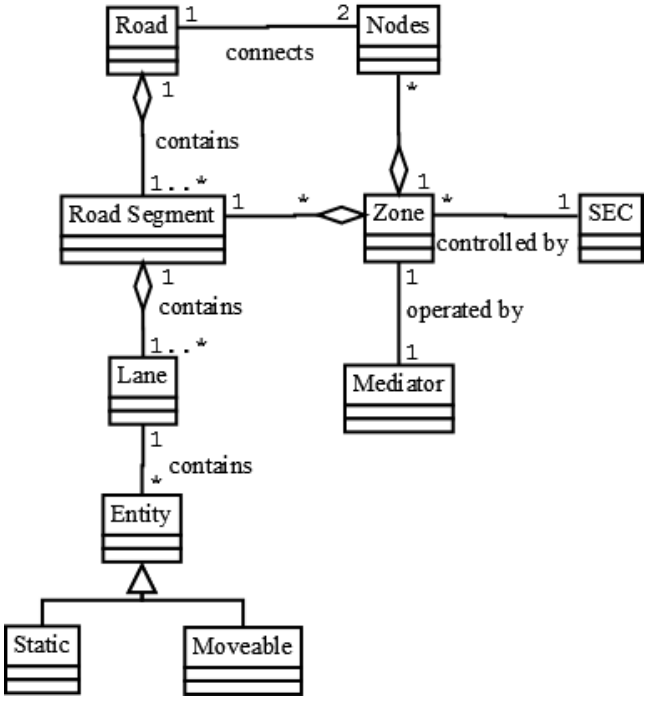


Figure 4 - Class Diagram of the environment domain

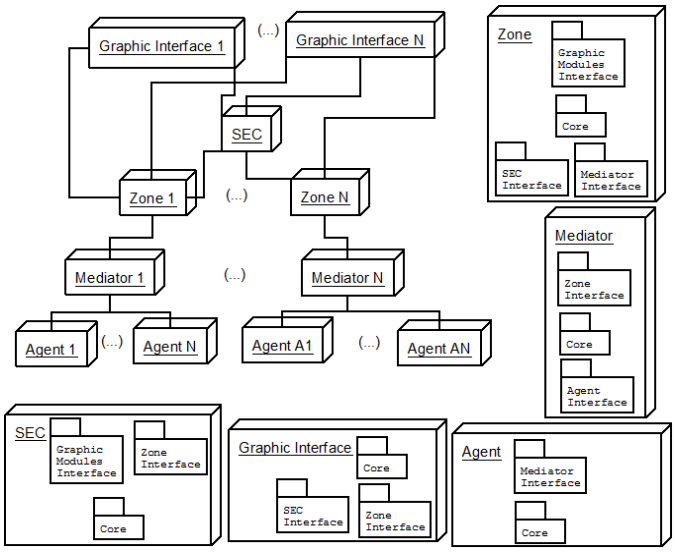


Figure 5 - System Physical Overview

Each mediator provides a communication interface responsible for sending the updated zone states to the different graphical interface modules so they can create real-time graphical representations of the simulation in runtime. There is also a centralised process that provides a communication interface for these graphical modules in order to allow them to stop or to start simulation runs, change environment characteristics such as the set of laws, load different networks, save simulation states, and so on. Such a centralised process is named SEC (**S**imulation **E**ngine **C**ontroller).

In Figure 4, it is possible to identify the domain entities, as well as their relations. A connection between two nodes represents a road. A road is a set of road segments. The division of a road into road segments depends on the different number of lanes or the different geometry a road can have. For example, if in the beginning of a road there are two lanes, but in the middle of the road it passes to have only one lane, it means that the road has two road segments – a road segment with two lanes and another road segment with just one lane. The world objects are decomposed into two different entities, namely the entities that have the capacity to move (vehicles and people, for instance) and the ones that are static (traffic controllers, road signs – both horizontal and vertical, road obstacles, and so on). In every given time an entity is situated in a lane.

A system physical overview is represented in Figure 5. In that structure a mediator has always the necessary information to construct perceptions for the correct behaviours of the world agents. Their interaction will follow the mechanism proposed in this research.

5.1 Environment Zones

Since the perception treatment and communication can be a heavy load for overcrowded networks the distribution of the environment perceptions becomes critical in order to improve the global efficiency of the simulation.

In order to assure that each agent receives the world perception efficiently in every time step, we assume that the process that delivers it has a limited capacity of the number of agents it has to inform. So a distributed division of the environment is a question of defining the correct capacity limit and number of perceptions a

mediator will be dealing with. Such an organisation easily resembles the concept of traffic zones, used in control and management systems in most urban areas.

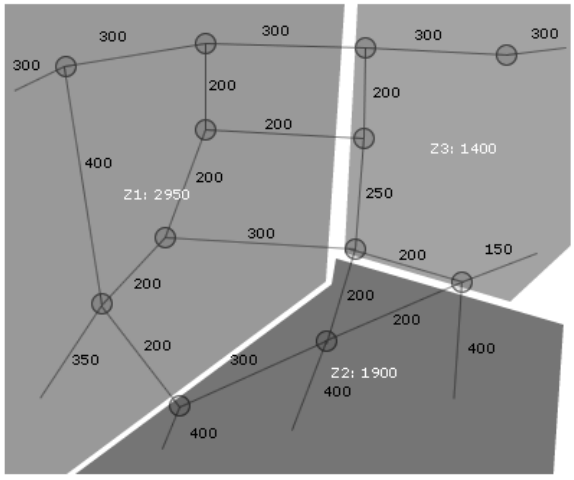


Figure 6 - Example of a possible network

As defined before, the entities responsible for the delivery of the perceptions are the *Mediators*. Analyzing the scenario presented in Figure 6 and assuming that M1 has a limit capacity of 3000 vehicles, M2 of 2000 vehicles and M3 of 1500 (the limit capacity of Mediators is calculated based on the processing capacity of the machine in which they are instantiated). The division into different Mediator zones is easy to obtain. Each link (Road Segment) has a physical capacity, limiting the quantity of vehicles it can contain. This means that in the worst scenario each road segment will only ensure that number of vehicles. So a mediator zone is defined as a set of road segments, whose sum of their capacities is equal or lower to the limited vehicle capacity of its mediator.

This way it is possible to guarantee that the mediator will process, in the worst scenario, the world perceptions of a number of drivers equal or lower to its own capacity. Translating it to the scenario of Figure 6, M1 will be assigned zone 1 (Z1 in the figure), M2 will be assigned Z2 and M3 will be assigned Z3. This means that each of the Mediators will be responsible for translating the zone objects' perceptible features to all agents inside its assigned zone that ask for it.

6. EXAMPLE SCENARIO

Consider the following scenario as depicted in Figure 7, representing the current state of the environment and already populated with all the casters and listeners that will interact throughout the example. The visual focus (for the current time step) of the agents controlling vehicles A, B, C and D is represented by the highlighted circle slices. In fact we opted to represent all of these vehicles to explain different situations that occur in traffic simulations and also to explain the concepts related to the "car following" (CF) and "lane changing" (LC) behaviours. Let us call the agents by the letters on the vehicles. Along with their foci, they have also expressed the influences over the environment for this time step.

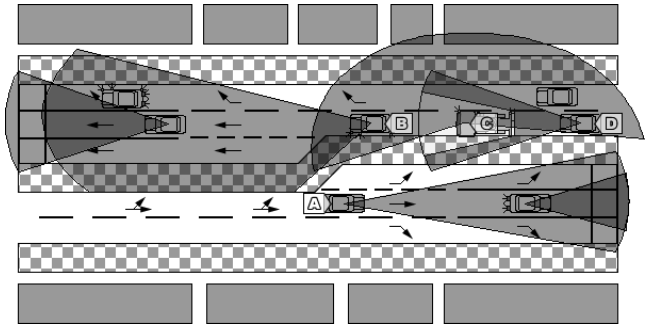


Figure 7 - An example of a time-step of the simulation

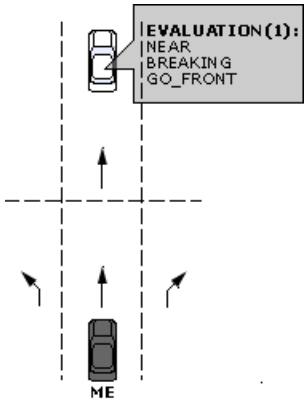


Figure 8 - Cognitive map of agent A

To ease the understanding of the concept of CF let us centre on agent A. Since it wants to go in front, its foci are naturally the front area. Take into account that as it becomes a listener the front vehicle becomes its caster. In the meantime the cognitive map of the agent (see Figure 8) is updated according to the interpretation of the received PF's (given by the Mediator). This information is given with regard to the object which is being observed by the subject driver, so perceptions are enclosed into balloons attached to the object being observed.

For this specific example agent A will take the particular action of "BRAKING". That's because it does not have any previous deduction (previous time step) of the other vehicle's velocity ("REALLY FASTER"; "FASTER"; "SLOWER" and "REALLY SLOWER"). In the next time step it will send that action to its mediator.

More complex situations can occur, like demonstrated by agents B and C. Agent B was having the same attitude as the one demonstrated before but new variables will make it to change (see Figure 9). Assuming that it wants to go in front, a new lane appears in that direction and the front vehicle was evaluated as going "SLOWER". It will take the action "CHANGE_TO_LEFT_LANE" then. This kind of actions

transpires when an agent wants to maintain or achieve its desired velocity and is inherited from the lane changing concept.

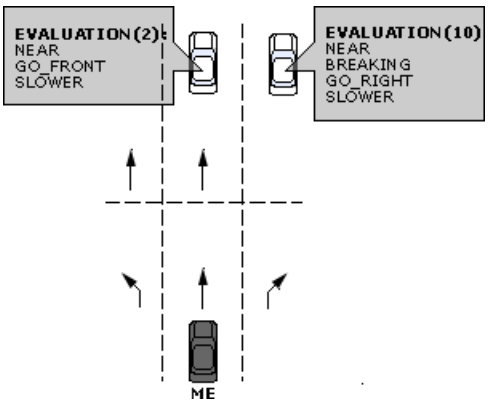


Figure 9 - Cognitive map of agent B

The previous figure also illustrates a representation of a "front right vehicle" that is having the intension of turning right. If in the next time step it transforms its intention into an influence, it will be deleted from agent B cognitive map.

At the same time agent C is in a delicate situation. It needs to go to the right lane to accomplish its path direction previously defined (supposing). Like in real situations, in which we need to look into the mirrors and take care with the front vehicle, it sets its foci to the front, back and right sides. The Mediator informs it about all the casters positions, velocities, acceleration and intentions (PF's) and the evaluation of its cognitive map will be like the one represented in Figure 10. The fact that the back right vehicle is being faster than itself will not permit the lane changing in the current time step (according to its own AA's) forcing it to wait for the next time step. If in future steps the back right vehicle does not pass him or new similar situations occur it will be impossible to make that action and agent C will be forced to stop.

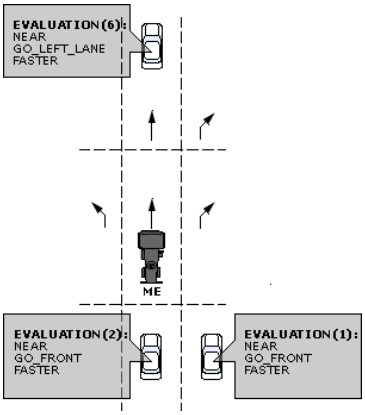


Figure 10 - Cognitive map of agent C

Taking into account that in human behaviours there is also a factor of cordial attitudes, it is agreeable to think that agent C can try to change to another lane to let pass the back vehicle (since its velocity evaluation is "FASTER"). This kind of actions is also inherited from the lane changing concept.

The representation of agent D intends to illustrate two different kinds of laws in the present scenario (transparent and opaque objects). The PA's are affected by this laws since the Mediator interprets the PF´s according to them and to the agent' foci. As a consequence the casters are not the same in the two different configurations. In the case of agent D there are two vehicles directly in front of it and inside its foci (C and B). If the laws of the environment are configured to opaque objects then the mediator won't give D the PF's of object B (vehicle C is a truck and blocks the visibility of agent D). Otherwise if the laws are configure to allow transparent objects then both C and B PF's will be included in the information that the mediator will send to agent D. This example shows the influence that the laws of the environment can have in the capture of perception of each agent.

In Figure 8, Figure 9 and Figure 10 notice the numbers that appear inside the parentheses and after the evaluation word. Those numbers represent the last time that the evaluation of that caster was done. That means those agents have more or less trust on their evaluations according to their PA's (for instance, if a back car is FAR and SLOWER the agent does not need to verify whether it is near every time step). It is possible to have a factor in each agent that dictates how each agent will trust on predicting future positions of its surrounding objects. For example, consider that an agent looks back in time step n and gets the perception of an agent called X. If in the time step n+20 the agent needs the information about agent X to perform an influence, it must decide whether to have to look back to update X perception on its cognitive map or if it trusts its future prediction on information perceived 30 time steps ago.

A prototype of the proposed system is being developed and some basic features of the communication mechanism were implemented, demonstrating the potential of this approach in extending traditional car-following and lane-changing behaviours. The environment is a first-order abstraction that plays an imperative role in this framework being developed. An example of its interface is depicted in Figure 11.

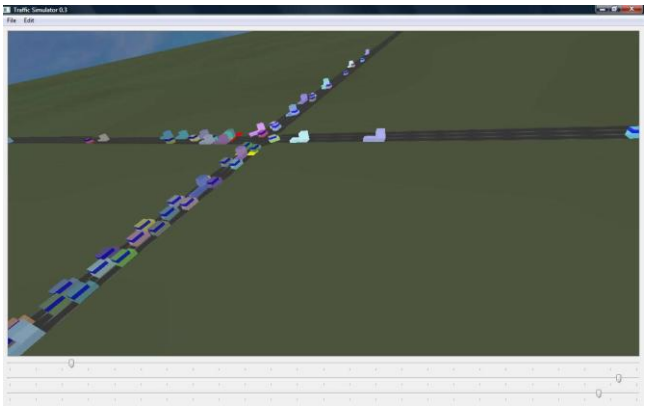


Figure 11 – Prototype of the simulation environment

7. CONCLUSIONS

In this work we propose a multi-agent model to cope with the complexity inherent in microscopic traffic simulation modelling in order to provide engineers and practitioners with an adequate framework for integrated analyses. The physical conceptualization of the environment using the interaction mechanisms presented as the basis for every interaction among agents and the environment itself allows for different perception abilities of individuals to be implemented and assessed, which is expected to have a direct

influence in the emergence of the system overall performance in different circumstances. Therefore, a truly agent-based microscopic simulation approach must necessarily be build on the basis of the concept of situated agents and consider the environment as a first-order abstraction, playing as relevant roles as other entities in the system. In this way, as drivers are integrant parts of the environment and interact directly with it, more realistic behaviours can now be considered. With such a concept of environment, traditional car-following and lane-changing models can be extended to feature more contemporary performance measures, which can include influence of road-side parking, collisions, interaction with traveller information systems, en-route decision-making, and many others. This is just possible as different perception abilities of drivers can now be considered in the way they interact with their environment. An initial prototype with very simple features of the presented model has been implemented, to demonstrate car-following and lane-changing behaviours. The very next steps in this research include the improvement of this prototype to fully demonstrate all the potential of the concept of situated agents and the role of the environment in implementing more realistic microscopic traffic simulations. Also in the agenda, we expect to devise an appropriate methodology for validating and calibrating such agent-based microscopic traffic models. Following this, some simulations and analyses of performance measures will be carried out as well.

ACKNOWLEDGMENTS

We gratefully acknowledge the financial support from the Department of Electrical and Computer Engineering at Faculty of Engineering, University of Porto.

REFERENCES

[1] Barcelo, J. 1991. Software environment for integrated RTI simulation systems. In Advanced Telematics in Road Transport, Proceedings of the DRIVE Conference. Elsevier, Amsterdam, v.2, 1095-1115.

[2] Bouchefra, K., R. Reynaud, and T. Maurin. 1995. IVHS viewed from a multiagent approach point of view. In Proceedings of the IEEE Intelligent Vehicles Symposium. IEEE, Piscataway, 113-117.

[3] Burmeister. B, A. Haddadi, G. Matylis. 1997. Application of multi-agent systems in traffic and transportation. IEE Proceedings on Software Engineering, v.144, n.1, 51-60.

[4] Chatterjee, K., M. McDonald. 1999. Modelling the impacts of transport telematics: current limitations and future developments. Transport Reviews, v.19, n.1, 57-80.

[5] Davidsson, P., L. Henesey, L. Ramstedt, J. Törnquist, F. Wernstedt. 2005. An analysis of agent-based approaches to transport logistics. Transportation Research Part C, v.13, n.4, 255-271.

[6] Franklin, S., A. Graesser. 1996. Is it an agent or just a program? A taxonomy for autonomous agents. In Intelligent Agents III. Agent Theories, Architectures, and Languages. LNAI, n.1193, 21-35.

[7] Gipps, P. G. 1981. A behavioural car-following model for computer simulation. Transportation Research Part B, v.15, 105-111.

[8] Gipps, P. G. 1986. A model for the structure of lane-changing decisions. Transportation Research Part B, v.20, 403-414.

[9] Grazziotin, P.C., B. Turkienicz, L. Sclovsky, C.M.D.S. Freitas. 2004. CityZoom - A Tool for the Visualization of the Impact of Urban Regulations. In Proceedings of the 8th Iberoamerican Congress of Digital Graphics. 216-220.

[10] Hidas, P. 2000. How can autonomous agents help microscopic traffic simulation. In 1st Workshop on Agents in Traffic and Transportation, Barcelona. 2000.

[11] McNally, M.G. 2000. The four step model. In Handbook of Transport Modelling. Pergamon Press, Oxford, 35-52.

[12] Odell, J., H.V.D. Parunak, M. Fleischer. 2003. Modeling Agents and their Environment: the communication environment. Journal of Object Technology, v.2, n.3, 39-52.

[13] Oliveira, E., N. Duarte. 2005. Making way for emergency vehicles. In the European Simulation and Modelling Conference. Ghent, EUROSIS-ETI, 128-135.

[14] Roozemond, D.A. 1999. Using autonomous intelligent agents for urban traffic control systems. In Proceedings of the 6th World Congress on Intelligent Transport Systems.

[15] Rossetti R.J.F., R.H. Bordini, A.L.C. Bazzan, S. Bampi, R. Liu, D. Van Vliet. 2002. Using BDI agents to improve driver modelling in a commuter scenario. Transportation Research Part C, v.10, 373-398.

[16] Rossetti R.J.F., S. Bampi. 1999. A Software Environment to Integrate Urban Traffic Simulation Tasks. Journal of Geographic Information and Decision Analysis, v.3, n.1, 56-63.

[17] Rossetti, R.J.F., S. Bampi, R. Liu, D. Van Vliet, H.B.B. Cybis. 2000. An agent-based framework for the assessment of drivers' decision-making. In Proceedings of the IEEE Conference on Intelligent Transportation Systems. IEEE, Piscataway, 387-392.

[18] Tummolini, L., C. Castelfranchi, A. Omicini, A. Ricci, M. Viroli. 2004. Exhibitionists and Voyeurs do it better: a Shared Environment for Flexible Coordination with Tacit Messages. In 1st International Workshop on Environments for Multiagent Systems. LNAI, n.3374, 215-231

[19] Weyns, D., H. Parunak, F. Michel, T. Holvoet, J. Ferber. 2005. Environments for multiagent systems, State-of-the-art and research challenges. In 1st International Workshop on Environments for Multiagent Systems. LNAI, n.3374, 1-47.

[20] Weyns, D., M. Schumacher, A. Ricci, M. Viroli, T. Holvoet. 2005. Environments in multiagent systems. The Knowledge Engineering Review, v.20, n.2, 127-141.

Replacing the Stop Sign: Unmanaged Intersection Control for Autonomous Vehicles

Mark VanMiddlesworth
Harvard University
Elec. Eng. and Comp. Sci. Department
mvandmidd@fas.harvard.edu

Kurt Dresner
University of Texas at Austin
Department of Computer Sciences
kdresner@cs.utexas.edu

Peter Stone
University of Texas at Austin
Department of Computer Sciences
pstone@cs.utexas.edu

ABSTRACT

As computers inevitably begin to replace humans as the drivers of automobiles, our current human-centric traffic management mechanisms will give way to hyper-efficient systems and protocols specifically designed to exploit the capabilities of fully autonomous vehicles. We have introduced such a system for coordinating large numbers of autonomous vehicles at intersections [4, 5]. Our experiments suggest that this system could alleviate many of the dangers and delays associated with intersections by allowing vehicles to "call ahead" to an agent stationed at the intersection and reserve time and space for their traversal. Unfortunately, such a system is not cost-effective at small intersections, as it requires the installation of specialized infrastructure. In this paper, we propose an intersection control mechanism for autonomous vehicles designed specifically for low-traffic intersections where the previous system would not be practical, just as inexpensive stop signs are used at intersections that do not warrant a full traffic light installation. Our mechanism is based on purely peer-to-peer communication and thus requires no infrastructure at the intersection. We present experimental results demonstrating that our system, while not suited to large, busy intersections, can significantly outperform traditional stop signs at small intersections: vehicles spend less time waiting and consume less fuel.

1. INTRODUCTION

Recent advances in technology have made it possible to construct a fully autonomous, computer-controlled vehicle capable of navigating a closed obstacle course. The DARPA Urban Challenge [1], at the forefront of this research, aims to create a full-sized driverless car capable of navigating alongside human drivers in heavy urban traffic. It is feasible that, in the near future, many vehicles will be controlled without direct human involvement. Our current traffic control mechanisms, designed for human drivers, will be upgraded to more efficient mechanisms, taking advantage of cutting-edge research in the field of Multiagent Systems (MAS). Previously, we introduced an MAS-based traffic management system that has the potential to vastly outperform current traffic signals [4, 5]. In this system, vehicles negotiate with an agent stationed at the intersection, which grants each vehicle a specific time and space for its traversal. However, the high infrastructure costs associated with this system make it uneconomical at low-traffic intersections. For these situations, we propose a new control mechanism, based on peer-to-peer interaction, that requires no specialized infrastructure at the intersection.

1.1 A Managed Intersection Control Mechanism

Previously, we proposed an intersection control mechanism to direct autonomous agents safely through an intersection [5]. This system is based on the interaction of two classes of agents: *intersection managers* and *driver agents*. Driver agents "call ahead" to an intersection manager at the intersection, reserving the time and space needed to cross. Specifically, when approaching an intersection, a driver agent sends a request message containing a predicted arrival time and velocity, along with basic information about the vehicle it is controlling. The intersection manager responds with either a confirmation message containing details of the approved reservation, or a denial message, signaling that the parameters sent by the driver agent are unacceptable. In the case of confirmation, the driver agent will attempt to meet the parameters of the reservation, and will cancel the reservation if it cannot. In the case of denial, the driver agent must try to make a different reservation.

Intersection managers base their decisions on the supplied parameters and an *intersection control policy*. The most efficient policies, including FCFS or "first come, first served", simulate the trajectory of the vehicle through the intersection. At each stage in the simulation, the intersection manager checks whether the vehicle is within a certain buffer distance of any other vehicle in the intersection. If the requesting vehicle can cross the intersection without entering any space-time reserved by another vehicle, the policy creates the reservation, and the intersection manager approves

the request. Otherwise, the policy does not create a reservation, and the intersection manager denies the request. By integrating these policies with traditional traffic light systems, we have also demonstrated that the system can accommodate human traffic [6]. This multiagent approach offers substantial safety and efficiency benefits as compared to existing mechanisms, such as traffic lights and stop signs. Vehicles pass through the intersection faster, and congestion at intersections is significantly reduced.

Although at the city level this system is mostly decentralized, at each individual intersection, traffic is coordinated by a single arbiter agent, the intersection manager. We therefore designate this system a *managed* intersection control mechanism. An intersection controlled by a traffic light is also a managed intersection—the traffic light being the arbiter agent. Conversely, we designate intersection control mechanisms without an arbiter agent, such as stop signs and traffic circles, *unmanaged* intersection control mechanisms.

1.2 One Size Does Not Fit All

Managed intersection control mechanisms have a major drawback: cost. An arbiter agent of some sort must be stationed at the intersection, and our previously proposed managed system, this agent must have sufficient computational resources and communications bandwidth to rapidly negotiate a high volume of requests. Although the throughput benefits in large intersections would certainly warrant this expense, the system would be uneconomical for small intersections. Stop signs are a low-overhead, unmanaged system designed for low-traffic intersections, complementing larger intersections managed by traffic lights. In this paper, we propose an unmanaged intersection control mechanism for autonomous vehicles, designed specifically for low-traffic intersections. Our system—based on peer-to-peer communication and requiring no specialized infrastructure—is a similar complement to the managed intersection we previously proposed [5]. We make similar assumptions about the driver agent, such that a driver agent capable of using the managed system can be modified to use both systems seamlessly. We also present empirical data comparing our system to both traffic lights and stop signs. We focus our analysis primarily on the comparison between our system and the class of intersections that would currently be managed by a stop sign (low-traffic intersections), as these are the intersections for which our system is intended.

The remainder of this paper is organized as follows. In Section 2 we introduce the goals of our system, state our assumptions about the agents' world knowledge, and outline the protocol of our system. Section 3 describes the behavior of each individual driver agent. In Section 4, we present and discuss the empirical results of our system. Section 5, contains a discussion of current related work and presents some directions for further research. We summarize and conclude in Section 6.

2. AN UNMANAGED AUTONOMOUS INTERSECTION

To address the issue of high cost associated with managed autonomous intersections, we have created a low-cost alternative for low-traffic intersections. In this section, we introduce our unmanaged autonomous intersection control mechanism. First, we specify the goals of our system. Next, we describe our assumptions about the driver agents. We then outline the protocol for communication between vehicles, and describe the rules that each vehicle must follow.

2.1 Goals Of The System

For an unmanaged intersection control mechanism for autonomous vehicles to be both effective and economically viable, we believe it should have the following properties:

- Vehicles using the system should get through the intersection more quickly than they do using current mechanisms (i.e. stop signs).
- The protocol should have minimal (ideally none) per-intersection infrastructure costs.
- The protocol should guarantee the safety of the vehicles using it. Specifically, if all vehicles follow the protocol correctly, no collisions should result.

2.2 Assumptions

To safely navigate an intersection, a driver agent needs access to specific information: the layout and location of the intersection, any speed limits, and a variety of other parameters. As with our managed system, we assume that vehicles have access to this information either on board the vehicle or via a remote database. We assume that each vehicle is outfitted with a wireless communication device with sufficient range to communicate with other vehicles approaching the intersection. This range is approximately 200 meters in our scenario, but could vary based on the size of the intersection. We assume that this communication device has sufficient bandwidth to handle vehicle-to-vehicle communication, although our implementation relies on very small data packets, and we do not expect bandwidth to be a serious constraint. Finally, we assume that the latency of this device is sufficiently low. In our testing, we simulate a 20ms latency, but this is not a strict requirement of our system, as the parameters of the protocol can be adjusted to suit the environment (see Section 2.3).

In addition to these intersection-specific assumptions, we also assume that each vehicle has all the abilities required of autonomous open-road driving. These include access to a GPS-like navigation system that can provide an accurate and precise position, as well as laser range finders or short-wave radar capable of reliably sensing other vehicles in the immediate vicinity.

Finally, we assume that driver agents have access to information about the vehicle they are controlling, including its current velocity, position, and heading.

By analyzing the physical layout of the intersection, agents can determine which of the paths through it are *compatible*. That is, which paths can safely be followed simultaneously without the risk of a collision. For example, right turns from the rightmost lanes in any direction are always compatible, whereas any paths that intersect are not. Rather than having each agent independently find these paths, we assume that the list of compatible trajectories is part of the agent's knowledge of the intersection. Because driver agents may use this information to plan their trajectory through the intersection, possibly allowing two vehicles to cross simultaneously, it is important that each agent have the same notion of which paths are compatible.

2.3 Communication Protocol

Unlike the protocol for our managed intersection [3], our

protocol for unmanaged autonomous intersection control is designed for communication among only one type of agent: driver agents. In our system, each agent sends and receives information to and from each other agent, maintaining up-to-date information about every vehicle approaching the intersection. Dropped packets and limited transmission distance may cause agents to have outdated or inconsistent information. Because data transmission is largely asynchronous in an ad-hoc wireless network of mobile agents, this protocol cannot rely on a dialogue between agents. As such, the protocol is simple, consisting only of broadcast messages. There are two types of messages: CLAIM and CANCEL.

2.3.1 Claim

A CLAIM message is sent by an agent in order to announce its intentions to use a specific space and time in the intersection. CLAIM contains information describing both the vehicle's intended path through the intersection, as well as when it believes its traversal will take place. Once the agent has chosen these parameters, it broadcasts its CLAIM repeatedly. The message contains seven fields:

- `vehicle_id`—The vehicle's unique Vehicle Identification Number (VIN).
- `message_id`—A monotonically increasing counter specific to this message. Other agents will use `message_id` to identify the most recent message from this vehicle. This number is not changed when a specific message is rebroadcast; it is incremented only when a vehicle generates a *new* message to broadcast.
- `stopped_at_intersection`—A boolean value representing whether the vehicle is stopped at the intersection.
- `lane`—The lane in which the vehicle will be when it arrives at the intersection. Each lane incident to the intersection has an absolute index available as part of the intersection's layout information.
- `turn`—The direction in which this vehicle will turn.
- `arrival_time`—The time at which this vehicle will enter the intersection.
- `exit_time`—The time at which this vehicle will exit the intersection.

2.3.2 CANCEL

An agent sends a CANCEL message to release any currently held reservation. This message cancels any pending reservation; even if other agents have differing or outdated information about an agent's reservation, the agent can still cancel. The CANCEL message is broadcast repeatedly, with the same period as CLAIM, to ensure it is received by all other agents. This message has two fields:

- `vehicle_id`—This vehicle's VIN.
- `message_id`—A monotonically increasing number specific to this message. This is the same as the `message_id` field in CLAIM.

2.3.3 Message Broadcast

Because each message contains all the latest relevant information about the sending vehicle, agents need only pay attention to the most recent message from any other vehicle. Each message is also broadcast repeatedly with a set period to ensure its eventual delivery, should a new vehicle enter the transmission range of the sender. As a result, although occasional dropped messages may increase the delay in communications between vehicles, they should not pose a significant threat to the safety of vehicles in our system. In situations with higher rates of packet loss, messages may need to be broadcast more frequently to compensate. Conversely, in low-latency, high-reliability scenarios, messages can be sent less frequently.

For security purposes, we also assume that each message is digitally signed, ensuring that driver agents cannot falsify the `vehicle_id` parameter. Messages that do not conform to the protocol or are not digitally signed are ignored.

2.3.4 Conflict, Priority, and Dominance

In order to facilitate the discussion of agent behavior and protocol analysis, we define the following relations on CLAIM messages.

Two CLAIM messages are said to *conflict* if all of the following are true:

- The paths determined by the `lane` and `turn` parameters of the CLAIM messages are not compatible
- The time intervals specified in the CLAIM messages are not disjoint

We define the relative *priority* of two CLAIM messages based on the following rules, presented in order from most significant to least significant:

1. If neither CLAIM specifies that the sending vehicle is stopped at the intersection, the CLAIM with the earliest `exit_time` has priority.
2. If both CLAIM messages specify that the respective sending vehicles are stopped at the intersection, the CLAIM whose `lane` is "on the right" has priority. Here, "on the right" is defined similarly to current traffic laws regarding four-way stop signs. This binary relation on the incident lanes is globally available as a characteristic of the intersection.
3. If neither message's `lane` can be established as being "on the right," the CLAIM whose `turn` parameter indicates the sending vehicle is not turning has priority.
4. If priority cannot be established by the previous rules, the CLAIM with the lowest `vehicle_id` has priority.

.

Finally, given two claims c_1 and c_2, we say that c_1 *dominates* c_2 if either of the following rules is true:

- The `stopped_at_intersection` field of c_1 is `true` and the `stopped_at_intersection` field of c_2 is false.
- The `stopped_at_intersection` fields of c_1 and c_2 are identical, c_1 and c_2 conflict, and c_1 has priority over c_2.

2.4 Required Agent Actions

The consequences of failure in a traffic management system can be disastrous. As such, in addition to a communication protocol, a rigid set of rules must govern the interaction of agents within the system. With human drivers, traffic laws serve this purpose: if every driver obeys traffic laws, there is little or no potential for automobile accidents. Our multiagent system relies on an analogous set of rules. While there is nothing physically preventing an agent from ignoring them, the safety of each agent's vehicle can only be guaranteed if that agent follows the rules. Note that the rules restrict only how the agent behaves while in the intersection; driver agents have full autonomy everywhere else. The rules are as follows:

1. A vehicle may not enter the intersection if its own CLAIM is dominated by any other current CLAIM.
2. A vehicle may not enter the intersection without first broadcasting an CLAIM for at least T_p seconds. In our implementation, $T_p = .4$.
3. A vehicle must vacate the intersection at or before the `exit_time` specified in its most recent CLAIM message.
4. If a vehicle is going to traverse the intersection, it must follow a reasonable path from the point of entry to the point of departure. This means, for example, that a vehicle going straight through the intersection must remain within its lane, and that a vehicle turning right must not enter any other lanes.
5. The `stopped_at_intersection` field of an agent's CLAIM must be set to `true` if and only if the agent is stopped at the intersection.
6. The agent may not broadcast unless it is within a certain distance of the intersection. This distance is called the *lurk distance*. In our implementation, the lurk distance is 75 meters.

2.5 Selfish and Malicious Agents

Agents in our system are assumed to be self-interested—they may take any possible legal action in order to ensure they traverse the intersection in as little time possible. Agents have little incentive to lie about their lane, path, or exit time, because lying about any of these puts the vehicle at risk for collision. However, an agent may have an incentive to falsely claim that it is stopped at the intersection. While there is a chance this may slow down the traffic in front of the offending vehicle, if there is no such traffic exists, an agent may gain some advantage by falsely claiming that it is stopped at the intersection, allowing its CLAIM to dominate the CLAIMS of other moving vehicles. This may result in the vehicle crossing the intersection earlier. This type of behavior is not currently disincentivized by our protocol, but if it were to become a problem, could be tested at random intersections to ensure compliance. This is analogous to current traffic enforcement, which relies on sporadic monitoring and associated penalties to decrease rule violations.

As with any multiagent system, malicious agents are a potential problem. In current traffic scenarios, nothing prevents someone from deliberately crashing into another vehicle, or disabling traffic signals. Similarly, a malicious driver agent could flood the network with useless traffic, preventing the system from operating properly. While nothing can be done to stop a determined saboteur, the fact that all messages are signed makes it impossible for vehicles to conceal their identity while using the protocol.

3. DRIVER AGENT BEHAVIOR

Our proposed unmanaged intersection control mechanism relies not only on the communication protocol defined in Section 2.3, but also on the existence of driver agents that can abide by the protocol. Our prototype driver agent's behavior is comprised of three phases: lurking, making a reservation, and intersection traversal.

3.1 Lurking

As the vehicle approaches the intersection, it begins to receive messages from other agents. However, it may not broadcast a reservation until it is within the *lurk distance*. The lurk distance is calculated to ensure that an agent is within transmission range of other vehicles long enough to be reasonably sure that it is aware of every pending CLAIM. CLAIMS are broadcast repeatedly at a set frequency; more frequent broadcasts reduce the amount of time an agent must spend within transmission range to assemble all pending CLAIMS. Therefore, lurk distance depends on both transmission range and broadcast frequency. In our simulations, we set lurk distance to 75 meters—a reasonable approximation given current communication technology.

3.2 Making a Reservation

The most important part of our driver agent behavior starts when vehicle reaches the lurk distance. At this point, it needs to let the other driver agents know how it intends to cross the intersection. We call this part of the process "making a reservation," as an analogue to our managed system, which also uses a reservation paradigm [5]. During this time, the vehicle needs to compute its expected arrival time, arrival velocity, departure time, and given the messages it has accumulated from other vehicles, determine the soonest time at which the intersection will be available. This behavior is shown in Algorithm 1.

Algorithm 1 Behavior of the driver agent from coming within lurk distance of the intersection to entering the intersection.

```
 1: loop
 2:   if do not have a current CLAIM then
 3:     generate a new CLAIM
 4:   end if
 5:   if not at the intersection and another vehicle is then
 6:     broadcast CANCEL
 7:   else
 8:     if arriving estimate changes or CLAIM is dominated
        then
 9:       generate a new CLAIM
10:     end if
11:     broadcast the CLAIM
12:   end if
13: end loop
```

As an agent approaches the intersection, it generates a CLAIM based on predictions of its arrival time, arrival velocity, and path through the intersection (line 3). To predict the time required to cross the intersection, the agent must know its arrival velocity. Initially, the agent calculates the earliest possible arrival time, and the predicted velocity of the vehicle at this time based on the speed limit and its own acceleration constraints (the physical constraints of the vehicle, in addition to the constraints imposed by traffic front of it) Based on this arrival velocity, the agent predicts the time at which it will exit the intersection, assuming that it can accelerate as needed within the intersection. If the agent has received no CLAIMS from other vehicles that dominate this CLAIM, the agent will begin to broadcast this CLAIM (line 11).

Otherwise, the agent generates a new CLAIM at the earliest possible time such that it will not be dominated by any existing CLAIM of another vehicle (line 9). To do so, the agent searches through existing CLAIMS to find the next block of time that it could potentially dominate, assuming

it can arrive at the highest legal velocity. After finding a suitable block, the agent predicts its arrival velocity based on arrival time (which is generally lower than the maximum legal velocity), which it uses to determine the actual time required to cross the intersection. If the agent can traverse the intersection in the available time, it begins broadcasting a CLAIM; if not, it searches for the next suitable block and repeats these calculations.

3.3 Intersection Traversal

Once a vehicle has made a reservation, it needs only to broadcast the CLAIM continually and to arrive at the intersection in accordance with its reservation. However, sometimes the vehicle may want to change an existing claim in order to take advantage of an unexpected early arrival (line 8). On the other hand, traffic patterns may occasionally cause a vehicle to arrive late. If a vehicle predicts that it cannot fulfill the parameters of its CLAIM message, it must either send a CANCEL a new CLAIM. Similarly, if a new CLAIM message arrives that dominates the driver agent's CLAIM, the driver agent must also make a new reservation.

Once the vehicle reaches the intersection, it crosses in accordance with its CLAIM. While in the intersection, for safety purposes, the vehicle continues to broadcast its CLAIM, however this CLAIM cannot be dominated, as the vehicle is already executing the intersection traversal, which is clear from the fact that the current time is after the `arrival_time` in the CLAIM. After a vehicle has vacated the intersection, it stops transmitting its CLAIM.

3.3.1 Vehicle Control

The driving actions taken by a vehicle to complete its reservation are very similar to those of the driver agent in our managed mechanism [3]. If a vehicle predicts that it will arrive late, it accelerates. If a vehicle predicts that it will arrive early, it slows down (unless it believes it can make an earlier CLAIM). The vehicle must also ensure that it arrives with sufficient velocity to traverse the intersection within the constraints of its reservation.

3.3.2 Canceling "Bad" Reservations

In some situations, a vehicle is unable to reach the intersection at the proper time and velocity. To detect these situations, the vehicle is constantly predicting its arrival time. As with the driver agent presented in our work on managed intersections, this agent calculates its arrival time and velocity either *optimistically* or *pessimistically* [5]. If a vehicle detects no vehicles in front of it, it will make an optimistic projection of arrival time, assuming it can accelerate as needed before it arrives. However, if a vehicle is obstructed by traffic, it will make a pessimistic projection of arrival time based on the assumption that it cannot accelerate before it arrives at the intersection. If the vehicle's predicted arrival time is later than that of its reservation, the vehicle will cancel its current reservation and attempt to make a reservation for a later time.

3.3.3 Improving Reservations

If a driver agent predicts that it will arrive at the intersection before the time specified in its reservation, it may be able to improve its reservation before reaching the intersection. To accomplish this, the agent looks for blocks of intersection time between its predicted arrival time and the arrival time specified in its reservation. If the vehicle determines that it can broadcast a suitably large CLAIM that will not be dominated, it will immediately begin broadcasting this CLAIM. As specified by the communication protocol, this implicitly cancels any previous reservation held by the vehicle.

If a vehicle arrives at the intersection before the time specified in its reservation, it changes its CLAIM to reflect that it is stopped and waiting to cross (as required by the protocol). As a result, this agent's CLAIM will now dominate the CLAIM of any vehicle not stopped at the intersection. The stopped agent will then begin broadcasting the earliest possible non-dominated CLAIM. If no other vehicles are stopped, this will be T_p seconds from the current time, as the vehicle must broadcast its claim for at least this amount of time before entering the intersection. If other vehicles are stopped at the intersection, the agent will broadcast a CLAIM for the earliest block of time not dominated by the CLAIM of any stopped vehicles.

4. EMPIRICAL RESULTS

This section presents empirical results comparing our unmanaged autonomous intersection to intersections outfitted with four-way stop signs and traffic lights. After describing our metrics and experimental setup, we compare the average delay induced by each of these control policies. We then use these results estimate the amounts of traffic for which a stop sign outperforms a traffic light. This range is the primary focus of the analysis of our system, as we consider it to be the range over which an unmanaged policy is more appropriate than a managed policy. We also compare the relative fuel consumption associated with the stop sign and unmanaged autonomous policies. Finally, we discuss the effects of dropped messages on our unmanaged autonomous control policy.

4.1 Metrics

In our analysis, we examine two key metrics: *average delay* and *average cumulative acceleration*. The primary metric is the average of the delay experienced by each vehicle as it crosses the intersection. The baseline for delay is the time it would take a vehicle to traverse a completely empty intersection. Because a vehicle must slow down to turn, the baseline is different for left turns, right turns, and straight passages through the intersection. We measured the trip time for an unobstructed vehicle following these three paths, giving us an accurate baseline for comparison. Delay is measured as actual trip time minus baseline trip time, which isolates the effect of the intersection control policies and allows us to accurately compare the among them.

The second metric we use is the average of the the cumulative acceleration of each vehicle during its trip through the intersection. We define the *cumulative acceleration* of a vehicle, denoted $\overline{a}$, as:

$$\overline{a} = \sum_{i=0}^{s} |a_i|$$

where s is the trip length of the vehicle measured in simulator steps, and a_i is the acceleration of the vehicle at simulator step i. Note that the baseline for $\overline{a}$ is nonzero in turning vehicles, as vehicles must slow down to turn and accelerate again to the speed limit afterwards. We chose to

compare the average cumulative acceleration to examine the relative fuel efficiency of each system. Although not a direct measure of fuel efficiency, a vehicle's cumulative acceleration provides a reasonable approximation of gasoline usage, because substantially more fuel is required to accelerate than to maintain a constant velocity. Average delay is also an indicator of fuel efficiency, as the delay experienced by a vehicle correlates with the amount of fuel consumed while the vehicle was not accelerating (either idling at the intersection or traveling at a constant velocity). Thus, we can compare the relative fuel efficiency of each system by comparing both average delay and average cumulative acceleration.

4.2 Experimental Setup

To test these policies, we use a custom simulator which simulates a four-way intersection with one lane of traffic in each direction (see Figure 1). This small, symmetrical intersection is representative of those intersections currently configured as a four-way stop, and thus provides the best test case for unmanaged control mechanisms. We control traffic levels via a Poisson process governed by the probability of creating a new vehicle in a given lane at each time step. We simulate traffic levels between 0 and 0.5 vehicles per second, with 15% of vehicles turning left and 15% turning right. Each data point represents the average of 20 simulations, with each run consisting of 30 minutes of simulated time. All data are shown with error bars indicating a 95% confidence interval.

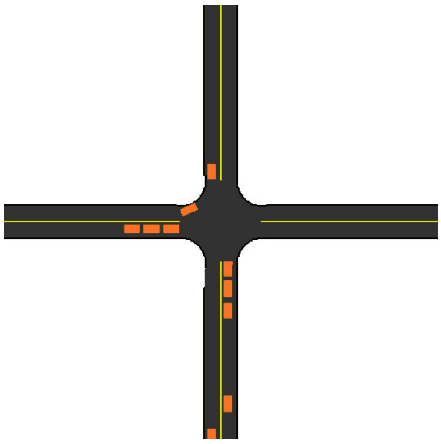


Figure 1: A screenshot of the simulator.

The traffic light timing is configured such that, in succession, each direction receives a green light for 10 seconds, followed by 3 seconds of yellow. There is a large body of theory and empirical evidence concerning the timing of traffic lights, but this work is largely irrelevant to our simulated scenario for two reasons. First, much of the theory deals with the timing of lights across multiple intersections, whereas we are examining one intersection in isolation. Second, our simulator generates symmetric traffic, which greatly simplifies light timing by eliminating the need to account for higher traffic levels in a particular direction or lane. For these reasons, we established a reasonable timing pattern experimentally by evaluating 10 different candidate patterns and selecting the one that led to the lowest average delay.

It should be noted that our four-way stop sign policy does not allow multiple vehicles to inhabit the intersection simultaneously. In the real world, stop signs can allow a limited sharing of the intersection. This is most apparent in intersections with multiple lanes of traffic in each direction: in this situation, cars traveling parallel to one another can cross the intersection at the same time. There is significantly less potential for sharing the intersection when there is only one lane of traffic in each direction. A human driver may observe the vehicle currently crossing the intersection and predict the vehicle's actions for the remainder of its journey (although this prediction is not always accurate!). If the other vehicle's path does not conflict with the intended path of the human driver, he or she may enter the intersection slightly before the other vehicle has exited. However, the benefits of this behavior are significantly reduced in small intersections. Therefore, we believe that our four-way stop sign policy is a reasonable approximation of a real-world four-way stop.

4.3 Delay

As shown in Figure 2, our system significantly reduces the average delay experienced by each vehicle. When traffic flow is below 0.35 vehicles per second, the four-way stop is a more effective policy than the traffic light. Because an unmanaged mechanism performs best over this domain, we consider [0,0.35] vehicles per second to be the target domain of our system.

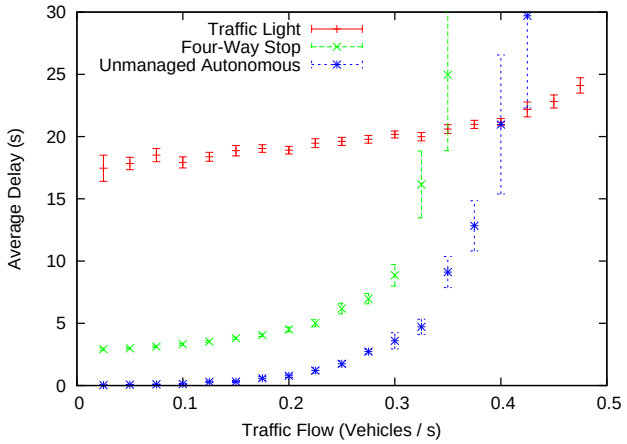


Figure 2: A comparison of average delay of the traffic light, four-way stop, and our unmanaged mechanism. The x-axis represents the traffic level, expressed in vehicles per second. The y-axis represents the average of each vehicle's delay, in seconds.

Our unmanaged system results in near-zero delay at traffic levels below 0.2 vehicles per second. In these situations, most agents are able to cross the intersection without slowing down to wait for other vehicles. With the four-way stop sign, each vehicle must stop even if no others are present, resulting in a baseline average delay of approximately 3 seconds. The traffic light system has a higher baseline average delay, around 18 seconds.

When traffic flow is between 0.2 and 0.35 vehicles per second, our system shows a somewhat increased delay. In these cases, cars must often slow down to accommodate other ve-

hicles, but but only rarely will a vehicle need to make a complete stop. With the stop sign policy, vehicles begin to queue at the intersection, and must often wait for vehicles in front of them to cross. The traffic light policy shows almost no increase in delay at these levels.

At traffic levels above 0.35 vehicles per second, the stop sign policy deadlocks. At these traffic levels, our system is similar to a four-way stop: because there is almost always at least one vehicle waiting to cross, agents must wait until they are stopped at the intersection to make a reservation (as described in Section 2.4). However, the intersection sharing in our system (allowing four simultaneous right turns, for example) provides a noticeable benefit at these traffic levels. Our unmanaged system can safely handle traffic levels up to approximately 0.4 vehicles per second, at which point traffic begins to back up. The traffic light shows only a slight increase in delay at these traffic levels. In these situations, our data suggest that a managed mechanism is more appropriate.

4.4 Average Acceleration

Another benefit of our system is reduced average acceleration, as shown in Figure 3. With the stop sign policy, every vehicle must come to a complete stop at the intersection and accelerate to the speed limit after crossing. If vehicles are queued at the intersection, each vehicle must stop at the back of the queue. As the queue moves forward, each vehicle accelerates for a brief period of time, then decelerates to a stop until another car leaves the front of the queue. This behavior results in a very high average acceleration for the stop sign policy.

For low levels of traffic, our system allows most vehicles to pass directly through the intersection without slowing or stopping. Even at high traffic levels, when our system is essentially a modified four-way stop, our system results in lower average acceleration than a four-way stop. This is because our system causes shorter queues than a stop sign, reducing the amount of acceleration and braking required for each vehicle to reach the front of the queue. Combined with the data on average delay, these results suggest that our unmanaged autonomous system would allow significantly reduced fuel consumption.

4.5 Dropped Messages

We designed our system to be resistant to occasional communication failures such as dropped messages. In our previously proposed managed intersection, the vehicles must wait for a response from the intersection manager before entering the intersection [5]. Because of this, dropped packets may increase the delay of the system, but will not cause a collision. In our system, we have found no statistically significant correlation between dropped packets and delay. Rather, dropped packets introduce a possibility of failure that increases with the percentage of packets dropped.

To quantify this effect, we varied the proportion of dropped messages between 0 and 0.7 at intervals of 0.1, running 400 thirty-minute simulations at each level. The traffic level in these simulations was 0.3 vehicles per second. When fewer than 40% of messages were dropped, the system behaved normally. Between 40% and 60% packet loss, the system began to experience safety failures—five of the 1200 simulations in this range resulted in collisions. At 70% packet loss, the frequency of collisions is significantly higher, with

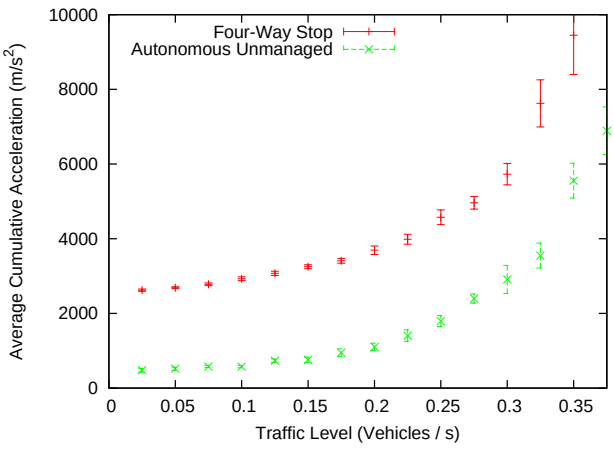


Figure 3: A comparison of average acceleration of the four-way stop and our unmanaged mechanism. The x-axis represents the traffic level, expressed in vehicles per second. The y-axis represents the average of each vehicle's cumulative acceleration, in meters per second per second.

collisions occurring in seven of 200 simulations.

These results suggest that, as proposed, our peer-to-peer protocol can tolerate moderate levels of packet loss with no ill effects, but that serious communication issues might make it unsafe. While a thorough analysis of communication failures is beyond the scope of this paper, research in distributed systems has shown that fast and reliable information dissemination in ad-hoc wireless networks such as the kind we are simulating is possible [2]. We thus leave further communication analysis to future work.

5. DISCUSSION AND RELATED WORK

We have presented a system which allows autonomous agents to coordinate their safe passage through an intersection without an intersection manager, and demonstrated that it outperforms the current mechanisms for both managed and unmanaged intersections over its target traffic levels. We have specified a detailed protocol meeting the constraints of vehicle-to-vehicle communication, which adds few assumptions on top of those in the managed autonomous intersection. Because of this, it would be easy to create a driver agent that can utilize both managed and unmanaged intersections. As this driver agent approaches the intersection, it determines whether the intersection is managed using previous experience or, if the agent has never encountered the intersection, by attempting to communicate with the intersection manager. If the agent receives a response, it uses the appropriate managed intersection protocol; if not, it uses our unmanaged intersection protocol.

5.1 Future Work

After introducing the reservation-based protocol for managed intersections based on the assumption that all cars are autonomous, we later presented a policy which allows both computer- and human-controlled vehicles to safely interact at the same intersection [6]. Our protocol for unmanaged intersections can be similarly adapted to accommodate hu-

man drivers using traffic signs. The human drivers would be directed to behave as if they were stopped at a two-way stop, yielding to all approaching vehicles (this also assumes that the computer-controlled vehicles have some signal identifying them as autonomous). Because our system is designed for low-traffic intersections, human drivers could generally expect to wait for no more than a few seconds. Our proposed system for accommodating human drivers and the corresponding managed system both put human-controlled vehicles at somewhat of a disadvantage–an incentive for human drivers to transition to fully computer-controlled vehicles. Future research could formalize and optimize a policy for accommodating human drivers in our unmanaged autonomous intersection.

Another potential area for future research is allowing the system to adapt to asymmetric traffic flow. Many intersections consistently receive higher traffic in some lanes than others. In these intersections, a two-way stop is often more efficient than a four-way stop. In our current system, all agents stopped at the intersection are given equal priority, regardless of the number of vehicles queued behind them. This approximates the behavior at a four-way stop. However, by granting priority to lanes with longer queues, our system could alleviate congestion in high-traffic lanes. This would allow our system to function like a two-way stop in situations with asymmetric traffic flow, while functioning like a four-way stop in situations with more symmetrical traffic.

5.2 Related Work

Intersection management—especially for intersections of autonomous vehicles—is an exciting and promising area of research for autonomous agents and multiagent systems. Many projects in AI and intelligent transportation systems address this increasingly important problem. Using techniques from computer networking, Naumann and Rasche created an algorithm in which drivers attempt to obtain *tokens* for contested parts of the intersection, without which they cannot cross [8]. While this allows vehicles to cross unimpeded in very light traffic, the system has no notion of "planning ahead"; only one vehicle may hold a token at any given time, no agent can plan to have the token in the future if another agent has it currently. Kolodko and Vlacic have created a system very similar to ours on golf cart–like IMARA vehicles [7]. However, their system requires all vehicles to come to a stop, irrespective of traffic conditions.

In the context of video games and animation, Reynolds has developed autonomous steering algorithms that attempt to avoid collisions in intersections that do not have any signaling mechanisms [9]. While such a system does have the enormous advantage of not requiring any special infrastructure or agent at the intersection, it has two fatal drawbacks that make it unsuitable for use with real-life traffic. First, the algorithm does not let driver agents choose which path they will take out of the intersection; a vehicle may even find itself exiting the intersection the same way it came in, due to efforts to avoid colliding with other vehicles. Second, the algorithm only *attempts* to avoid collisions—it does not make any guarantees about safety.

6. CONCLUSION

Recent research has already produced fully autonomous, computer-controlled vehicles. As these vehicles become more common, we will be able to phase out human-centric traffic control mechanisms in favor of vastly more efficient computer-controlled systems. This will be especially beneficial at intersections, which are a major cause of delays. For a transition of this magnitude, infrastructure cost will be a central, if not primary, concern. This paper presents a novel, unmanaged intersection control mechanism requiring no specialized infrastructure at the intersection. We have described in detail a protocol for our unmanaged autonomous intersection, and created a prototype driver agent capable of utilizing this protocol. As illustrated by our empirical results, our protocol can significantly reduce both delay and fuel consumption as compared to a four-way stop. Unsignalized intersections far outnumber those that are sufficiently large or busy to warrant the cost of a managed solution. Whereas busier intersections may need to wait for the funding and installation of requisite infrastructure, our proposed mechanism has the potential to open every one of these unsignalized intersections to be used safely and efficiently by autonomous vehicles.

7. REFERENCES

[1] The DARPA grand challenge. `http://www.darpa.mil/grandchallenge`.

[2] V. Drabkin, R. Friedman, G. Kliot, and M. Segal. Rapid: Reliable probabilistic dissemination in wireless ad-hoc networks. In *The 26th IEEE International Symposium on Reliable Distributed Systems*, Beijing, China, October 2007.

[3] K. Dresner and P. Stone. Multiagent traffic management: A protocol for defining intersection control policies. Technical Report UT-AI-TR-04-315, The University of Texas at Austin, Department of Computer Sciences, AI Laboratory, December 2004.

[4] K. Dresner and P. Stone. Multiagent traffic management: A reservation-based intersection control mechanism. In *The Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 530–537, New York, NY, USA, July 2004.

[5] K. Dresner and P. Stone. Multiagent traffic management: An improved intersection control mechanism. In *The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 471–477, Utrecht, The Netherlands, July 2005.

[6] K. Dresner and P. Stone. Sharing the road: Autonomous vehicles meet human drivers. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, pages 1263–68, Hyderabad, India, January 2007.

[7] J. Kolodko and L. Vlacic. Cooperative autonomous driving at the intelligent control systems laboratory. *IEEE Intelligent Systems*, 18(4):8–11, July/August 2003.

[8] R. Naumann and R. Rasche. Intersection collision avoidance by means of decentralized security and communication management of autonomous vehicles. In *Proceedings of the 30th ISATA - ATT/IST Conference*, 1997.

[9] C. W. Reynolds. Steering behaviors for autonomous characters. In *Proceedings of the Game Developers Conference*, pages 763–782, 1999.

Towards a Reliable Air Traffic Control[1]

Minh Nguyen-Duc, Zahia Guessoum
Computer Science Laboratory
Paris 6 University
104, av. Président Kennedy
75016 Paris, France
{minh.nguyen-duc, zahia.guessoum}@lip6.fr

Olivier Marin, Jean-François Perrot, Jean-Pierre Briot
Computer Science Laboratory
Paris 6 University
104, av. du Président Kennedy
75016 Paris, France
{olivier.marin, jean-françois.perrot, jean-pierre.briot}@lip6.fr

Vu Duong
Eurocontrol Experimental Centre
Centre du Bois des Bordes B.P. 15
F-91222 Brétigny-sur-Orge
France
vu.duong@eurocontrol.int

ABSTRACT
Since critical socio-technical systems include people interacting with equipments in workplaces, their intrinsic reliability problems have been concerned with both these two "actors". *Air Traffic Control* (ATC) is going to be such a system in which controllers use a large number of distributed software tools to provide safety ATC services. The reliability of these services relies on the availability of the various tools. Indeed, a partial failure of a tool in use can have tragic consequences. This paper presents a multi-agent approach to this problem. We propose an agent-based decision-aided system that helps controllers in using their multiple software tools in situations where some tools are not available due to technical incidents. We build and test our system in an ATC simulation environment, thus develop an *Agent-Based Simulation* (ABS). Experimental work has demonstrated the significance of our system to air traffic controllers.

Keywords
Agent-based decision-aided system, Reliability, Socio-technical system, *Air Traffic Control*.

1. INTRODUCTION
Air Traffic Control (ATC) provides services whose objective is to direct aircraft on the ground and in the air. Its tasks are to separate aircraft (keep an aircraft in a minimum distance from another aircraft), to ensure safe orderly and expeditious flow of traffic, and to give information to pilots, such as weather and navigation information.

1.1 A critical socio-technical system
The forecast growth in air traffic requires the adoption of new technologies and related procedures enabling the safe and efficient provision of ATC services to a larger number of aircraft. This will be made possible by the use of software tools to support air traffic controllers (see the *First ATC Support Tools Implementation* (FASTI) program [18]).

Our work is concerned with the next generation of software systems for ATC. These systems will process some advanced flight data [7], which will be much more complicated than the currently used data. This will increase the controllers' capacity at the expense of a complexification of their task, but also will raise the technical issue of reliability.

On the "human" side, moreover, the integration of sophisticated tools in the controllers' daily work currently faces difficulties, like in any critical socio-technical system [9][16][20][21][23][28]. On the one hand, controllers have to change their usual, trusted working procedures. The safety and power that the tools are expected to provide will only become effective if the controllers are able to make the most of the functionalities of their tools. And this strongly depends on their familiarity with the tools. On the other hand, the controllers need to feel confident in the reliability of their software tools. In this paper, we report on an experiment with a *Multi-Agent System* (MAS) which aims at building confidence for air traffic controllers.

1.2 A multi-agent approach
We argue that the best way to prove a support system's reliability is to show that the ATC system, as a whole, can still provide full traffic control services when errors suddenly appear. Indeed, if the controllers are timely and adequately informed of the incidents, they can accordingly adjust their current control tasks and the following tasks. They can often manage without some of their tools. According to the *Guidance Material for Contingency Planning* [6], this kind of working mode can be seen as a type of *Degraded Mode of Operation*.

Therefore, there exists a need for a decision-aided system that helps the controllers in using their multiple software tools, particularly in situations where some tools are not available, or in other words when technical incidents happen. Our aim is to show that a suitable use of multi-agent technology can help in this respect. To develop such a system, we propose a solution based on software agents (see Section 3).

1.3 Outline of the paper
The paper is organized as follows. Section 2 presents the ATC system and analyzes a typical example of technical incident.

[1]This is an improved version of a paper with the same title which was accepted as a short paper at AAMAS'08 Industrial Track.

Section 3 proposes our MAS solution. Section 4 describes the development of our ABS for experiments. Then in Section 5, an experimental scenario clearly shows how our agents react to a typical network failure. Section 6 summaries feedback from ATC experts on our MAS, which has been gathered in several demonstration sessions. Section 7 discusses related work. Finally, Section 8 draws a conclusion.

2. TYPICAL EXAMPLE

2.1 Future ATC system architecture

The current ATC system is airspace-based. The airspace is divided into many sectors whose size depends on the average traffic volume and the geometry of air routes. There are usually two air traffic controllers to handle the traffic in each air sector: an executive controller who communicates with pilots, and a planning controller who plans his colleague's work. Also, the sectors are regrouped into regions each of which is under control of a control center. For example, the Athis-Mons center is responsible for air traffic control in the Parisian region.

The general structure of the ATC system sketched above would not be expected to change. But a new architecture would have to support the introduction of distributed software tools. The system will be distributed over local area networks (LANs) in each control center and the wide area network (WAN) between centers. Figure 1 illustrates a typical application context where two different control centers are connected with a common flight data-processing center through the inter-center network (a WAN). In each control center one (or several) application server(s) host(s) the various software tools in use. These application servers are connected with the *Controller Working Positions* (CWP) by means of a local network (LAN). The LANs of the control centers are connected via the inter-center WAN. Each tool is thus at the same time exchanging data with the common flight data-processing centre and interacting with the controller user interfaces of the different controllers in the same control centre.

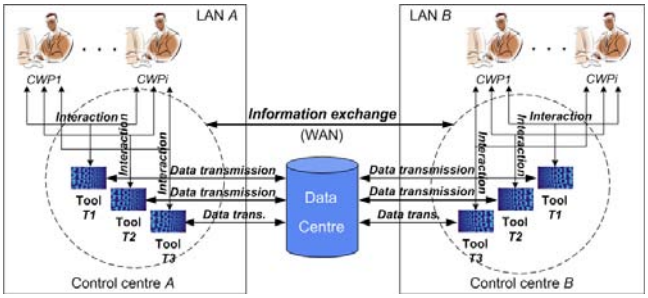


Figure 1. Basic future ATC system architecture.

A typical example of support tool is the *Medium-Term Conflict Detection* (MTCD) [18]. Once aircraft trajectories have been predicted, they can be employed to detect medium-term conflicts. There also exist many other tools such as *Short-Term Conflict Alert* (STCA), *MONitoring Aid* (MONA), *Airspace Penetration Warning* (APW), and *Minimum Safe Altitude Warning* (MSAW).

2.2 Scenario

In this section, we would like to analyze a typical situation in which a technical incident occurs. It can help with understanding the influence of the incident on the controllers' work and what they need in such a situation.

We hence consider two (executive) controllers (named Co_1 and Co_2) responsible for two neighboring sectors (named S_{10} and S_{12}), at the border of two control centers (named *A* and *B*). They are often in handover situations, *i.e.* they have to transfer the control of aircraft flying from one sector to the other (and therefore from the responsibility of one control center to the other center).

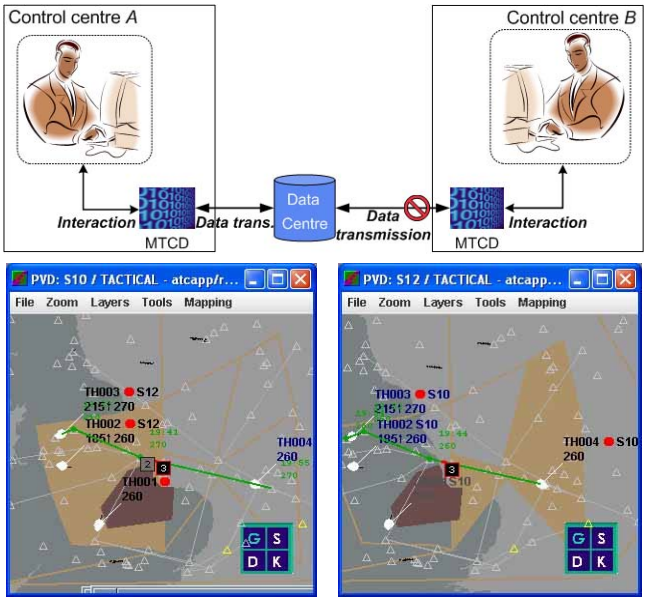


Figure 2. Typical example of a handover situation: the two controller's screens on both sides of the border.

We suppose that at a moment there are several potential conflicts among which a particular one concerns two aircraft: *TH003* flying from S_{10} to S_{12}, and *TH004* flying in the opposite direction. Moreover, all the conflicts are going to happen in S_{10}.

These conflicts can be automatically detected by Co_1's MTCD or "manually" by Co_1 himself. He does or does not perform control operations to resolve a detected conflict, depending on the aircraft real trajectories which probably evolve before the conflict happens. Since Co_1 can only resolve conflicts one by one, he has to sequence all the detected conflicts to be resolved. Therefore, he needs to decide to (or not to) resolve a conflict at least *T* minutes before it happens. Indeed, *T* is common to all the detected conflicts, and it has to be sufficiently large that the controller can perform good conflict sequencing.

We suppose that a network failure occurs at the time when the potential conflicts appear: center *B* is disconnected from the flight data-processing center (see the basic ATC system

architected illustrated by Figure 1). Consequently, a demand for exit flight level change for *TH004* sent by Co_2 to the data-processing center is lost. Accordingly, the flight data of *TH004* are no longer accessible from center *A* and therefore unusable for Co_1's MTCD.

This failure makes Co_1's MTCD unable to detect conflicts not only for *TH004*, but also for all the aircraft flying from center *B*. However, it still correctly detects the "local conflicts" that only concern the aircraft flying in Co_1. So we can see it as "locally available".

We now consider the controller's (Co_1's) reaction to the unavailability of his MTCD. It is supposed that there exists a fault detection equipment [1] which will give him some warning. We are thus interested in when he is informed of the tool unavailability and in which additional information he gets. We would like to underline the two following cases.

In the first case, Co_1 is only aware of the unavailability of his MTCD when some potential conflicts are closely going to happen (in less than *T* minutes). As discussed above, this will embarrass his conflict sequencing, and can then make him nervous.

In the second case, Co_1 is already aware of the unavailability of his MTCD but does not know its "local availability". He has to detect himself all the potential conflicts. To do that, he verifies and follows all the aircraft he suspects. However in reality, MTCD is not totally unavailable, and still locally available. Such an exhaustive verification will unnecessarily increase Co_1's workload because, in fact, he can still rely on the results given by MTCD for the local conflicts.

In conclusion, we notice that, in situations where some tools are not available, the controllers need not only to be timely informed of the unavailability of the tools, but also to obtain adequate information about their state, *e.g.* the "local availability" of MTCD. One could see that if the controllers are guaranteed to be provided what they require to manage without the unavailable tools, they will feel more confident on the reliability of the support system.

3. OUR MULTI-AGENT SYSTEM

3.1 Objectives

To fulfill the need presented in the previous section, a decision-aided system for air traffic controllers is needed. Its missions are to communicate with the controllers, to inform them of the environment state and to show them information of tools' availability. More ambitiously, the decision-aided system would be endowed with the capacity to propose corrective actions to be performed following technical incidents.

Besides, this system helps with mitigating the effects of software faults in a distributed environment. It monitors software components which run on different machines, and keeps an eye on the interactions between the users (*i.e.* the controllers) and these components. To this end, it also has to be distributed. It observes complex data (*e.g.* air traffic data) at the input and output of each computation module of any software tool.

More importantly, this system builds up confidence for users of a safety-critical software system. In consequence, it has to guarantee a safety level with respect to the services it offers. All its monitoring services have to run in real-time so that it can inform the users of some change of the software system's state as soon as it happens. Moreover, information it provides need to be not only concise but also adequate, in such a way that the users can determine exactly what to do in response to this change.

In view of these requirements of the decision-aided system to be developed, we propose a MAS solution. Our MAS communicates with the controllers through assistant agents and monitor the software tools through monitor agents. The capacity of the agents to exchange data with each other will allow acquiring in real-time information to be presented to the controllers.

3.2 Agent design

Since our agents have to take care of the monitoring of software tools and of the communication with the controllers, we design different kinds of agents to perform these two common tasks. We currently use three monitoring agents for each tool, *i.e.* a data sentinel, a middleware sentinel and a computation sentinel, and assign an assistant agent to each controller.

1 *Data sentinel agent*: observes the input and output data of a specific software tool and communicates with other agents in order to discover data losses; for example, as shown in the experimental scenario below, data sentinels ubiquitously check exchanged data (the absence of needed data at some network node often results from data losses).

2 *Computation sentinel agent*: observes the input and output data of a specific software tool and communicates with other agents in order to discover computation faults; for instance, a computation sentinel checks timeout errors of a computation module of a tool.

3 *Middleware sentinel agent*: receives from the middleware the notifications of faults related to a specific software tool (this also means that the monitoring agents do not employ any sophisticated fault detection technique [10][12][24]).

4 *Assistant agent*: communicates with other agents in order to determine the automated tools' availability, and informs a controller of this availability; an assistant agent can observe the controller's actions in such a way that it can notify the monitoring agents of relevant events.

At the individual level, all the agents presented above need not to be complicated. A monitoring agent simply reacts to technical incidents that it discovers itself or of which it is notified by other monitoring agents. An assistant agent would only be endowed with some limited reasoning capacity to be able to propose corrective actions to perform (which are predefined) following incidents. The simplicity of the agents

would bring not only more reactivity, but also more robustness to the MAS.

3.3 Distribution of our MAS

Based on the basic ATC system architecture presented in Section 2, we install in a control center a group of coordinated agents that are distributed over the whole corresponding LAN. The agents on the various LANs also communicate over the inter-center WAN, making up a global MAS.

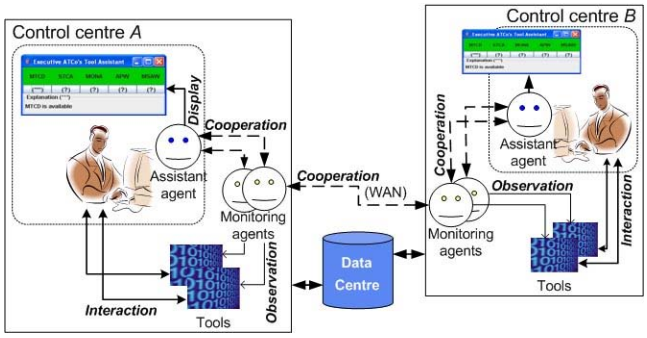


Figure 3. Monitoring and assistant agents.

Each local group of agents is composed of assistant agents and of monitoring agents. We associate an assistant agent with each *Controller Working Position* (CWP). Each tool instance is observed by monitoring agents. Please note that monitoring and assistant agents may be hosted on any of the machines, or even on additional independent network nodes, as long as they retain the capacity to display information on the controller's screen. At the level of implementation, only special agents playing the role of interface between our MAS and the whole tool system (see 4.3) need to be hosted on the same machines as the tools to which they connect.

When an incident occurs, the related tool's monitoring agent first discovers the critical situation by using the data it gathers from the tool's input/output, as well as the information it receives from other monitoring or assistant agents. Then, it transmits information about the tool's state to the assistant agents of the CWPs that use this tool. These assistants display green/yellow/red flags on their controller's screen, thereby indicating the tool's total/partial availability, together with the relevant information.

By exchanging observed tools' data and controllers' actions with each other, the agents can inform the controllers about the possible actions to take after the incident. For instance, in the scenario presented in Section 5, the monitoring agents exchange events of request for data change so that they can find out lost data due to network failures. Then, when the controllers see this information shown by their assistants, they know that the related aircraft's flight plan is inconsistent and that they cannot use MTCD anymore to detect conflicts for it.

4. SIMULATING OUR SYSTEM

Of course, the future ATC system we have described above has not yet been implemented. Moreover, any novel application to a critical system like ATC has to be tested in simulations before its real world implementation. We hence build and test our MAS into a simulation environment, thus develop an *Agent-Based Simulation* (ABS). We employ the eDEP platform (*Early Demonstration & Evaluation Platform*) [5], which offers not only realistic air traffic data but also a distributed simulated ATC environment. We also use the DimaX platform [2], which helps with developing reliable MASs.

4.1 ATC simulation platform - eDEP

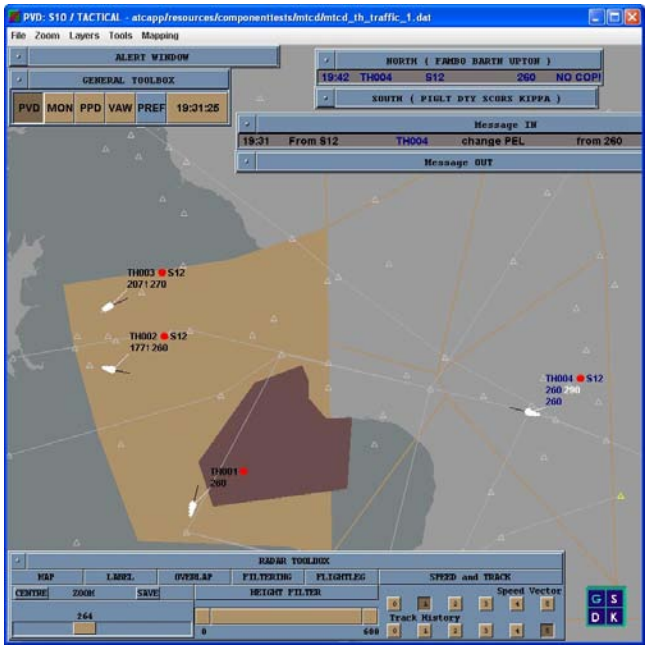


Figure 4. CWP for the executive controller of the sector *S10* (*S10* is the sector in light brown; white radar tracks mark aircraft; and red dots indicate aircraft potentially in conflict).

eDEP [4] is a Java platform developed by Eurocontrol that uses RMI (*Remote Method Invocation*) to distribute its components over a LAN. It gives a set of standard ATC elements, *e.g.*:

1 *Airspace*: a database of static airspace information;

2 *Integrated air surveillance*: a database of surveillance radar tracks;

3 *Flight Path Monitor*: uses track data produced by the IAS to monitor the progress of a flight according to its planned path through the airspace;

4 *Initial flight plan*: an initial plan that defines route constraint points and altitude limits;

5 *Trajectory predictor*: a trajectory prediction algorithm which uses aircrafts' kinematic models to predict the real motion of a particular aircraft;

6 *Controller Working Position* (CWP): the main graphical interface to the system based on a plane view display of the control sector (see Figure 4).

The support tools for air traffic controllers, *e.g.* STCA and MTCD, are implemented in eDEP as independent components which can run on different machines.

4.2 Multi-agent platform – DimaX

DimaX [2] is a Java multi-agent platform which provides a generic and modular agent architecture, and allows high heterogeneity in agent types (reactive, deliberative and hybrid). It is in fact based on the extension of modeling and implementation facilities offered by object-oriented languages. In DimaX, an agent at the smallest granularity is simply a single-threaded object, and a complicated agent can be constituted by smaller agents. Also, this platform allows adding new behaviors to any agent by using programming libraries.

In addition, DimaX provides a *Naming Service* which localizes agents at the time of message sending. A name server maintains the list (*i.e.* white pages) of all the agents within its administration domain. When an agent requests interacting with the others, it does not need to know their physical locations. Given the agent identifiers, the name server returns the corresponding agents physical addresses.

Since we would like our MAS to be used in a critical socio-technical system like ATC, the MAS itself has to be reliable. DimaX can help with developing such MAS. This multi-agent platform is in fact the result of the integration of its previous generation (named DIMA – *Development and Implementation of MAs*) and a fault-tolerance framework (named DarX [17]), which brings in services, *e.g. Fault Detection Service* and *Replication Service*, which provides transparent support for making MAS fault-tolerant through adaptive replication.

4.3 Implementing our ABS

We manage at least two *Controller Working Positions* (implemented by the CWP component in eDEP), belonging to two different control centers. The LAN of each control center is realized on at least two computers (one for the CWP and the other for the application server). The data-processing center is realized as a separate machine. This machine together with the two LANs make up our image of the inter-center WAN. Each application server runs a copy of each of five tools, *i.e.* MTCD, STCA, MONA, APW, MSAW (also provided as eDEP components).

The integration of our DimaX agents and eDEP components follows the FIPA *Agent Software Integration Specification* [8]. The DimaX platform already includes a generic wrapper agent ready to provide any other agent (*e.g.* a monitoring agent or an assistant agent) with services which allow this latter agent to connect to software components. Special wrappers are then built by extending the generic one. They need to be hosted on the same machine as the components they "wrap".

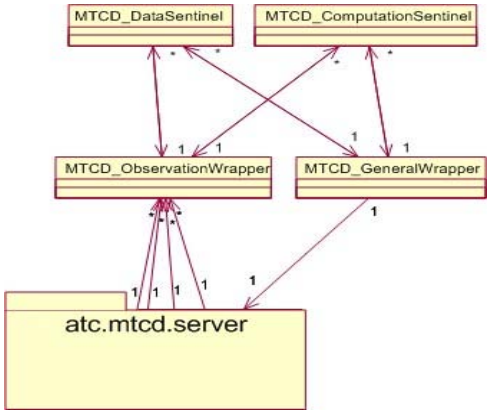


Figure 5. Monitoring and wrapper agents for MTCD (partial UML diagram).

Based on the agent model discussed in 3.2, as a first step we install three monitoring agents and two wrapper agents for each of software tools:

1 `XXX_DataSentinel`, `XXX_ComputationSentinel`, `XXX_MiddlewareSentinel`: observes the XXX[2] component's input/output data, communicates with other agents and collects notifications from the middleware in order to discover faults;

2 `XXX_ObservationWrapper`, `XXX_GeneralWrapper`: special wrappers which respectively provide XXX observation and general-purpose services to the three other XXX_agents;

We endow the CWP with a `CWP_Assistant` which communicates with other agents in order to determine the automated tools' availability, and shows this availability in its user interface. The following figure illustrates the `CWP_Assistant`'s user interface. It uses green/yellow/red flags to show the status of the various tools that run on the LAN. Additionally, it displays a few lines of explanation about the tool that is marked with (***).

Figure 6. `CWP_Assistant`'s user interface.

The `CWP_Assistant` observes the controller's actions through observation services provided by a `CWP_InterfaceWrapper`, which monitors the CWP's GUI in the same way as the monitoring agents access to the software tools through their observation wrappers.

[2] XXX stands for the tool name, *e.g.* MTCD or STCA.

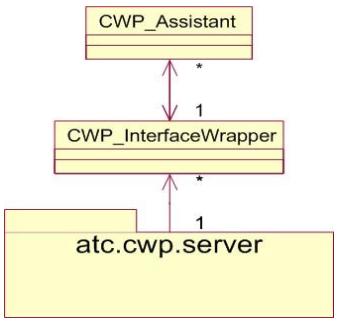


Figure 7. `CWP_Assistant` and wrapper agents (partial UML diagram).

5. AN EXPERIMENTAL SCENARIO

5.1 Objective

The first tests of our agents on the ABS use several experimental scenarios one of which corresponds to the example presented in Section 2. In this section, we will describe in detail this scenario which will show how the assistant and monitoring agents behave in situations where a typical fault occurs within the support system. This experiment also aims at demonstrating the usefulness of our MAS for air traffic controllers.

Precisely speaking, this scenario illustrates the reaction of our MAS to the possible unavailability of a tool due to a failure of the connection between a control centre and the common flight data-processing centre.

5.2 Experimental setup

The experiment runs on the following connected machines:

1. two client machines hosting two CWPs for two controllers belonging to two different control centers (named centers *A* and *B*);
2. two tool servers hosting two MTCD instances for the two control centers;
3. a data server placed in the common flight data-processing center.

5.3 Event sequence

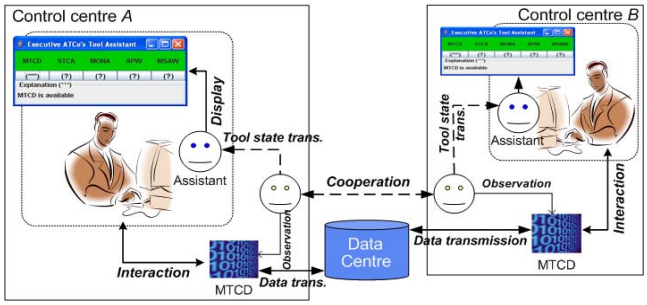


Figure 8. All machines run smoothly and are fully connected in a handover situation.

We are in a handover situation (as described in 2.2): there are aircraft flying from control center *B* to control center *A*. At first, all machines run smoothly and are fully connected. Each controller has unlimited access to the tool server on his LAN and can freely obtain the flight data he needs. The assistant agents display green labels indicating that the software tools are working at full capacity (see Figure 8).

The controller in center *B* (called *CB*) then makes a flight data change request (*e.g.* a demand for exit flight level change for an outgoing aircraft). However, due to some accident, control center *B* has been disconnected from the flight data-processing center. Due to the disconnection, this request is not sent to the data center.

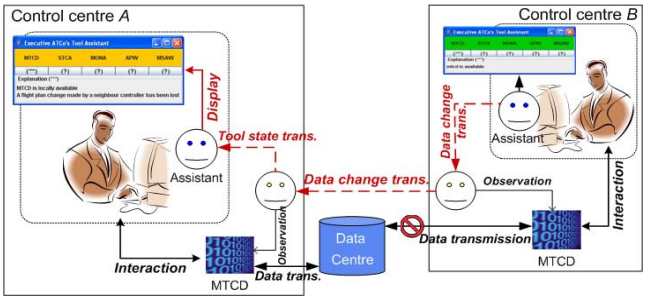


Figure 9. Control center *B* is disconnected from the flight data-processing center (1st phase).

Now, CB's assistant agent detects that a data change request was issued by *CB*. It notifies the data sentinel agent of MTCD in *B* of this request. This agent in its turn informs the monitoring and assistant agents in control center *A* through their simulated WAN connection.

The data sentinel agent of MTCD in *A* discovers that no such flight data change was received from the data-processing center. This also means that the flight data concerning an aircraft which is controlled by center *B* are no longer accessible from *A* and therefore unusable for conflict detection.

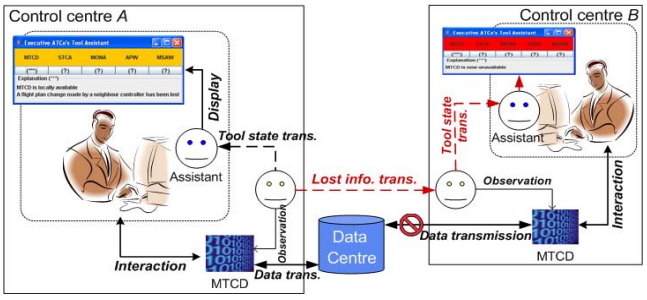


Figure 10. Control center B is disconnected from the flight data-processing center (2nd phase).

In consequence, the assistant agent of the controller in center *A* displays a yellow flag, informing his controller that MTCD is

only available locally, *i.e*. it only gives correct results for aircraft under control of center *A* (see Figure 9).

Knowing this, the data sentinel agent of MTCD in control centre *A* signals back to the monitoring agents in *B* that there was on its side a flight data change request which was not taken into account. This agent notifies the *CB*'s assistant agent of this incident.

Finally, the *CB*'s assistant agent then displays a red flag, informing his controller that MTCD is now unavailable (see Figure 10).

6. VALIDATION

6.1 Technical incidents

We have used our ABS to demonstrate typical scenarios to experts including both researchers in the field of ATC and professional air traffic controllers. These scenarios showed the reaction of our system to various technical incidents, each of which belongs to one of the following categories:

1 *Network failure*: the instant unavailability of tools due to a failure of the WAN connection between centers (like in the scenario presented above);

2 *Timeout error*: the instant unavailability of tools due to an unexpected timeout error of a tool's elementary computation module;

3 *Machine failure*: the instant unavailability of tools due to successive failures of tool servers.

6.2 Feedback

The first series of demonstration sessions was addressed to researchers from Eurocontrol (*European Organisation for the Safety of Air Navigation*). They questioned the kind of information exchanged by agents. They particularly sought to understand the advantages of this exchange in comparison to a simple duplication of lost data. Furthermore, they stressed the need of the involvement of operational points of view in the validation process.

The second series of experiments was therefore concerned with professional controllers. They firstly concentrated on the `CWP_Assistant`'s interface and recommended many modifications for it. Although they admitted that adequate information of tools' unavailability would be important for them (if there was any), they still found it difficult to accept eventual technical incidents, to think of incidents and to discuss solutions.

In order to merge the two different visions for our MAS solution to the reliability problem in ATC, we mixed researchers and operational operators (*i.e*. controllers) together in the third series of experiments. With the help of the researchers, the controllers concentrated on the information they require in each situation and gave valuable recommendations for each of scenarios. For example, concerning the one we have presented in Section 5, they suggested that although this scenario was only related to MTCD, in such situation of network failure, all the other tools (*i.e*. STCA, MONA, APW, MSAW) would find themselves in the same state as MTCD.

7. RELATED WORK

Researchers often take into account human factors in critical socio-technical systems [9][21] either by specifying users' working procedures [16][20] or by applying system design methods that help to prevent human errors [23][28]. Little work has dealt with the daily relation between human operators and their powerful equipments, particularly in situations where technical incidents happen. On the other hand, fault-tolerant methods applied to this kind of system have mainly solved purely technical reliability problem. Then they could not build total confidence for human operators while using automated tools.

Concerning the use of so-called "sentinels" in fault-tolerant component-based systems, as well as in certain MAS, the work of Klein, Dellarocas and colleagues [3][13][14] is also related to the monitoring of a complex critical system. However, they do not use simple communicating sentinel agents but complicated "sentinel components" to detect and deal with exceptions occurring inside application components. These "big" sentinels hence have their own reliability problem. Besides, Hägg [11] and Shah et al. [25][26][27] employ sentinel agents to detect and recover errors in negotiation processes between BDI agents. Nevertheless, these application agents have to be sufficiently "small" that the sentinel agents can fully inspect their code. This condition does not hold in a system having complicated equipments like ATC.

Regarding other agent-based systems used in ATC, we have been motivated by the work of Ljungberg, Rao and Georgeff [15][19] on a decision-support tool that helps air traffic controllers at an airport to predict optimal landing flows. It is a good example of *Distributed Artificial Intelligence* built into a single program. However, we argue that in order to develop such assistant tools in the future distributed ATC system (see Section 2) it will be necessary to distribute them over LAN/WAN networks, and to make them interact with other automated tools. In this perspective, they will not assist an individual controller but a group of cooperating controllers. Moreover, one could recognize that the technical framework presented in this paper, which supports the integration of a MAS in a distributed ATC simulation environment, would not be limited to fault-tolerant objectives. It could be reused to build and test agent-based systems that, for example, make predictions on the evolution of the traffic.

8. CONCLUSION AND FUTURE WORK

This paper describes the way in which a MAS can help in mitigating the effects of software malfunction in a complex critical system and building confidence for its users, *i.e*. air traffic controllers. Because of safety restrictions, experiments on real traffic control are not allowed. Therefore, we have developed an ABS, by using eDEP, an ATC simulation platform, and DimaX, a multi-agent platform, following the FIPA specifications [8].

This simulation has been used to demonstrate the usefulness of our MAS for the future ATC system to air traffic controllers. Indeed, we run typical applicative scenarios that show the reaction of our MAS to the instant unavailability of software

tools due to incident techniques. Some experiments with professional controllers have also been performed in order to validate the conformity of the information provided to them with what they require in situations where some software tools are not available.

However, the agents themselves, like any supplementary layer added to a system, bring their own liability to fault. A natural extension of the present work will be to set up mechanism for ensuring a degree of fault-tolerance at the agent level, which would be of a computational, domain independent nature. The possible techniques would include adaptive replication [17] and exception handling [4].

9. REFERENCES

[1] Cristian, F., Dancey, B. and Dehn J. 1996. Fault-tolerance in air traffic control systems. In ACM Transactions Computer Systems, vol. 14, n°. 3, pp. 265-286.

[2] DimaX project team. 2007 *http://www-poleia.lip6.fr/~guessoum/DimaX/index.html*.

[3] Dellarocas, C. 1998. Toward Exception Handling Infrastructures in Component-Based Software. International Workshop on Component-based Software Engineering.

[4] Dony, C., Knudsen, J. L., Romanovsky, A.B. and Tripathi, A. 2006. Advances Topics in Exception Handling Techniques. In Lecture Notes in Computer Science, vol. 4119.

[5] eDEP project team. 2007 *http://www.eurocontrol.fr/projects/edep/*.

[6] European Safety Programme. 2007 Draft of the Guidance Material for Contingency Planning. Technical Report, EUROCONTROL.

[7] EUROCAE project team. 2006 Flight Object Interoperability Proposed Standard (FOIPS) Study. Technical Report, EUROCONTROL.

[8] FIPA. 2001 FIPA Agent Software Integration Specification.

[9] Gregoriades, A., Sutcliffe, A. G. and Shin, J. E. 2002. Assessing the Reliability of Socio-Technical Systems. Proceedings of the 12th Annual Symposium of the International Council on Systems Engineering (INCOSE 2002).

[10] Gustafsson, F. 2007. Statistical Signal Processing Approaches to Fault Detection. In Annual Reviews in Control (JARAP), vol. 31, n°. 1, pp. 41-54.

[11] Hägg, S. 1996. A Sentinel Approach to Fault Handling in Multi-Agent Systems. In Distributed AI, Lecture Notes in Computer Science, vol. 1286, pp. 181–195.

[12] Isermann, R., 2006. Fault-Diagnosis Systems - An Introduction from Fault Detection to Fault Tolerance. Springer-Verlag Berlin Herdelberg.

[13] Klein, M. and Dellarocas, C. 1999. Exception Handling in Agent Systems. Proceedings of the Third International Conference on Autonomous Agents, Seattle, pp. 62–68.

[14] Klein, M., Rodriguez-Aguilar, J. A. and Dellarocas, C. 2003. Using Domain-Independent Exception Handling Services to Enable Robust Open Multiagent Systems: The Case of Agent Death. In Autonomous Agents and Multi-Agent Systems, vol. 7, n°. 1-2, pp. 179–189.

[15] Ljungberg, M. 1992. The Oasis Air Traffic Management System. Technical Note 28, Australian Artificial Intelligence Institute, Carlton, Australia.

[16] Mearns, K., Whitaker, S., Flin, R., Gordon, R. and O'Connor, P. 2000. Factoring the Human into Safety: Translating Research into Practice. In Benchmarking Human and Organisational Factors in Offshore Safety, OTO 2000 061, Sudbury: HSE Books, vol. 1.

[17] Marin, O., Bertier, M. and Sens, P. 2003. DARX - A Framework for the Fault-Tolerant Support of Agent Software. ISSRE'03, Denver.

[18] Petricel, B. and Costelloe, C. 2007. First ATC Support Tools Implementation (FASTI) Operational Concept. Technical report, EUROCONTROL.

[19] Rao, A. and Georgeff, M. 1995. BDI Agents From Theory to Practice. Technical Note 56, AAII.

[20] Reiman, T. and Oedewald, P. 2006. Organizational Culture and Social Construction of Safety in Industrial Organizations. In Svenson, O., Salo, I., Skjerve, A. B., Reiman, T. and Oedewald, P., eds., Nordic Perspectives on Safety Management in High Reliability Organizations: Theory and Applications.

[21] Rouhiainen, V. 2006. Safety and Reliability. Technology Theme – Final Report.

[22] Saha, G. K. 2006. A Single- Version Scheme of Fault Tolerant Computing. In Journal of Computer Science & Technology, vol. 6, n°. 1.

[23] Scacchi, W. 2004. Socio-Technical Design. In Bainbridge, W. S., ed., The Encyclopedia of Human-Computer Interaction. Berkshire Publishing Group.

[24] Scheina, J., Bushbya, S. T., Castroa, N. S. and House, J. M. 2007. A Rule-Based Fault Detection Method for Air Handling Units. In Energy and Buildings, vol. 38, n°. 12, pp. 1485-1492.

[25] Shah, N., Chao, K-M., Godwin, N. and James, A.E. 2005. Exception Diagnosis in Open Multi-agent Systems. Intelligent Agent Technology, IEEE Computer Society, pp. 483–486.

[26] Shah, N. Chao, K-M., Godwin, N., James, A.E. and Tasi, C-F. 2006. An Empirical Evaluation of a Sentinel-Based Approach to Exception Diagnosis in Multi-Agent Systems. Advanced Information Networking and Applications, IEEE Computer Society, vol. 1, pp. 379–386.

[27] Shah, N., Chao, K-M., Godwin, N., Younas, M. and Laing, C. 2004. Exception Diagnosis in Agent-Based Grid Computing. International Conference on Systems, Man and Cybernetics, IEEE, pp.3213–3219.

[28] Wilpert, B. 2005. Psychology and Design Processes. In European Psychologist, vol. 10, n°. 3, pp. 229-236.

A Multi-Agent Geo-Simulation Approach for the Identification of Risky Areas for Trains

Nabil Sahli
Telematica Instituut
P.O. Box 589, 7500 AN Enschede
The Netherlands
(+31) 53 - 485 0352
Nabil.Sahli@telin.nl

Mehdi Mekni
Department of Computer Sciences and Software Engineering,
Laval University
Québec (QC) G1V 0A6, Canada
(+1) 418 - 656 2131
Mehdi.Mekni.1@ulaval.ca

Bernard Moulin
Department of Computer Sciences and Software Engineering,
Laval University
Québec (QC) G1V 0A6, Canada
(+1) 418 - 656 5580
Bernard.Moulin@ift.ulaval.ca

ABSTRACT

The identification of risky areas along the railroad in the context of a railway system is a complex problem. A railway system is spatially and functionally distributed; its subsystems have a high degree of autonomy, and are in constant interaction with each other and with their geographic environment. In order to identify risky areas in the vicinity of rock falls zones we need to model and simulate the train behaviours in large scale geographic environments. Such a process involves coping with a variety of dynamic variables including the train characteristics, the environment properties as well as the weather conditions. The traditional mathematical and statistical modelling techniques which are usually used for the identification of risky areas do not satisfy all the requirements of such a complex process where spatial constraints are of high importance. In this context, m*ulti-Agent geo-Simulation* provides a flexible approach that can be used to easily simulate complex systems in large scale georeferenced environments. The purpose of this paper is to present *Train-MAGS*, an agent-based geosimulation tool which simulates train behaviours and identifies risky areas in large scale geographic environments. We show how agent-based simulation opens interesting perspectives regarding the development of new functionalities to improve risk assessment in the transportation field, more particularly for railway networks.

Keywords

Agent Based Simulation, Geosimulation, Train Derailment, Large-Scale Geographic Space, Risk Assessment.

1. INTRODUCTION

Our literature review showed that the agent paradigm and multi-agent systems in traffic and transportation are used to address several issues in various fields including *traffic modelling*, *decision support systems for better transportation*, *logistics planning*, *sea freight transportation*, *vehicle dispatching*, and *railway transportation traffic and scheduling* [1-3].

In this paper we focus on transportation and traffic systems. While most of works in this domain concern transportation infrastructures (road or networks) which do not really depend on geographical constraints, we are interested in large railway systems which are highly constrained by the geographical environment. Indeed, we use Agent-Based Simulation (ABS) with the purpose of predicting and analyzing potential perturbations which are not only related to the transportation infrastructure (here, tracks and trains) or to its users' behaviours (here train conductor) - as previous works do - , but also to the real environment (particularly the geographic space) in which the transport infrastructure is situated. We thus aim at showing how ABS can solve complex problems in large railway systems.

In this paper we present the Train-MAGS project, an agent-based geosimulation prototype. Geosimulation is a modelling approach which is concerned with the construction of high-resolution spatial models. These models are used in order to explore ideas and hypotheses about how spatial systems operate when developing simulation software and tools to support agent-based simulation, and applying simulation to solve real problems in geographic contexts [4]. In this project, we aim at investigating the contribution of the multi-agent geosimulation to help identify risky areas in the vicinity of rock falls zones in large scale geographic environments. We thus use agents which have an enhanced knowledge with respect to the virtual geographic environment [4]. Moreover, we examine how our agent-based approach can be applicable to a wide range of complex transportation systems' simulations where the spatial dimension is of major importance. The rest of the paper is organized as follows. Section 2 introduces the problem of the identification of risky areas in large scale railway systems. Section 3 presents the multi-agent geosimulation approach for the modelling and simulation of complex system in spatial environments. Section 4 highlights the main characteristics of Train-MAGS: the proposed multi-agent geosimulation simulator for trains. Section 5 presents the main functionalities of the Train-MAGS tool and how the simulation results are assessed. Section 6 presents future perspectives and extensions of the Train-MAGS platform. Finally, Section 7 concludes with some summary discussion.

2. IDENTIFICATION OF RISKY AREAS

Canada has a large railway system, with 49,422 kilometres, that today primarily transports freight [5]. There are two major privately owned transcontinental freight railway systems, the

Canadian National and Canadian Pacific Railway. These companies operate hundreds of freight and passenger trains each day over some of the world's most rugged terrain, and in some of the world's worst weather conditions. Train derailment constitutes a major problem for these companies. In 2007, 465 derailments were reported in Canada. Apart of their high cost, these accidents may lead to the release of hazardous materials, and to property and environment damage [5].

Train derailment, which is prone to uncertainty, is influenced by a large number of constraints such as the geologic state of the terrain (which can cause rock falls), weather conditions (e.g. rain, snow, fog), human behaviours (e.g. fatigue of the conductor), and train characteristics (e.g. braking system, weight). In this paper, we define a *risky area* as a portion of the rail track that precedes a possible obstacle (e.g., a rock fall) and on which the train should limit its speed so that it can avoid derailment if the probable obstacle is confirmed. Even if some special sensors can be used (e.g. sensor spots are currently used to detect rock falls), monitoring a very large railway system remains impossible because it would be too expensive. However, developing software to help identify risky areas could help railway companies to optimize the train traffic by setting adequate speed limits in these areas and allowing higher speeds elsewhere. The stakes for railway companies are high since they are committed to deliver on time the freight to their customers. Any delay implies penalties that increase the railways' operating costs. In such a context, any way of diminishing the risks of train derailment may be invaluable. In this project, we are particularly interested in rock fall hazard zoning [6], and the identification of risky areas in the vicinity of such zones that are prone to various types of rock falls. The traditional way of identifying such risky areas consists in having an expert follow the tracks on a special car in order to visually inspect the landscape surrounding the tracks in order to visually identify the areas where there is a higher probability of rock falls. Then, for each risky area, the inspector needs to assess the maximum speed of the train that would enable a train operator to brake in time in order to stop the train before reaching a possible obstacle on the tracks. Indeed, the distance required to bring a train to a complete stop depends on the capacity of the train's operator to detect the obstacles on the tracks, on the train's speed and on the tracks' conditions (i.e. slipperiness, presence of snow). The inspection of risky areas[1] is complex, requires a scarcely available expertise and can be only carried out on certain portions of an extended railway network and at certain times. Moreover, the procedure should be done under several different circumstances (e.g., different weather conditions). Hence, there is a significant interest to develop simulation software that may help practitioners to identify such risky ares under different conditions.

Few papers have addressed problems related to the simulation of train behaviours from an analytical point of view [7-10]. However, the increase of the parameters that must be considered to achieve a realistic train operator's perception of obstacles leads to a complex mathematical system. Moreover, the formulation of a mathematical model, which includes various factors such as the geologic state of the terrain, weather conditions, human behaviours, and train characteristics, seems complex and not obviously feasible. The complexity of the formulation and the resolution of such analytical models motivated us to find an alternative approach that addresses the simulation of train behaviours while taking into account the abovementioned parameters and factors. In addition, other modelling and simulations capabilities are required if we want to plausibly address the problem of obstacle perception. We need to create geosimulations (simulation of phenomena taking place in virtual geographic spaces generated from georeferenced data) involving autonomous agents that are '*spatially-aware*' in the sense that they are able to perceive the terrain characteristics of the virtual geographic world.

Agent technologies started to penetrate the transportation domain only recently [3, 11]. Agents are able to represent various kinds of entities in the transportation domain. Agents may simulate users involved in traffic, means of transport (trucks, trains, planes, ships), or elements of the traffic infrastructure. Agents can also be used to simulate the behaviours of such entities as well as their interactions with each other and with their geographic environment [12]. Thanks to the flexibility provided by the agent paradigm for the characterization (attributes, capabilities, and behaviours) of trains in their context, agent technology is an appropriate choice for modelling trains in the transportation domain [13].

3. GEO-SIMULATION AND MULTI-AGENT SYSTEMS

Geosimulation is a modelling approach which is concerned with the construction of high-resolution spatial models. These models are used in order to explore ideas and hypotheses about how spatial systems operate when developing simulation software and tools to support agent-based simulation, and applying simulation to solve real problems in geographic contexts [4]. Geosimulation differs from conventional urban simulation in its constituent 'elements'. Geosimulation models operate with human individuals and infrastructure entities, represented at spatially non modifiable scales such as households, homes, or vehicles. Many of these entities are animated (visually and dynamically) [14]. Geosimulation is a useful tool to integrate the spatial dimension in models of interactions of different types (economics, political, social, etc.) [15]. The GIS plays an important role in the development of geosimulation models. New methodologies for manipulating and interpreting spatial data developed by geographic information science and implemented in GIS have created added-value for this data [16].

[4, 14, 16] presented *Multi-Agent Geo-Simulation* as a coupling of two technologies: the Multi-Agent Based Simulation technology (MABS) and the Geographic Information systems (GIS). Based on the MABS technology, the simulated entities are represented by software agents that can be autonomous in their behaviours. They can interact with other agents and with the spatial environment. They may be reactive, proactive, mobile, social or cognitive [16]. Thanks to the agents' capabilities, we can use them to model and simulate complex entities or systems. Using the GIS technology, spatial features of geographic data can be introduced in the simulation.

[1] In this paper we only consider rock falls as a potential direct risk.

Multi-agent geosimulation is a powerful concept that can be used to simulate complex systems in georeferenced environments. According to our literature review, there exist a small number of multi-agent geosimulation candidate platforms that can be used to simulate systems in geographic environments using the agent paradigm. As examples, we can cite the platforms CORMAS (*Common-pool Resources and Multi-Agent Systems*) [17] and MAGS (*Multi-Agent Geo-Simulation*) [16]. In our work we use the MAGS platform to simulate the train behaviours in large scale geographic environments [16]. Our objective is to show how multi-agent geosimulation opens interesting perspectives regarding the development of new functionalities to improve risk assessment in the transportation field, more particularly for railway networks.

4. THE TRAIN-MAGS APPLICATION

In this section we first present an overview of the Train-MAGS application design. Next, we focus on the system architectural features as well as the involved agent types.

4.1 Design Overview

The Train-MAGS application can be thought of as a layered architecture as illustrated in Figure 1 (rectangles on the right side). We briefly present the four layers of our architecture in general as well as its application to the identification of risky areas in the vicinity of rock falls zones for trains in large scale geographic environments. This design philosophy is inspired by the layered simulation model proposed in [4]. It aims at building a parallel between the real world (rectangles on the left side, Figure 1) and the *Virtual Geographic Environment* VGE (right side).

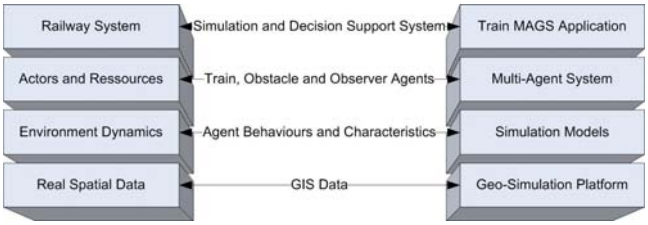


Figure 1: The Train-MAGS Design Overview

1st Layer: It is the software platform which is in charge of reproducing the real geographic world in the VGE. A GIS is essential to reproduce real spatial data in the simulation environment. In addition, human users need a visual tool to supervise the geographic environment. It is thus necessary to transform GIS data into a simulation software platform which is visual for human users and navigable for software agents. Therefore, the Train-MAGS application embodies, at each cell of the simulated environment, information elevation, percentage slope, slope direction, etc. This information is relevant to the simulation of train and conductor's behaviours.

2nd layer: It is responsible of modelling the dynamic factors influencing the real world. In fact, a GIS cannot describe all the spatial data that influence the environment. External factors such as atmospheric phenomena and weather conditions make the environment much more dynamic and unpredictable. Complementary models (such as rock falls) have to be used to simulate this dynamism. Data used by these models should first be captured from the real world and then continuously updated. The model can thus provide a reliable and progressive simulation of the environment. In our example, there is a need to model dynamic data.

3rd layer: This is the multi-agent layer. It represents actors (those who perform actions in the real terrain) of the real world. There is a need for software agents situated in the terrain (Concretely, these could be sensors or electronic devices which sense their neighbourhood and in which agents are embedded). We also need agents in the VGE. Agents within the real world may then communicate with agents within the VGE, which guarantees a better coherency between data collected from both the real and the VGE. Each actor should have a software agent as a representative within the VGE. In our example, an actor is a *Train* which would possess a mobile platform in which an agent (*Train Agent*) is embedded. This train interacts with the *Train Agent* via an interface and communicates with its representative in the VGE via remote messages. Before they can act (navigate, perceive, etc.) within the VGE, software agents need to be coherently linked to real world actors.

4th layer: It represents the functionalities which are domain-specific. Actually, the three previous layers provide a foundation for applications aiming to help decision makers as well as users of the simulation platform with the goal of better strategic decisions. These applications are located in the fourth layer. For our example, the goal of the simulation of the train behaviours is to identify risky areas and to propose recommended speed limits in the vicinity of rock falls zones.

4.2 Architecture

The Train-MAGS architecture includes the simulation core engine (corresponding to the 1st layer, Figure 1) which interacts with several simulation models (2nd layer) such as the physical model (weather conditions: rain, snow, fog), the model of risk (rock falls frequency and location), and a braking model which represents the braking process of the train in case of obstacle detection. The simulation engine also exploits a VGE built from GIS data. Finally, the Train-MAGS tool enables us to create two types of agents (forming the 3rd layer):

- Mobile (moving) agents with navigation, perception and decision making capabilities [16]. These mobile agents are immersed in the VGE and processed by the simulation core engine. In Train-MAGS, the *Train Agent*, which represents the train, is a mobile agent.
- *Observer Agents* which do not represent any real entities. They are responsible for collecting data (see sub-section 4.2.3).

All these agents are spatially-aware in the sense introduced in Section 2. Figure 2 illustrates the conceptual architecture of the Train-MAGS application. All the elements mentioned above are detailed in the following sub-sections.

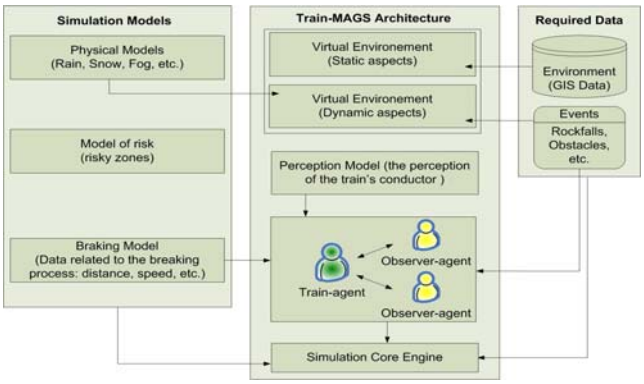


Figure 2: The Train-MAGS Conceptual Architecture.

4.2.1 Virtual Geographic Environment (VGE)

The VGE is a grid (bitmap) which is created from topographic data of a portion of the *Albreda region* (Canada), a digital elevation model and a data base providing the characteristics of the rail tracks. In addition, a module of the Train-MAGS system generates a set of other bitmaps [16] which are used by the *Train Agent* to get information about the space surrounding it and to simulate its perception process [11]. First, the system generates an *Elevation Map* which provides agents with data about the grades and slopes in the simulated area (at each cell of the grid). Next, it generates a bitmap called *Ariadne Map* from GIS data [16]. This map represents the rail track which is followed by the *Train Agent* (see Figure 3). The spatial information is recorded in a raster mode which enables the *Train Agent* to access the information contained in various bitmaps that encode different kinds of information about the terrain characteristics (e.g. slope, elevation) and the objects contained in the VGE (e.g. bridge, tunnel).

4.2.2 Train Agent Characterization

The *Train Agent* actually represents both the train and its operator. As the train representative and similarly to a real train, it follows the rail tracks, adjusts its speed according to the terrain's characteristics (mainly according to the grade of the terrain), and simulates the braking system. The *Train Agent* uses its navigation capability (see sub-section 4.2.2.2) to achieve its goals. As the operator's representative, it mainly simulates the perception of the operator (see sub-section 4.2.2.1).

The *Train Agent* (as all agents in the MAGS Platform [16]) is characterized by a number of variables whose values describe the agent's state at any given time. We distinguish static states and dynamic states. A static state does not change during the simulation and is represented by a variable and its current value. For example, the train category (e.g., passenger train, freight train) is a static characteristic which does not change during the simulation. A dynamic state is a state which can possibly change during the simulation. For example, the *Train Agent*'s speed and its braking distance to stop can change during the simulation depending on the context. Using both static and dynamic variables, the system simulates the evolution of the *Train Agent* in the VGE and triggers behaviours depending on changes of its states and on the achievement of its goals [16].

4.2.2.1 Perception

Perception is an important agent's ability which must be carefully simulated in a VGE if we want that agents exhibit plausible cognitive spatial behaviours. The *Train Agent* simulates both the displacement of the train and the perception of its operator who checks for potential problems on the rail tracks ahead of the train position. By analogy to human spatial perception, we identified several perception modes for the *Train Agent*: 1) perception of terrain characteristics (elevation and slopes) in the area surrounding the agent; 2) perception of the landscape surrounding the agent (including trees and static objects such as rocks on tracks); 3) perception of other mobile agents navigating in the agent's range of perception (pedestrians or cars crossing rail tracks, other trains); 4) perception of "*dynamic areas*" with specific properties such as foggy areas.

Several research works have already tried to address the problem of simulating perception in an environment represented using a height map. The goal of these techniques was to determine the visibility of all the cells of the height map which are in an observer's field of vision. They use lines of sight in order to test the cells' visibility (labelled as visible or not) from the observer's location.

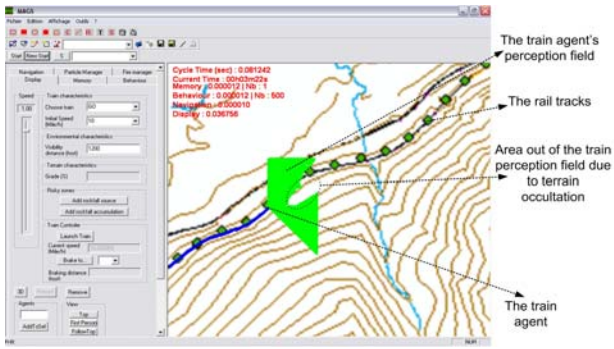


Figure 3: The *Train Agent* Field of Perception.

In MAGS [16] we use an approach extending Franklin's algorithm in a way that enables agents to perceive the environment as well as other agents in real time [16]. The *Train Agent*'s perception field is represented by an isosceles triangle, the main vertex being at the agent's location, the congruent sides of the triangle limiting the perception field and the bisector of the main angle corresponding to the agent's direction of movement. The length of the bisector corresponds to what we call the *Perception Radius* (PR). The angle of perception is a parameter that can be adjusted (currently set to 160 degrees to mimic conductor's perception in the front of the train) (Figure 3). Since perception takes into account the terrain elevations, an area which is hidden by an elevated portion of terrain will not be perceived (obviously) as shown in Figure 3.

4.2.2.2 Navigation

In order to optimize the agent's navigation function, we exploit the *Ariadne Map* (Figure 3). The *Train Agent* can directly access it in order to determine which cells around it correspond to its path (rail tracks). The *Train Agent* navigates using a *following-a-path mode* [16] which consists in forcing the agent to follow a predefined path in the VGE corresponding to the rail tracks. The Train-MAGS's navigation module is able to access

the *Ariadne* Map's portion which is perceived by the agent. Data extracted from the *Ariadne* Map is then used to compute the agent's next move. Moreover, depending on the terrain's characteristics, the *Train Agent* adjusts its speed in the VGE in order to reflect the real situation.

4.2.2.3 Objective-Based Behaviours

In Train-MAGS an agent is associated with a set of objectives or goals that it tries to reach. The objectives are organized in hierarchies, which are trees composed of nodes representing composite objectives and leaves representing elementary objectives that are associated with actions that the agent can perform. Each agent owns a set of objectives corresponding to its needs. An objective is associated with rules containing constraints on the activation and the completion of the objective. Constraints are dependent on time, on the agent's states, and the environment's state. The selection of the current agent's behaviour relies on the priority of its objectives. Each need is associated with a priority, which varies according to the agent's profile. An objective's priority is primarily a function of the corresponding need's priority. It is also subject to modifications brought by the opportunities that the agent perceives or by temporal constraints [16].

4.2.3 Observer Agents

In the MAGS platform, an *Observer Agent* is an agent whose behaviour is dedicated to data collection during the simulation process [16]. The *Observer Agents* are immersed in the VGE at specific geo-referenced locations in order to closely observe particular phenomena. Their perception and memory capabilities are basically used to perform data collection. This data is then analyzed in order to better understand the observed phenomenon. In the context of train behaviours' simulation, we use *Observer Agents* located in the obstacles' surroundings in order to observe the changes of the train's behaviour when it detects the obstacle. The train position, added to obstacles' positions, the braking distance, and the terrain grade are used to assess the capability of the train to brake before hitting the obstacle. *Observer Agents* are located at the intersection of rail tracks and rock fall zones which are represented graphically by nested ellipses as shown in Figure 5. The simulation process, when the *Observer Agent* perceives *Train Agent*, it starts collecting data related to the context such as the *Train Agent*'s position, the terrain topology, and the risk level of the area. On the basis of this data the system carries out an analysis in order to assess the *Train Agent* behaviour in the vicinity of risky areas. The result of such an analysis is presented in Section 5.

4.2.4 Obstacle Agents

In the Train-MAGS platform, obstacles are introduced in the simulation scenario using a specific type of agent called *Obstacle Agents*. *Obstacle Agents* represent physical obstructions in the real world. This type of agent may be *stationary* or *mobile* [16]. *Stationary Obstacle Agents* (SOA) include landslide, fallen trees, significant quantities of concrete materials, and equipment or freight fallen from other trains as well as any object that is liable to pose a danger to the safe passage of trains. On the other hand, *Mobile Obstacle Agents* (MOA) include farm livestock or other animals that have entered upon the track as well as road vehicles at a level crossing. These agents can be immersed in the VGE by the user at simulation run time at specific geo-referenced locations. In the current version of the Train-MAGS platform, only *Stationary Obstacle Agents* are implemented. The integration of *Mobile Obstacle Agents* is part of the future works as discussed in section 7.

4.2.5 Weather Conditions

Certain gaseous phenomena such as smoke and fog are related to the VGE's atmosphere and cannot be modelled using agents. They are associated with areas or volumes whose properties (boundaries, local density, etc.) change dynamically under the influence of external forces like the wind. A good way to simulate such phenomena is to use particle systems. The MAGS platform provides such tools which have been used to simulate tear gas in crowd simulation in urban areas [16]. However, in the current version of the Train-MAGS tool, atmospheric phenomena such as fog, rain and snow, are not yet included since we did not get the corresponding data yet for the area in which we conducted our experiments. Weather conditions need realistic data to be modelled and simulated.

4.3 Operating Modes

The Train-MAGS is a two-phase project which aims at providing a multi-agent geo-simulation platform which is able to operate in two different modes: ***Pre-execution mode*** and ***Real-time mode***. The *Pre-execution mode* is already supported by the current version of this simulation tool, while the *Real-time mode* is under implementation.

4.3.1 Pre-execution Mode

A decision maker may use the Train-MAGS platform in order to analyze and assess different situations by simulating various scenarios involving stationary and mobile obstacle agents. The *Train Agent*'s behaviours are thus evaluated by the decision maker and speed limits are defined with respect to the to the risk identification process. Using Train-MAGS in the *Pre-execution* the user can achieve the following goals:

- Identification of risky areas;
- Elaboration of a table which summarizes the recommended speed limits with respect to the identified risky areas in large scale geographic environments.

4.3.2 Real-time Mode

The Train-MAGS system's architecture can be coupled to a real-time monitoring system. Hence, the Train-MAGS application may be used for different purposes taking advantage of its simulation capabilities. Indeed, this tool could first keep track in the simulation environment of the effective moves of the train in the real world using a GPS (*Global Positioning System*) tracking system. Second, it could periodically update the information regarding the geographic environment using the data sent by sensors and actuators located at specific geo-referenced positions. Finally, the Train-MAGS could suggest possible strategies as soon as an unexpected event is reported. Possible events include detection of stationary or mobile obstacles along the rail tracks. The proposed suggestions consist in modifying the train behaviour such as reducing its speed. The *Real-Time mode* is a challenging objective since it aims at

exploiting the Train-MAGS platform as a monitoring system for trains. This topic will be discussed in Section 6.

5. SIMULATION AND RESULTS

In this section, we discuss the implementation of the *Pre-execution mode*. The Train-MAGS application, which corresponds to the 4th layer in Figure 1, enables a user to create the VGE and to specify the train's characteristics (category, speed, perception radius, etc.). It also offers the possibility to create two different types of simulation scenarios: the *interactive scenario* and the *modelling rock fall Probabilities scenario*.

The *interactive scenario* allows the user to introduce, at simulation run time, geo-referenced rock fall hotspots in the VGE. The *Train Agent* can perceive rock fall hotspots using its perception function. When an obstacle is detected on the tracks, the *Train Agent* triggers its braking process. The braking process is a complex computation model which takes into account several variables such as the train category, the speed, the terrain characteristics, and the state of the rail track. These variables are collected thanks to the *Observer Agents*. The braking process is launched after the train's conductor perceives the obstacle. Thus, the simulation of the braking process strongly depends on the *Train Agent*'s perception (it also depends on the response time of the conductor, but this parameter is not considered here for simplification reasons). The objective of the *interactive scenario* type is to determine the capacity of the simulated train to brake at a safe distance from the perceived obstacle, given the aforementioned parameters.

Modelling Rock Fall Probabilities Scenario enables the user to simulate the risk instead of the event (rock fall zones instead of obstacles detection) (Figure 4). Thanks to data generated by statistic models of the rock fall phenomenon, rock fall probabilities are modelled by nested ellipses and introduced in the VGE as shown in Figure 5. Ellipses are used instead of the original non regular shapes for simplification reasons. Each ellipse represents a probability level. The probability increases towards the centre line of these ellipses. Thus, the smallest ellipse represents the highest probability of rock fall. From the *Train Agent*'s point of view, a higher probability of rock fall corresponds to a higher level of risk. The *Train Agent* uses its perception field and computes the minimum distance from which the operator may perceive a possible rock within the zone delimited by the ellipse while taking into account the elevation of the terrain (*grade*) measured by the *Observer Agents*.

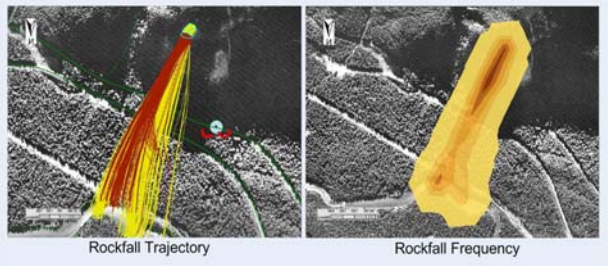


Figure 4: The distribution of rock fall probabilities **in key areas of the *Albreda* region (Canada). The map was generated by statistic models and built upon GIS layers for areas of special interest around railway traffic corridor, with layers depicting rock fall risks.**

Given this distance, the Train-MAGS application determines (using heuristic data provided by Canadian National) the maximum speed which is allowed for the train in order to be able to brake on time if an obstacle is perceived inside one of the ellipses.

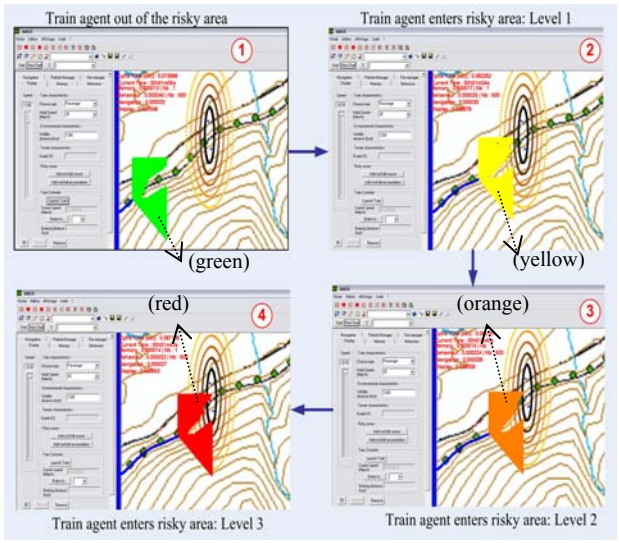


Figure 5: Perceiving the Rock fall Zones .

In order to determine the risk level (from the *Train Agent*'s point of view) depending on the train's position, the *Train Agent*'s perception field is coloured ranging from green to red, where the green colour warrants a safe area and the red colour indicates that the train is in a risky area (a risky area, defined in Section 2, is thus the distance between the current position of the train and the ellipse's perimeter). Figure 5 presents the changes of the colours of the *Train Agent*'s perception field as a consequence of the risk level of the area crossed by the train. The computation of the braking distance to the obstacle can also be used to determine the maximum allowed speed in order to enable the train to stop before reaching the rock fall zone. This speed depends on the perception distance (which depends on several constraints including the weather conditions and curves), the terrain topology (grade of the tracks), and the distance between the *Train Agent* and the rock fall zone. The Train-MAGS system generates for each rock fall probability other output such as a table indicating the maximum allowed speeds for a given region (Table 1).

Table 1: Recommended Speeds for a risky area (the considered train weights 24 tons: 3 locomotives+196 cars).

Visibility (m)	Train Position (X,Y)		Perceived Rock fall zone (X,Y)		Risk Level	Distance to Rock fall zone (m)	Grade	Recommanded Speed (Km/h)
450	48	780	73	767	1	280	0.000	20
450	48	780	78	762	2	350	0.089	23
450	48	780	83	757	3	410	0.093	24
450	52	779	88	753	4	440	0.098	24
450	60	776	93	748	5	430	0.103	24
450	69	770	103	740	4	450	0.097	27
450	89	751	108	737	3	230	0.093	16
450	90	751	113	735	2	280	0.088	20
450	103	740	118	734	1	160	0.084	15
450	111	736	123	733	0	0	0.000	50

Using such a table, the train behaviour may be changed by reducing its speed in order to be able to brake on time in case of an obstacle is detected (e.g., perceived by the conductor). In this table we suppose that the conductor has a limited visibility (450m) and that the train is approaching a curve. The zone between (73,767) and (123,736) is a zone where rock falls are possible. Since this zone is partially hidden -due to the topology of the area (the curve)-, a conductor may see (according to our simulation) the beginning of this zone only from position (48,780), i.e. at a distance of 280m (even if the visibility is 450m). However, later and once he is at position (60,776), he may see the rock fall zone with the highest risk (level 5) at a longer distance 430m (probably because at this position, the conductor has a better view of the rail track, i.e. no curve). The recommended (maximum allowed) speeds are calculated given the distance between the train and the rock fall zone (e.g., 280m, 430m), the risk level (from 1 to 5), and the terrain grade measured by the *Observer Agents* (in Table 1, grades are positive, which indicates an uphill rail portion. Recommended speeds would have been lower if grades were negative). In Table 1, the risky area, defined in Section 2, is the rail track portion between (48,780) and (111,736) (1st and last rows of Table 1). Beyond this area, the risk level is "0" and the train can have a higher speed (e.g., 50Km/h in the last row of Table 1).

6. TOWARDS REAL TIME RISK ASSESSMENT

The Assessment of risk through the identification of risky areas is achieved using the Train-MAGS platform in the *Pre-execution mode*. Moreover, with advances in computation, telecommunication, and sensing technologies, it is nowadays possible to monitor large spaces and particularly the geologic state of a given portion of the terrain [6].

The next phase of the Train-MAGS project aims at synchronizing the simulation environment with the real world in order to assess risk in *Real-time mode*. To this end, the VGE is synchronously coupled with the real environment [9], which means that the VGE should reflect in real-time what is actually happening in the terrain. Therefore, we suppose that trains are equipped with GPS devices and that specific sensors and actuators are deployed in critical zones to measure certain geologic parameters (which are used to monitor certain geologic parameters that could indicate a potential rock fall) [6]. The *Train Agent* is synchronously linked to the real train and keeps track of its effective moves within the VGE. Each sensor device deployed in the terrain has its representative in the VGE: *Sensor Agent*. *Sensor Agents* are thus in charge of receiving data from the physical sensors which are deployed in areas of interest. In this case they are considered as situated agents who are not really proactive. Nevertheless, *Observer Agents* communicate with these *Sensor Agents* to collect, filter and fuse data, and derive potential risks of rock falls. Hence, the *Observer Agent* that detects a danger notifies the *Train Agents* which are concerned (those which, according to their current geo-referenced positions, are in the vicinity of and heading to this risky zone) by this potential threat. Each *Train Agent* modifies autonomously its behaviour by adjusting its speed when approaching the risky area. This briefly summarized our vision of a real-time risk assessment system for trains.

7. DISCUSSION AND CONCLUSION

Modelling and simulating the train behaviours including the geographic environment, the train's characteristics, the train operator's capabilities (perception and decision making), and the obstacles along the track, is a complex process. Several simulation tools have been developed to simulate train behaviours ranging from the game industry, to the physical dynamics of the train [8, 9] ending with traffic analysis and the performance assessment of the railway traffic [3, 10]. These simulators are generally built using mathematical and statistical models. In general, the complexity of these mathematical models leads to a trade-off between simplicity and accuracy of the model. Increasing the complexity usually improves the fit of a model. However, it can also make the model difficult to understand and to work with, and can also raise computational problems such as numerical instability. For example, when modelling the train's behaviour, we may include different parts and sub-systems of the train (the train, the tracks, etc.) in the model and would thus get a more detailed view of the system. However, the computational cost of adding such a large amount of details would effectively inhibit the usage of such a model. Additionally, the uncertainty would increase as a result of an overly complex system, because each separate part induces some amount of variance in the model. It is therefore usually appropriate to make some approximations to reduce the model to a sensible size. Engineers often can accept some approximations in order to get a simpler model. However, in the context of the identification of risky areas in large scale geographic environment, we need to model and to simulate the train's behaviour as well as its interactions with the spatial environment. Fortunately, the multi-agent geo-simulation approach provides an efficient tool for the simulation of complex systems while taking into account the spatial dimension. This approach gives a particular attention to the specific features of geographic data which can be introduced in the simulation and be accessible for agents. Hence, we used such a multi-agent geosimulation approach to model and to simulate the train behaviour using *spatially aware* agents in order to identify risky areas in large scale geographic environment.

Railway transportation is a complex system that may benefit from artificial intelligence techniques towards the development of 'intelligent transportation systems' [33]. In this paper we showed how certain characteristics of this system may be adequately modelled using autonomous agents and multi-agent systems. The Train-MAGS platform is a proof of concept of the proposed agent-based geosimulation prototyping approach which relies upon the agent's capabilities and the geographic environment built upon reliable GIS data. This simulation tool provides the possibility to build a table of recommended speeds in the surroundings of risky areas. This table may help to inform the train's operator about the vicinity of risky areas along the rail tracks. The Train-MAGS experiment also demonstrates the importance of using realistic GIS data in order to build the VGE as well as the identification of risky areas. We also showed how agent-based geosimulations can contribute to identify risky areas in large scale geographic environments. As a result of this kind of modelling, one could also meet the growing interest in making traffic and transportation more secure, efficient, resource-saving and

ecological. Besides, only few research works have been found addressing the effects of weather condition on train behaviours, we still need to analyze train behaviours under various weather conditions. Indeed, the breaking model as well as the *Train Agent*'s perception, navigation, and decision making capabilities are deeply related to such factors. Therefore, we are currently working on the integration of particle systems in order to extend the *Train-MAGS* capabilities to support gaseous phenomena simulation. Such capabilities will allow us to simulate atmospheric phenomenon including heavy rains, fog, and snow.

8. ACKNOWLEDGEMENT

This research was supported by the Canadian Network of centers of excellence in geomatics GEOIDE (Project GIST2), in collaboration to the Canadian National and the Centre for Risk Assessment for Geohazard Studies. The MAGS platform development has been supported by GEOIDE, RDDC Valcartier and the Natural Sciences and Engineering Council of Canada.

9. REFERENCES

[1] Gambardella, L. M., Alessandrini, A., and Epifani, C. 2000. "The Use of simulation in the socio-economical evaluation of the management of an intermodal terminal," In Proc. of the Maritime & Industrial Logistics Modelling and Simulation, Portofino, Italy.

[2] Ezzedine, H., Kolskia, C., and Péninou, A. 2004. "Agent-oriented design of human–computer interface: application to supervision of an urban transport network," *Engineering Applications of Artificial Intelligence*, vol. 18(3), pp. 255-270.

[3] Cuppari, A. 1999. "Prototyping Freight Trains Traffic Management Using Multi-Agent Systems," In Proc. of the Proc. of the 1999 International Conference on Information Intelligence and Systems, pp. 646-653.

[4] Sahli, N. and Moulin, B. 2006. "Agent-Based Geo-simulation to Support Human Planning and Spatial Cognition," LNCS, Springer, vol. 3891, pp. 115-132.

[5] TSBC. 2007. "The Transportation Safety Board of Canada," TSB Canada July 2007.

[6] Vařilová, Z. and Zvelebil J., 2005. "Sandstone relief geohazards and their mitigation: rock fall risk management in the Bohemian Switzerland National Park," In Proc. of the Sandstones Landscapes in Europe, FERRANTIA, Luxembourg 25-28 Mai, pp. 53-58.

[7] Malay, A. D. and Lawrence, P. J., 2001. "Simulation modeling at union pacific railroad," in Proc. of the 33nd conference on Winter simulation. Arlington, Virginia, IEEE pp. 1048-1055.

[8] Jeong, D., Perlman, B., and Chris, P. 2006. "Dynamic Simulation of Train Derailments," In Proc. of the International Mechanical Engineering Congress and Exposition, Chicago, IL 5-10 Nov..

[9] Durali, M. and Bahabadi, M. M. J. 2004. "Investigation of train dynamics in passing through curves using a full model," In Rail Conference, Proc. of the 2004 ASME/IEEE Joint, pp. 83-88.

[10] Lewellen M., and Tumay K., 1998. "Network simulation of a major railroad," In Proc. of the 30th conference on Winter simulation, Washington, D.C., United States, IEEE pp. 1135-1138.

[11] Davidsson, P., Holmgren, J., Kyhlbäck, H., Mengistu, D., and Persson, M. 2006. "Applications of Agent Based Simulation," In Proc. of the Multi-Agent-Based Simulation VII, Hakodate, Japan, Springer LNCS 3891, pp. 15-27.

[12] Cuppari, A., Guida, P. L., Martelli, M., Mascardi, V., and Zini, F. 1999. "An Agent Based Prototype for Freight Trains Traffic Management," In Proc. of the FMERail'99 Workshop (satellite event of FM'99), Toulouse, France.

[13] Zhu, K. and Bos, A. 1999. "Agent-based design of intermodal freight transportation systems," In Proc. of the NECTAR Conference, Delft.

[14] Benenson, I. and Torrens, P. 2003. "Geographic Automata Systems: A New Paradigm for Integrating GIS and Geographic Simulation," In Proc. of the GeoComputation conference, pp. 24-30.

[15] Mandl, P. 2000. "GeoSimulation- Experimentieren und Problemlösen mit GIS-Modellen," Angewandte Geographische Informationsverarbeitung XII, pp. 345-356.

[16] Moulin, B., Chaker, W., Perron, J., Pelletier, P., Hogan, J., and Gbei, E. 2003. "MAGS Project: Multi-agent geosimulation and crowd simulation," In Proc. of the COSIT'03 Conference Spatial Information Theory, Ittingen (Switzerland), LNCS 2825 pp. 151-168.

[17] Bousquet, F., Bakam, I., and Proton, H. L. P., C. 1998. "Cormas: Common-Pool Resources and Multi-Agent Systems," LNAI, pp. 826-838.

10. BIOGRAPHY

Nabil Sahli obtained a PhD in Computer Science from Laval University (Quebec). He is now a scientist researcher at Telematica Instituut (The Netherlands). His main interests are: multi-agent systems, agent-based geosimulation, agent-based planning, hazards simulation, trust and reputation, and virtual communities.

Mehdi Mekni received his B.Eng. (2000) degree from the department of information sciences (Tunisia), and his M.Sc. (2006) degree from the department of computer sciences at Laval University (Canada). He is now a PhD Student (supervised by Bernard Moulin since 2006). His research interests include Multi-Agent Systems and Geo-simulation.

Bernard Moulin has obtained an Engineering degree from Ecole Centrale de Lyon (France), a Degree in Economics (University Lyon2) and a PhD (applied mathematics- computer science, University Lyon1). He is now a full professor at Laval University in the Computer Science Department. His main interests are: multi-agent systems, agent-based geosimulation, and application of AI techniques on modeling and simulation of software agents.

Multi-Agent Technology for Air Traffic Control and Incident Management in Airport Airspace

Vladimir Gorodetsky
SPIIRAS
39, 14-th Liniya, 199178
St. Petersburg, Russia
7-812-3233570l
gor@mail.iias.spb.su

Oleg Karsaev
SPIIRAS
39, 14-th Liniya, 199178
St. Petersburg, Russia
7-812-3233570l
ok@mail.iias.spb.su

Vladimir Samoylov
SPIIRAS
39, 14-th Liniya, 199178
St. Petersburg, Russia
7-812-3233570l
samovl@iias.spb.su

Victor Skormin
CAIT
Binghamton University
Binghamton, NY 13902, USA
1-607-777-4471
vskormin@binghamton.edu

ABSTRACT

The paper presents a model, multi-agent architecture, implementation approach and software prototype of a multi-agent system for autonomous air traffic control within airport airspace capable of automatic detection of potential violations of safety policies by individual aircraft and consequent incident management. It features a model facilitating practical implementation of the concepts of openness and agent-based autonomy of air traffic control, social rules, distributed safety policy for conflict resolution, as well as predictive analysis and P2P interaction–based coordination of aircrafts' motion. The main results are validated by simulation.

Categories and Subject Descriptors

H.4.2 [**Decision Support**]: Distributed autonomous control system – *distributed multi-agent system, autonomous real-time control, peer-to-peer behavior coordination.*

General Terms

Algorithms, Management, Design, Experimentation, Security.

Keywords

Multi-agent system, autonomous air traffic control, P2P agent interaction, safety policy, incident detection and deconfliction.

1. INTRODUCTION

Due to ever increasing intensity of air traffic and increasingly rigid safety requirements, development of novel principles of Air Traffic Control (ATC) currently became a well recognized problem. Indeed, Air Traffic Control Operators (ATCO) are currently overloaded with their responsibilities and perform at the limit of their capacity. That is why the expected increase of air traffic intensity will inevitably exceed the capacity of existing ATC systems. An additional factor making the control problem highly critical is the increased frequency of abnormal situations, such as aircraft hijacking. In such situations, due to their highly dynamic and unpredictable nature, ATCO may completely fail to monitor and control the situation.

It is well recognized that satisfactory resolution of the described situation hinges upon providing individual aircraft as much control autonomy as possible and delegating them end-to-end routing and collision avoidance from the very take-off and to landing. Consequently, the free flight concept [1] for aircraft routing during cruising was formulated in the professional community ([8], [9]). This concept implies that every aircraft is provided some routing flexibility and the collision avoidance task is delegated to the autonomous pilot-assisting software based on distributed safety policy. Unfortunately, little attention is paid to the development of new principles of ATC within the airport airspace (AAS), where air traffic density is much higher while control processes are highly dynamic and the physical space is very limited.

Recent achievements in Multi-Agent System (MAS) thory provide a convenient framework for modeling and a technology for software implementation of autonomous ATC system within AAS. Indeed, agent-based modeling of collective behavior of distributed autonomous entities constrained by social rules and supported by distributed policy for conflict resolution, is the focus of many recent MAS research [6]. It provides adequate framework for autonomous ATC systems in question. Additionally, recent results in Peer-to-Peer (P2P) agent systems, in particular, development of reference model of P2P agent platform [7] and its subsequent software implementation ([2], [4]) provide unique architecture and technology for development of *open* systems with highly transient population of autonomous entities of MAS. It is important to note that the last property is intrinsic for ATC tasks.

The paper presents a conceptual model, multi-agent architecture, specification technology, and software prototype implementing ATC system within AAS. Together, these technologies implement the principles of openness and autonomy based on social rules, distributed safety policy for conflict resolution, predictive analysis and P2P interaction–based coordination of aircraft motions. Section 2 outlines basic domain knowledge and separation standards intended to assure the safety of aircraft motion. Section 3 describes typical behavior patterns of "normal" and hijacked aircraft and offers an organization concept of an ATC focused on agent-based autonomous path planning and P2P conflict resolution strategy. Section 4 outlines the developed distributed conflict resolution policy. Section 5 describes the developed architecture of a multi-agent ATC. Section 6 illustrates graphical user interface of the developed software prototype implementing basic ideas of the paper. Section 7 provides the conclusion describing the paper contribution and future work.

2. AIR TRAFFIC CONTROL DOMAIN KNOWLEDGE

2.1. Airport Airspace Topology

The high level notion of the *airspace topology* is intended to specify admissible trajectories of aircrafts sharing the airport airspace. It is worth noting that airspace topology does not address real-time air traffic configuration that concerns positions, speeds and courses of the set of aircrafts operating within AAS. Fig. 1 and 2 exemplify airspace topology (in horizontal and vertical projections respectively) in the New York City area uniting three airports, JFK, LaGuardia and Republic.

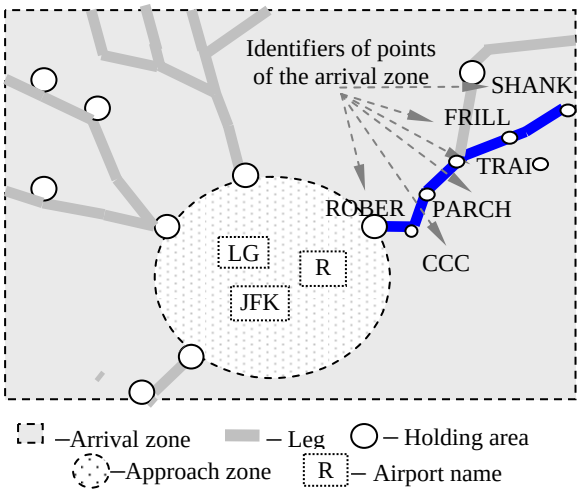


Figure 1. Airspace topology within New York City area (Horizontal projection), and arrival and approach

Airport airspace encompasses two zones: (i) arrival zone and (ii) approach zone. Arrival zone comprises *Arrival schemes*. E.g., Fig. 1, shows nine arrival schemes. Every arrival scheme begins with the entry point and is specified as a sequence of legs [3] ending with the *holding area*.

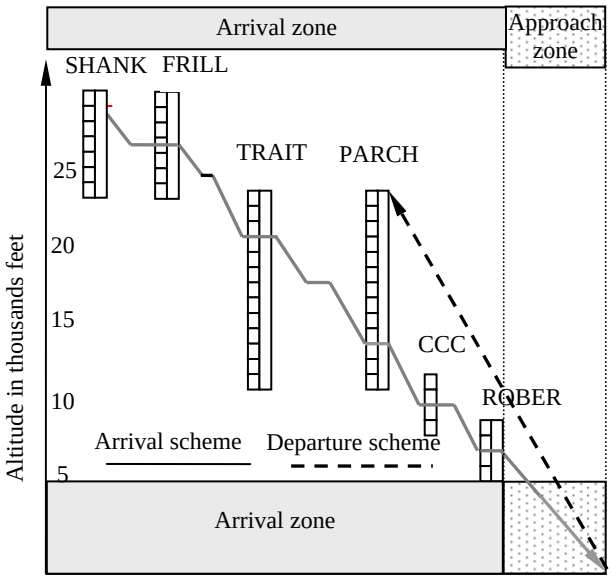


Figure 2. Airspace topology within New York City area

(Vertical projection) and arrival and approach zones)

Approach zone comprises *approach schemes*. These schemes are not depicted in Fig. 1 due to too small scale of the figure. Each approach scheme begins at the approach zone entry point, consists of sequence of legs and ends at an airport runway.

Movement schemes within each approach zone can be classified in two categories (with some vagueness), (i) *standard* approach schemes and (ii) *missed* approach schemes, where the latter occurs in exceptional situations (technical problems, hijacking, etc). As a rule, a missed approach results in the necessity to use a holding area. *Transition schemes* bind the destination points of the arrival schemes and entry points of approach ones. As a rule, each arrival scheme is bound with several approach schemes. Transition schemes are used for binding different arrival schemes.

Fig. 2 depicts movement schemes (arrival and departure) projected onto vertical plane. In the left part of the figure, along the vertical axis, the echelon scale (from 0 till 30,000 feet with quantization step of 1000 feet) is depicted. The vertical projection of landing path through the arrival and approach schemes passing through SHANK, FRILL, etc. points is given by solid line.

The specification of the airport airspace topology also determines admissible echelons, i.e. admissible altitude ranges for passing through exit points of the legs. For example, while passing through the SHANK point, aircraft are required to use the echelons in between 24, 000 – 30,000 feet. Some legs may be bound with holding areas. E.g., all legs of an arrival scheme shown in Fig. 2, excluding the CCC leg are bound with the holding zone.

Airport airspace topology specification also contains departure schemes. They begin at a runway and end at exit points of airport airspace. Since climbing rate of an aircraft typically exceeds its descending rate, the exit points are located (in horizontal plane), between outer boundaries of the approach and arrival zones.

2.2. Separation standards

Separation standards defined for various air traffic–related situations constitute the basis for air traffic safety. They must be observed at any time by all pairs of aircraft that autonomously follow the distributed rule-based safety policy (see subsection 4.3) thus assuring conflict-free air traffic. Let us outline the separation standards for pair-wise motion of aircrafts for various situation cases.

a. Horizontal movement of aircraft occupying different echelons. An attribute determining minimal admissible *vertical distance* between pair of aircrafts if they are flying strictly horizontally is

further denoted by the symbol D_A (Fig. 3).

b. "Following" motion of aircrafts within the same echelon of altitude. The attributes determining separation standards for this

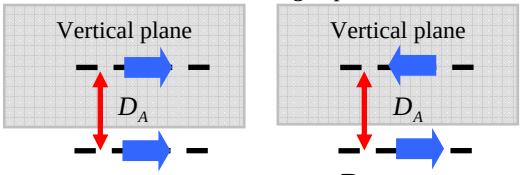


Figure. 3. Distance D_A

case are D_B –minimal longitudinal distance measured along the axis line of the legs and D_C –minimal distance between trajectories of aircrafts measured in directions orthogonal to the

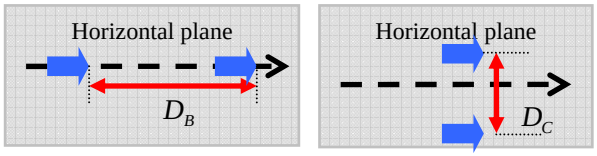


Figure 4. Distances D_B and D_C

longitudinal axes of aircrafts (Fig. 4).

c. Transversal motions of aircraft occupying the same altitude echelon It is said that the aircrafts are moving along the cross-cut trajectories if the angle value between the trajectories in horizontal plane is more than of 70 and less than of 110 degree (Fig. 5). The attribute determining the separation distance between such aircrafts is denoted as D_D. It represents the distance from an aircraft to the trajectory crossing point when one of the aircrafts has reached the crossing point.

d. Head motion of aircraft one of which is changing the altitude

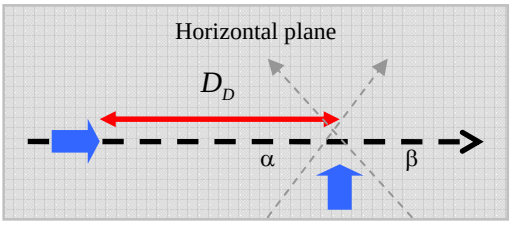


Figure. 5. Distance D_D

echelon. It is said that aircrafts have *head motion* if one of them is moving horizontally while the other one is climbing or descending with a vertical speed V_A if the angle between the course of horizontally flying aircrafts and projection of the course of the other aircraft onto horizontal plane is more than of 110 degrees. The distance D_E corresponds to horizontal distance between aircrafts when one of them has reached the trajectory crossing point. Two cases are to be distinguishes here: (1) the aircraft that earlier reached the crossing point is the one changing the echelon; (2) the aircraft that earlier reached the crossing point is the one flying horizontally. The difference between these cases is that D_E in the first case has to be greater than in the second case. Denote corresponding values of D_E as D_{E1}, and D_{E2} respectively (see Fig. 6 and 7 respectively). It is important to note

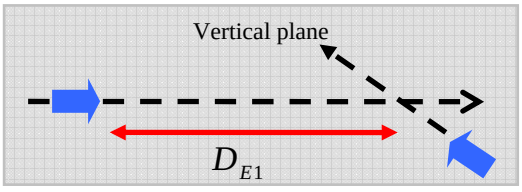


Figure 6. Distance D_{E1}

that admissible values of D_{E1} and D_{E2} depend on the vertical speed V_A of the aircraft changing the echelon.

Generally, admissible values of distances D_A, D_B, D_C, D_D, D_{E1} and D_{E2} depend on different air traffic–related situation

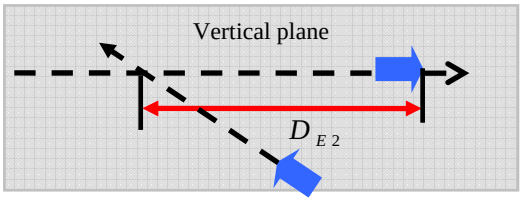


Figure 7. Distance D_{E2}

attributes. The following admissible values of these distances have been assumed:

- D_A = 0.3 km;
- D_B = 10 km in the arrival zone and 5 km in approach one;
- D_C = 10 km in the arrival zone and 5 km in approach one;
- D_D = 20 km in arrival zone and 10 km in approach one;
- D_{E1} = 30 km if V_A <10 m/sec and 60 km otherwise;
- D_{E2} =15 km if V_A < 10 m/sec and 30 km otherwise.

The same attributes are used to represent separation standards between normal aircraft and the abnormal one (hijacked, technically-challenged, etc.). Moreover, the same policy providing safety of normal aircrafts in the presence of a hijacked one is used.

3. TYPICAL BEHAVIOR PATTERNS OF NORMAL AND HIJACKED AIRCRAFTS

Existing model of an aircraft movement intended for landing or take-off comprises the typical behavior patterns and negotiation procedures with corresponding ATCO as it is described below.

a. Landing: Entry into airport airspace

As the aircraft is approaching the arrival zone, its pilot informs the ATC operator of the corresponding sector of the arrival zone about the intended altitude and entry point of arrival. Depending on the situation, the pilot does or does not receive the approval of his intention and the assigned arrival movement scheme.

b. Landing: Behavior patterns within arrival zone

Within arrival zone, aircraft is moving along the axes of legs constituting the assigned arrival scheme. During the movement, the aircraft is passing through the arrival zone points, exit points of the previous legs and entry points of the subsequent ones.

Every arrival scheme point is assigned the admissible altitude echelons and therefore, while *passing through a scheme point*, the aircraft is permitted to pass through this point using one of the echelons assigned by the arrival zone ATCO. At some of these points, the holding areas exist. While approaching such a point, the aircraft either receives permission to enter the subsequent leg, or a request to enter the corresponding holding area where aircraft has to wait for ATCO permission to continue movement along the next leg of the assigned scheme.

While moving inside legs, the aircraft holds assigned altitude echelons and changes them during descending according to designation made by ATCO.

When one *aircraft has to pass another one* (e.g. due to the difference in admissible speeds for aircrafts of different classes) both have to deviate from the leg axis at predefined distances to the different sides. When passing is completed both aircrafts have to return to the leg axis and continue the movement. An important requirement is that both aircrafts have to return to the leg axis prior the current leg exit point. For this behavior pattern the aircrafts are permitted to simultaneously perform the passing and echelon change evolutions.

When an aircraft is *moving inside a holding area* it spends some time performing several circles within the holding zone depending on the situation. Within the zone, the aircraft stays at a single altitude echelon, but it is also may descend to a lower one.

Vectoring is an important behavior pattern that violates the leg boundaries. When vectoring is completed, the aircraft has to enter a leg of the same or other arrival scheme. Every vectoring requires building a new trajectory. Typical vectoring caused by weather conditions, technical problems, etc., implies turning at 30 degrees from the leg axis in horizontal plane, flying 20 km, and returning to the former course using the same or other echelon.

c. Landing: Movement inside approach zone
Entry into approach zone requires permission of responsible ATCO. While having no permission, the aircraft has to wait inside a holding area of the arrival zone. *Movement inside the approach zone* is carried out according to the designated approach scheme.

If, due to a reason, an aircraft that entered an approach zone *cannot perform landing*, it continues movement using a scheme of *missed approach* linked to its approach scheme while returning to one of the landing trajectories. In any case, to entry a new (or next) landing trajectory, the aircraft needs permission of ATCO of the respective approach zone. Otherwise, it has to wait within a holding zone specifically designated for missed approach case.

d. Take-off
Prior to take-off the aircraft pilot is assigned a movement scheme, informs ATCO about expected take-off time and *waits for the permission* for take-off. Depending on the current air traffic situation, the permission may be received with some delay. *Inside the approach zone* the aircraft uses the predefined departure scheme. *While moving inside arrival zone*, the taking-off aircraft uses the predefined departure scheme ending at the selected exit point of departure from the airport airspace.

e. Behavior patterns of hijacked aircraft
An important difference between the motion patterns of normal and hijacked aircraft is that the latter may ignore commands of ATCO and violate the rules of air traffic within AAS by not using predefined legs, waiting zones, entry and exit points, violating the predefined echelon altitudes, etc. In the paper, a limited set of typical behavior patterns are simulated. They include (a) motion of hijacked aircraft within the arrival zone, and (b) patterns using "broken line" trajectory. Nevertheless, even such geometrically simple patterns significantly complicate the air traffic control task.

4. ATC ORGANIZATIONAL PRINCIPLES

4.1. Existing Organizational Principle of ATC

Organizational principles of ATC determine how air traffic control functions are divided between ATCO and a pilot operating within AAS. Thus, two main roles of the ATC domain organization, "*pilot*" and "*air traffic operator*"[1], are defined. According to the currently existing ATC organizational principles, the main control operations performed by ATCO are:

a. Commands to aircraft approaching to the airport airspace:
A. Permission to entry into the AAS.

b. Commands to an aircraft operating within arrival zone:
B. Permission to transit into next leg.
C. Directives to transit into lower altitude echelon.
D. Coordinating evolutions of aircrafts in the passing situations.
E. Permission to entry the approach zone for the subsequent landing.
F. Changing the aircraft speed.
G. Performing vectoring.

c. Commands to an aircraft operating within approach zone:
H. Permission to a taking-off aircraft to take-off.

Unfortunately, such ATCO–centered organization of ATC is too inflexible and unable to support a significant increase of air traffic intensity and safety.

4.2. Advanced ATC Organizational Principle

The proposed ATC organization is focused on achieving openness and autonomy of ATC system supported by distributed safety policy for conflict resolution and P2P interaction–based autonomous coordination of aircrafts' movement. Like the existing ATC organization system, the main participants of the proposed one are *Aircraft pilots* responsible for autonomous solution of the tasks A, B, C, D, F and G within the approach zone and *Air traffic operator* whose responsibilities address control functions regarding tasks E in the arrival zone, and the tasks H within approach zone. Two important issues constitute the basis of control functions of the aircraft pilot role: 1) organization of information exchange and 2) safety policy determining the aircraft's autonomous behavior. Let us consider these issues.

Autonomous behavior of an aircraft in constrained environment, i.e. the airport airspace, assumes that each aircraft has to possess the information on current positions, courses and anticipated movement plans of other aircrafts operating within AAS, at least those that potentially may violate the separation standards. In the proposed ATC organization, this information is gathered by *Aircraft pilot* on the P2P basis. The list of potential peers may include only the aircraft that follow the same or overlapping arrival schemes, and this fact can be used for a significant decrease of information exchange. The latter is achieved via decomposition of the aircraft of the arrival zone in *independent groups.* The formation of groups and, hence, the *decomposition* may be achieved *on the sector basis.* Every sector is composed of the sequence of legs between two consequent entry points of the holding zones and the name of sector's exit point is assigned as

[1] This notice is important since "*role–based*" Gaia methodology [13] for multi-agent ATC system design is below used.

identifier of the sector. Thus, every sector is composed of sequence(s) of legs belonging to one or several arrival schemes that end in particular entry point of holding zone and start in an exit point of the previous holding zone. Therefore, the total count of the *sectors* is equal to the total count of holding areas. Note that the whole approach zone is considered as a sector.

The aircrafts operating within the same sector constitute a *group*(*Sector*). Since arrival scheme may include several sectors, each aircraft can belong to several groups depending on its behavior strategy and current air traffic situation. That is why aircraft groups may be overlapping. Each group *group*(*Sector*) is assigned the name of the corresponding sector. To further decrease the computational complexity and communication overhead, it is assumed below that, every aircraft, at any time instant *t*, takes into account potential conflicts within two groups, namely the sector of its current location id_1 and the next one it plans to transit to, id_2,. Thus, in the developed approach, to compute its own conflict–free behavior in the arrival zone, the aircraft relies upon information exchange with aircrafts of no more than two groups determined on P2P interaction basis.

Table 1. Information to exchange among aircrafts of a group

Aircraft's related data	
Aircraft	*<Aircraft's identifier>*
Class	*<Aircraft's class>*
Current sector	*<id of sector in which aircraft is currently located>* ,
Next sector	*<id of sector into which the aircraft has to overcome next>*
Update time	*<time of information update>*
Movement related data	
On Altitude	*<Current altitude echelon>*
To Altitude	*<Next selected altitude echelon>*
In holding area	<Holding area usage>
Information about transition into the next sector	
Transition point	*<Name of entry point>*
Transition time	*<Next sector transition time>*
Transition status	*<Intention /Decision>*
Approach	<Flight Scheme within the next zone> (For the aircraft of the approach zone)
Schedule delay	
S-Delay	*<Accumulated delay>*
F-Delay	*<Total accumulated delay of the flight>*

According to the proposed ATC organizational structure, within the arrival zone, the aircrafts have to *autonomously* solve the tasks A, B, C, D, F, G using P2P communication both for group discovery and conflict resolution if any. The information circulating among aircrafts of the same group is given in Tab. 1.

Let us note that every aircraft has to possess the information about all aircraft in the group(s) to which it belongs at the current and next movement step. Note that the above data are updated and sent to group peers when aircraft is making a decision from the set {A, B, C, D, F, G}. After receiving the updated information, the aircraft software has to assess the impact on safety of its own planned movement. If a conflict occurs, to avoid it, the aircraft starts P2P negotiation implementing the safety policy (see below).

4.3. Outline of Safety Policy and Deconfliction Algorithm

Safety policy is a set of rules determining priorities of the aircraft of the same group to be addressed by the current movement plan. It is implemented as a distributed deconfliction algorithm. To reduce the complexity of the deconfliction task, the algorithm is performed in two steps. At the first step, every aircraft computes its own pair-wise priorities regarding to all peers and at the next step the aircraft of the highest priority is automatically "granted" permission to use the sector's "resources" of the arrival zone (legs, holding zones) according to its current plan. Then these steps are iteratively repeated by the rest of the group aircrafts while taking into account the resources already reserved by aircrafts of the same group that have higher priority. Note that described safety policy concerns only landing aircrafts. Presently, the safety policy for taking-off aircrafts is still being developed.

The general approach to priority assignment to normal aircraft is as follows. First, relative orders for any pair of the airspace sectors are introduced. They are determined as "geometrical" precedence of the sectors of airport airspace starting from entry points and ending at runways. According to a general rule, an aircraft that entered through a particular entry point of its trajectory proceeds along a uniquely predefined sequence of the sectors of the arrival zone. Thus, it is said that sector X_i immediately precedes the sector X_j ($X_i < X_j$) if the former is the next sector in the aircraft landing scheme. This relation determines the order in which sectors and, therefore, groups are deconflicted. At the next step, the aircrafts of the same group are prioritized according to the set of safety policy rules given below.

Rule 1

If $Y_1 \in group(X_1)$ and $Y_2 \in group(X_2)$ and $X_1 < X_2$ then aircraft Y_1 is of higher priority than aircraft Y_2

Rule 2

If sector X_i is a sector of the arrival zone and both aircrafts, Y_1 and Y_2, belong to the sectors that immediately precede the sector X_1 then their priorities are determined either by *Rule 3* if no hijacked aircraft exists in AAS or *by Rule 4* otherwise.

Rule 3

Let two aircrafts, Y_1 and Y_2, have the exit times from the current sector $t_{Sector\ Exit}(Y_1, t_c)$ and $t_{Sector\ Exit}(Y_2, t_c)$ respectively scheduled. If $t_{SectorExit}(Y_1) < t_{Sector\ Exit}(Y_2)$ then priority of the aircraft Y_1 is higher than Y_2 one.

Let us note that, at a time instant, the aircrafts may belong to different sectors but plan to use the same sector as next one. In this case the *rule 3* is also applicable.

If two aircrafts occupy different sectors but potentially may conflict with hijacked aircraft then special functions $Conf(t, SectorX_1)$ and $Conf(t, SectorX_2)$ representing "the degree of conflict" are introduced for those sectors. Values of these functions are used as the arguments of the next rule,

Rule 4

If normal aircraft $Y_1 \in SectorX_1$ and $Y_2 \in SectorX_2$, and $Conf(t, SectorX_1) > Conf(t, SectorX_2)$ then priority of the aircraft Y_1 is higher than priority of the aircraft Y_2.

It is worth to note that in practice more rules and more aircraft attributes have to be used in the ordering process, e.g.,

- Class of aircraft (it determines the range of aircraft's speed depending on the flight altitude and therefore influences of aircrafts' preferences);
- Current echelon occupied by aircraft;
- Current deviation of the aircraft attributes from the scheduled ones;
- Fuel status, etc.

The mandatory requirement to any set of rules is that they have to provide conflict–free movement of the aircrafts. The selected rule sets can differ in resulting efficacy of ATC according to a criterion of multiple ones. A natural criteria, for instance, of air traffic control is maximal averaged capacity in terms of the total count of landing and departing aircrafts subject of safety requirements. But this task is out of the paper scope so far.

5. ARCHITECTURE OF MAS ATC: FORMAL SPECIFICATION

For design of Multi-agent ATC and Airspace Deconfliction System (ATC–AD MAS), extended Gaia methodology [10], was used. According to it, MAS architecture is specified in terms of a diagram representing its *meta–model* (Fig. 8) and "*liveness expressions*" specifying architecture in more detail. The meta–model specifies MAS architecture in terms of the roles to be assigned to agent classes, active software entities and the interaction of these components. Liveness expressions specify the role scenarios in different use cases. For the system in question, each role is mapped onto particular agent class. Therefore, the terms "*role*" and "*agent class*", in this application, may be denoted by the same identifier, however, the term "agent class" is below mainly used. In the developed system, three agent classes and one active software entity are introduced as described below:

- *Pilot assistant agent class* (*PA-agent class*); each agent of this agent class assists to the pilot of particular aircraft in autonomous ATC and in deconfliction situations;
- *Air traffic control operator agent class* (*ATCO-agent class*); each agent of this class assists the ATCO in decision making within the approach zone (sector);
- *Hijacked aircraft agent class* (*HA-agent class*); each agent of this class simulates and monitors of hijacked aircraft behavior.

Simulation server plays here the role of an active program entity. It is intended for simulation and visualization of real time situation in the airport airspace. It initiates real time events reflecting the results of operation of entities involved in air traffic and air traffic control. Simulation server also provides the interface to user; in particular, it supports the following functions:

- Visualization of the current air traffic situation within the airport airspace;
- Generation of the hijacked aircraft trajectory;
- Visualization of conflicts occurring between pairs of normal aircrafts and between normal aircrafts and hijacked ones.

According to Gaia methodology used in this development, formal specifications of agent classes (roles) are done in terms of liveness expressions. They specify the basic scenarios of agent classes' behavior in various tasks (use cases). In particular, specification of *PA-agent class* consists of 14 liveness expressions (*Initialization*, *Simulation cycle*, *Aircraft grouping*, *Arrival timetable monitoring*, etc.) presented in Fig. 8. *ATCO agent class* model consists of two liveness expressions, *Query* and *Permission.* Agent class simulating movement of the hijacked aircraft includes also two *liveness expressions* that are *Initializatio*n and *Trajectory forecasting*.

Specification of target system architecture in terms of liveness expressions is developed in detail but omitted in the paper due to lack of the space. These descriptions are done in context of the *Use cases* in which these liveness expressions are involved. In the developed model, seven such use cases (tasks of the target system), *U1–U7* (see Fig.8), are identified:

- (*U1*) Initialization of *PA-agent* class instances and initialization of the agent instance simulating the movement of the hijacked aircraft;
- (*U2*) Execution of simulation cycle;
- (*U3*) Grouping of aircrafts (instances of *PA-agent* class) that is used to reduce the overall information exchange traffic and to achieve the reduction of computational complexity of the airspace deconfliction algorithm;
- (*U4*) Autonomous planning of its own movement within the arrival zone by *PA-agent* class instance representing individual aircraft;
- (*U5*) Re–planning of own movement within the arrival zone by *PA-agent* class instances in order to avoid conflicts with normal and hijacked aircrafts;
- (*U6*) Normal aircraft's take-off control;
- (*U7*) Control of the arriving aircrafts during their movement within the approach zone (from the time when aircraft requests permission to entry the approach zone until the landing).

While performing these tasks, corresponding agents implement behavior specified in terms of liveness expressions.

Two types of liveness expression *Initiation* are used:

- Agent initiates running of a liveness expression in response to input messages, e.g. (Fig. 8), *PA- agent* class instance initiates running of the liveness expression "*Take-off*" when receives the message from *ATCO–agent* with "*Take-off permission*".
- Agent itself initiates a liveness expression as a result of its proactive emergent behavior, i.e. as a result of occurrence of some event(s) within the environment. An example is the initiation of *PA-agent* class instance liveness expression "*Grouping*" after transition of the normal aircraft from a sector to another one.

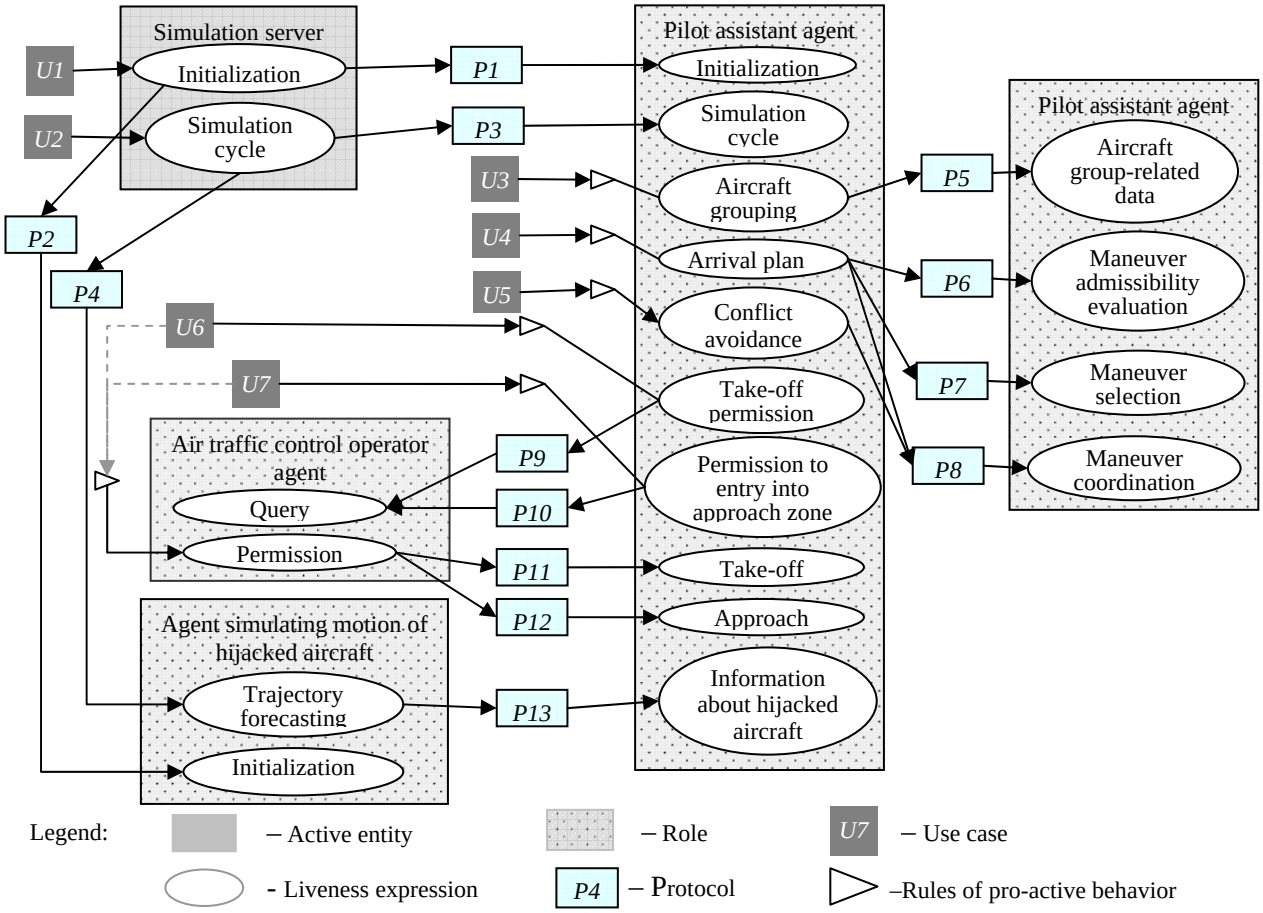


Figure 8. Meta-model of Multi-agent Airspace Deconfliction System

6. GRAPHICAL USER INTERFACE

Main window (Fig. 10) is used for visualization of the followings:

- Airport airspace topology (horizontal projection);
- Current positions of the aircrafts at the simulation time instant, and
- Detected conflicts.

If, at the current time instant, a conflict between a pair of the aircrafts is detected then this conflict is depicted by red line connecting the conflicting aircrafts when it exists. Interface also depicts some "statistics" of the detected conflicts. For this purpose, the sequence of the executed simulation cycles is depicted in the lower part of the window; the cycles exhibiting conflict(s) are depicted in red color whereas conflict–free cycles are depicted in green color.

The program component supporting graphical user interface operation checks separation standards while doing this independently of the ATC MAS and depicts the results; this functionality may also be used for agent behavior validation.

Graphical Interface Options assume image scaling, optional filtering of data visualized on image, and altitude-based filtering of data represented in horizontal projection mode.

Simulation control includes such functional capabilities as scaling of the simulation speed; simulation process interruption in the case if a conflict happens, simulation mode control (continuous or cycle-based simulation; detection of the time instant when hijacked aircraft appears. Additionally, *Simulation mode control* assumes the selection of a movement scheme and visualization of aircrafts' movements in vertical plane and manual input of hijacked aircraft movement data.

An example of visualization of an arrival scheme-related situation in vertical plane is given in Fig. 10. In this figure, the arrival scheme corresponding to sequential passing through points

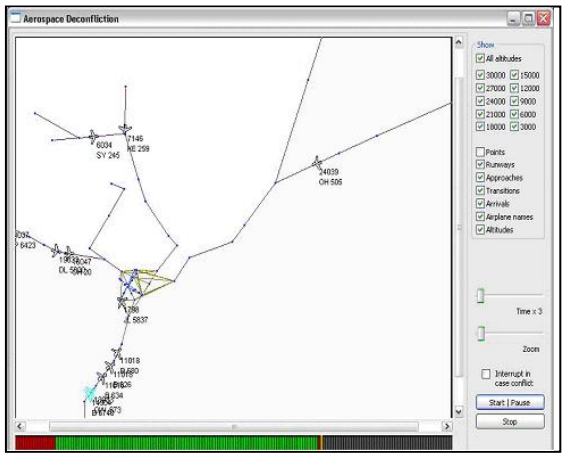


Figure 9. Main window

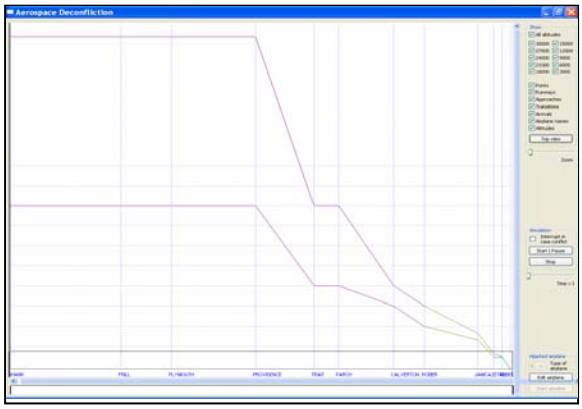


Figure 10. Visualization of movement scheme – related situation in vertical plane

HANK, FRILL, PLYMOUTH, PROVIDENCE, TRAIT, PARCH, CALVERTON, ROBER (see Fig. 9) of the JFK (New York City) airport is depicted. In this picture, the trajectories of the aircrafts situated in the "proximity" of the legs (at distances less than 5 km) constituting this scheme are depicted.

On the background of this window, horizontal lines depict the echelons ("every third" one is depicted in order to make the picture readable). In particular, 10 echelons presenting altitudes of 3000 feet, 6000 feet etc. till the altitude 30000 feet)are depicted in Fig. 10. The lines represent altitude boundaries of admissible echelons for corresponding legs assumed by the JFK AAS topology. Since flexible selection of the echelons is considered as a basic strategy of ATC, this graphical interface may be also utilized for graphical validation of the agents' behavior and the overall deconfliction algorithm.

Trajectory of the hijacked aircraft movement is specified prior to the simulation process. This specification is done in two steps. In the first step, its trajectory is specified in horizontal projection. This is done via selection of a sequence of the points of the trajectory. During the second step, the trajectory is specified in vertical projection. For this purpose special interface (not shown in the paper) is used. It defines the altitudes of the points selected at the first step of the procedure. The time instant corresponding to the appearance of the hijacked aircraft is defined manually during the simulation procedure. The hijacked aircraft is selected from the database presenting the aircraft speed as the function of the altitude and aircraft class.

Specification of particular air traffic situation is based on the use of real life timetable of arrival and departure. This is done using an editor of graphical user interface. In the developed version, a timetable of the JFK airport of New York City was utilized.

8. CONCLUSION

The paper offers a model, multi-agent architecture, formal specification methodology, and a software prototype implementing ATC system. The main paper contribution is that it suggests a feasible realization of ATC system featuring such important properties as openness and autonomy based on social rules, distributed safety policy for conflict resolution (collision avoidance), as well as on predictive analysis and P2P interaction–based autonomous coordination of aircrafts' motions. Design of the MAS in question is accomplished using Multi-Agent System Development Kit (MASDK 4.0), the recent version of the software tool that is being developed by the authors since 2000. This software tool implements extended version of Gaia methodology [10] in purely graphical style [5].

The further efforts are need to bring the model of air traffic control to the reality. Many aspects should be taken into consideration. The most important of them is the necessity to enrich the set of the aircrafts' behavior patterns while including vectoring, taking-off patterns, "missing approach" behavior, pattern intended to change the target airport, influence of weather conditions. It is also necessary to enrich the behavior patterns of hijacked aircrafts.

The set of rules determining behavior of aircrafts according to distributed security policy has also to be enriched and investigated from two view points, computational efficiency and quality of the air traffic control, e.g. from capacity view point. They should constitute the topics of further research. Nevertheless, the paper ideas will basically be preserved.

ACKNOWLEDGMENTS

This research was conducted under a contract from the Information Institute of the AFRL at Rome NY, USA. The authors are grateful to Mr. John Graniero of the AFRL for his support of this effort. The authors are also grateful to V.V. Kupin (St. Petersburg University of Civilian Aviation) for his thoughtful consultations on the existing air traffic control and challenges.

REFERENCES

[1] AEEC: Engineering Standards for Avionics and Cabin Systems, http://www.arinc.com/aeec

[2] http://space.iias.spb.su/ap/, 2007

[3] Draft 2 of Supplement 19 to ARINC Specification 424: Navigation System Data Base. Aeronautical Radio, INC, http://www.arinc.com/aeec

[4] Gorodetsky, V., Karsaev, O., Samoylov, V. ,Serebryakov, S. 2007. P2P Agent Platform: Implementation and Testing. The AAMAS Sixth International Workshop on Agents and P2p Computing (AP2PC 2007), Honolulu, 2007, pp. 21-32.

[5] Gorodetsky, V., Karsaev, O., Samoylov, V., Konushy, V., Mankov, E., Malyshev. 2005. A. Multi Agent System Development Kit. In R.Unland, M.Klusch, M.Calisti (Eds.) Software Agent-Based Applications, Platforms and Development Kits. Whitestein Pub.

[6] Kephart J. 2007. Multiagent Systems for Autonomic Computing. Invited talk at AAMAS'2007.

[7] FIPA P2P NA WG6: Functional Architecture Specification Draft 0.12. http://www.fipa.org/subgroups/P2PNA-WG-docs/P2PNA-Spec-Draft0.12.doc

[8] Tumer, K. and Agogino, A. 2007. Distributed Agent-Based Air Traffic Flow Management. Sixth Intl. Joint Conf. on Autonomous Agents and Multiagent Systems, 330-337.

[9] Tomlin, C., Pappas, G., and Sastry, S. 1997. Conflict resolution for air traffic management: A case study in multi-agent systems". IEEE Transactions on Automatic Control.

[10] Wooldridge, M., Jennings, N., and Kinny, D. 2000. The Gaia Methodology for Agent-Oriented Analysis and Design. International Journal of Autonomous Agents and Multi Agent Systems, 3(3), 285-312.

Author Index: