Decentralized Insertion Heuristic with Runtime Optimization for On-demand Transport Scheduling

Alaa Daoud and **Flavien Balbo** and **Paolo Gianessi** and **Gauthier Picard** [1]

Abstract. On-Demand Transport (ODT) systems have attracted increasing attention in recent years. Traditional centralized dispatching can achieve optimal solutions with NP-Hard complexity, making it unsuitable for online and dynamic problems. Decentralized approaches can achieve fast feasible solution at run-time with no guarantee on the quality. Starting from feasible not optimal solution, we present in this work a new decentralized, insertion heuristic algorithm to solve a special case of the dynamic Dial-A-Ride-Problem (DARP) as an ODT system, in which vehicles communicate via Vehicle-to-vehicle communication (V2V) and take decentralized decisions. In our solution model we add an optimization phase in order to improve the quality of global solution achieved by decentralized decision at run-time by exchanging resources between vehicles. We evaluate and analyze the promising results of our contributed technique on synthetic data for taxis operating in Saint-Étienne city, against a classical greedy approach.

1 Introduction

The concept of On-Demand Transport (ODT) was formulated for the first time around 1990 as a solution to the growing disaffection of potential users of public transport, especially at night [2]. There exist similar terms in the literature describing several transportation systems that could be similar in some characteristics and differ in others; e.g. the term Demand-Responsive-Transport (DRT) refers to the same concept. Seeking simplicity, here we only use the term ODT:

Definition 1 *"On-Demand Transport is a transit mode consisting of passenger vehicles, vans or small buses that respond to the calls of passengers or their agents to the transit operator, who then sends a vehicle to collect passengers and transport them to their destinations."[13]*

The allocation problem is one of the most studied optimization problems in the literature [5]. It can be attacked by linear programming because its relaxation of formulation admits optimal integral solutions. In reality, if a central dispatcher exists, it requires vehicles to have continuous access to their portal via the global communications infrastructure (e.g. 4G), which is expensive and can cause a critical bottleneck on the portal side.

The computational complexity of the problem, which extends the NP-Hard traveling salesman problem (TSP), makes it difficult for the central dispatcher to manage the dynamics of the problem during execution (online requests, variable fleet size or heterogeneous fleets, traffic problems and other environment dynamics). Therefore one may expect that decentralization can cop off with these problems. The

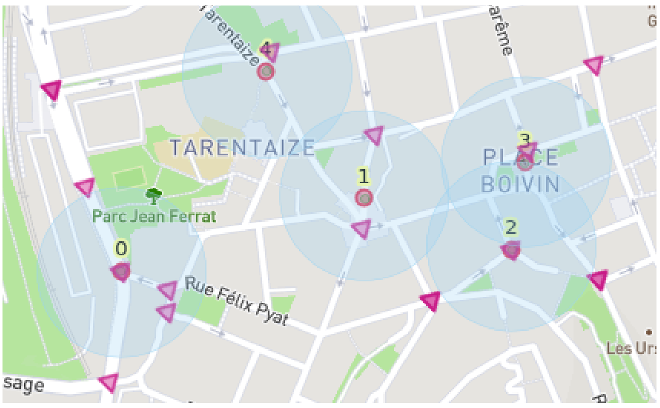

Figure 1. A sample AVFAP problem, with demand sources (triangles) and taxis (circles) with their respective communication ranges (in blue).

insertion heuristics proposed by [16] is a popular method for solving a variety of scheduling and routing problems. It can be used as a method to quickly find a feasible solution (with no guarantee of solution quality). In Vehicle Routing Problem (VRP), it creates the solution by repeatedly inserting unscheduled demands in a partially constructed route or as the first demand in a new route. Local search with the k-exchange neighborhoods (k-opt), is one of the most commonly used heuristic methods for problems inherited from TSP. k-opt is a path improvement algorithm, where at each planning phase, k steps from the current plan are replaced by k steps to get a cheaper path [12].

In this work, we propose the Autonomous Vehicle Fleet Allocation Problem (AVFAP), which extends the traditional DARP with the usage of fleet of autonomous vehicles, replacing the central dispatcher with the coordination between each other via Peer to Peer (P2P) communication in order to take decentralized decision, as illustrated in Figure 1. V2V communication via Dedicated Short-Range Communication (DSRC) provides low latency, fast network connectivity with a range of communication up to 300 meters [6]. Having limited communication ranges, each vehicle is only aware of a subset of demands, and communicates with entities (vehicles or demand sources) if they are in the range of each others. We more precisely propose a new decentralized heuristic for the AVFAP, named *Pull-demand*, benefiting from the fast responsiveness of insertion heuristics, and the good results gained by k-opt optimization.

The paper is organized as follows. In Section 2, we overview the efforts done in literature to address related ODT problems. Section 3 expounds the model we consider for the decentralized resource allocation problem for autonomous vehicle fleets with an illustrative scenario showing the main components of the ODT system. Then the contribution for fast decision is presented in Section 4 and an opti-

[1] Univ Lyon, Mines Saint-Étiennee, CNRS, Laboratoire Hubert Curien, UMR 5516, F-42023 Saint-Étienne, France, email: name.surname@emse.fr

Copyright © 2020 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR-WS.org/Vol-2701/paper_7.pdf

mization protocol is detailed in Section 5. Experimental parameters, evaluation and results are detailed in Section 6. Finally, we conclude the paper in Section 7 with some perspectives.

2 Related Works

Traditional approaches for ODT consider centralized dispatcher architecture like in [7, 15] or decentralized MAS to reduces problem complexity with a central coordinator like in [8, 11]. While there exist several approaches for decentralized decisions and self-coordination, like the work of [9] to introduce a multi-agent bid-based real-time scheduling solution in fully decentralized settings, in which each vehicle represented by an agent that can negotiate via radio channels with flexible decision criteria. A pattern recognition algorithm is used to predict most likely locations for the next demand using agent-based data mining to recommend movements to these locations.

Investigating the applicability of genetic programming (GP) for developing decentralized MASs that solve dynamic DARP, [17] presents a method to automatically generate a MAS that can solve the DARP for a specific set of scenarios. GP is used to generate a heuristic that is effective in solving the DARP compared to centralized solutions. The best result achieved with this approach is by planning only one demand in advance by vehicle, which maximize the agent's local interests (greedy) and produce a feasible solution very fast.

An agent-based model based on simulated events for the real taxi market was proposed by [10], where supply and demand matching depends on event-based interactions. According to their conclusion, one of the main limitations of [10] model is the assumption of uniform distribution of demand in the service area.

To address the uncertainty caused by the dynamic nature of online demands, with ignoring the time required to execute a planning algorithm, [1] proposed using a deterministic rolling horizon solution approach, in which plans are drawn up using all known information in a planning horizon, that is "rolled" forward to include more known information.

Based on vehicle coordination via message passing using P2P communication, In our previous work [14] we proposed the Online Localized with Communication Constraint Resource Allocation (OLC^{2}RA) for concurrently solving the allocation problem over a fleet of autonomous taxis, in which a vehicle decide its next destination (scheduling only one demand in advance). On the contrary, ALMA decentralized heuristic proposed by [5] is completely decoupled and does not require communication between the participants. They demonstrate an upper bound of the speed of convergence which is polynomial to the desired quantity of resources and competing agents per resource; In the realistic case where the mentioned quantities are limited whatever the total number of agents/resources, the convergence time remains constant as the total size of the problem increases. However, in [5] conflict detection still requires communication with other vehicles, resources or a central entity to enable resources to share information about their status, such as the blackboard coordination mechanism.

So far and to the best of our knowledge, efforts done in ODT planning domain provide either optimized solutions ignoring the execution time, or provide very fast, just feasible solution to respond quickly to dynamic online demands. In this work, as optimal solution for this dynamic problem is unachievable with rational computational time, our proposal combines the benefits of these approaches in one heuristic that provide fast auction based response (fast feasible solution), and at runtime this solution is improved gradually with demand-exchange rescheduling.

3 AVFAP Problem Model

The Autonomous Vehicle Fleet Allocation Problem (AVFAP) extends the traditional DARP by considering a fleet of autonomous vehicles that coordinate without a central dispatcher. The vehicles make decentralized decisions based on information exchanged via Peer to Peer (P2P) communication as illustrated in Figure 1.

The city map graph, shown in Figure 2, consists of nodes representing geographic locations and edges representing the road connections between these locations. A fleet of autonomous vehicles is distributed throughout the city, each vehicle has a set of properties whose values are constant (capacity, cost and average speed) or variable (location, schedule) because they are time-dependent. Passengers emit demands from different locations (which we will call sources later). Each one takes the form of a request that defines: the pick-up and delivery locations associated with the desired service time window. We define a solution as a schedule for each vehicle that meets the demands by satisfying their constraints, minimizing passenger waiting time and minimizing vehicle travel costs.

The vehicles communicate by broadcast via an ad-hoc Vehicle-to-vehicle (V2V) communication network, where the communication range is limited. Each vehicle that receives new information broadcasts it again and the vehicles are thus connected by transitivity within the communication range.
In this context, a vehicle is not aware of requests outside its communication area. The vehicle's belief evolves during the execution time, as the vehicle moves around, receives new requests from request sources, meets other vehicles and exchanges messages with them.

The AVFAP model is defined as:

$$\begin{aligned} AVFAP &:=< M, V, D, T > \\ M &:=< G, w > \\ V &:= \{v_1, v_2, \dots, v_n\} \\ D &:= \{d_1, d_2, \dots, d_m\} \\ T &:= \{t_0, t_1, \dots, t_{end}\} \end{aligned}$$

where, M defines the urban infrastructure map (locations, roads and distances); the offer is represented by a V fleet of n autonomous vehicles; D defines a dynamic set of passenger demands that occur at the time of execution; and T defines the time horizon within which vehicles must respond to passenger demands. We define time T as a discrete set of $ticks$.

3.1 Urban Infrastructure Map

The urban infrastructure map is defined by a weighted directed graph M, as shown in Figure 2.

$$\begin{aligned} G &:=< N, E > \\ w &:= \{w_{e_1}, w_{e_2}, \dots, w_{|E|}\} \end{aligned}$$

G is a directed connected graph where N is the set of nodes, E is the set of edges between nodes. The valuation function w associates each edge e with the value w_e based on a measure of temporal distance (for example, average driving time in minutes), which will be used to calculate the operational costs of vehicle trips.

3.2 Vehicle Properties

Each $v \in V$ is defined by

$$v :=< capa, cpd, range >$$

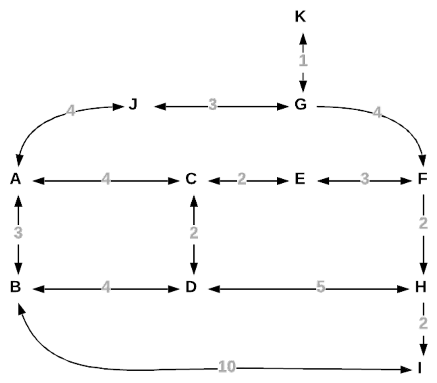

Figure 2. A sample AVFAP city infrastructure

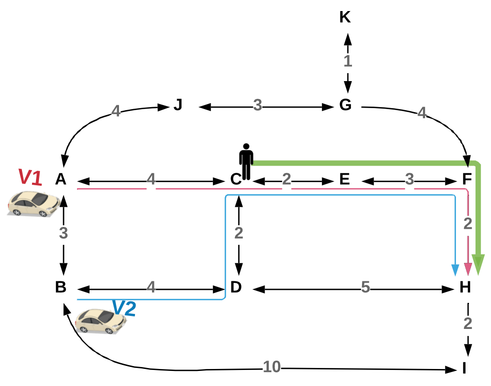

Figure 3. V_1 wins the auction to serve d_1 from C to H

and has a set of dynamic properties

$$v\_location : V \times T \rightarrow N \cup E$$
$$free\_seats : V \times T \rightarrow \mathbb{N}^+$$

where $capa$ defines the total number of passenger seats in the vehicle, cpd defines the cost of the vehicle per unit of distance, and $range$ defines the communication range within which the vehicle can communicate with other entities. At any given time $t \in T$ a vehicle $v \in V$ knows its current situation: $v\_location$ defines the location of v that could be located in a node or moving through an arc; and $free\_seats$ defines the number of free seats available in v at time t.

3.3 Demand Properties

A demand $d \in D$ is defined as a request

$$d :=< required, tw, pick\_up, drop\_off >$$

where $required$ is the number of seats required; tw defines a time interval (t_{min}, t_{max}) in which the pickup event is considered acceptable; $pick\_up$ and $drop\_off$ are the origin and destination of the request, respectively. As for vehicles, at time $t \in T$, we will consider that a request $d \in D$ can also be communicated using the V2V network. The request could be issued by the customer or the infrastructure (roadside unit).

4 Auction-based Insertion Heuristic

The main objective of vehicle agents is to provide feasible schedules that maximize their utility, by minimizing the global operational cost of serving the maximum number of requests.

Given a vehicle v having a potential demand set D_v, providing a schedule for v to that satisfies all the requests $d \in D_v$ means solving a TSP to produce the best ordering of requests in the schedule. Considering the dynamic aspect of our model, we use an insertion heuristic like the one described in [16] to continuously adapt local vehicle schedules. The result of this algorithm is a set of requests, each of them is associated with the potential time at which a vehicle will be at the pick-up location. The insertion heuristic is proven to be efficient in finding feasible schedules very fast [3], but since it handles the scheduling of requests one by one, its performance is relative to the order of these demands.

In our model the agents handle request based on their priority order. The priority function is used by an agent to assign priority values to the requests he knows. Given a vehicle v, the priority function returns for a demand d a value that is inversely proportional to the cost $costDemand(d, v)$ of inserting the demand in the schedule of v: $1/costDemand(d, v)$. Thus the lower the cost, the higher the priority.
In the simplest form the cost function returns the distance of d's pickup location from v's location, thus agents assign highest priority to the request with the nearest pick-up location.

Each agent determines his schedule to maximize the value of the quality of his solution. Since several vehicles may be interested in the same request, we need a coordination mechanism to resolve these conflicts. We will use an auction mechanism for this purpose, which is one of the effective and proven ways to solve such problems [4].

4.1 Bid Criterion

When a vehicle v becomes aware of a request d, it ranks it in its queue according to the priority it has assigned to it.

At time t, v selects the first request in the queue d_s, generates a set of alternatives, each of which is a potential schedule resulting from the insertion of d_s in a feasible step of the v's current schedule. Every alternative is associated with a *cost* which is the marginal operational cost of adding this request to the schedule. The choice with the best *cost* is considered to broadcast an offer

$$Bid_v^d(t_{start}, cost)$$

with t_{start} the time of $pick\_up$ for d_s.

In this document, we consider the operational cost of a trip as the total length of its route (sum of the distances shown in Figure 2). The marginal cost of insertion is therefore the difference in path length between the initial path and the new path. Bids remain available for a specific time period t_{expire}. Thus, if the bid cost of v is less than any other bid received at $t + t_{expire}$ to serve a request d, it considers itself the winner of the auction, and updates its schedule with the new bid path.

Example 1 *The simple scenario in Figure 3 shows two vehicles V_1 and V_2 located in A and B, with empty schedules at the beginning. At time t_1, the first request is announced $d_1 :< 1, (t_{10}, t_{20}), C, H >$. Both vehicles now know d_1. V_1 can serve it by following the path $A \rightarrow C \rightarrow E \rightarrow F \rightarrow H$, so V_1 places the offer $Bid_{V_1}^{d_1}(t_{10}, 11)$. V_2 can serve it via the path $B \rightarrow D \rightarrow C \rightarrow E \rightarrow F \rightarrow H$, so issues the offer $Bid_{V_2}^{d_1}(t_{10}, 13)$. V_1 is considered a winner and adds d_1 to its schedule, so that the overall operational cost of the fleet is now* 11.

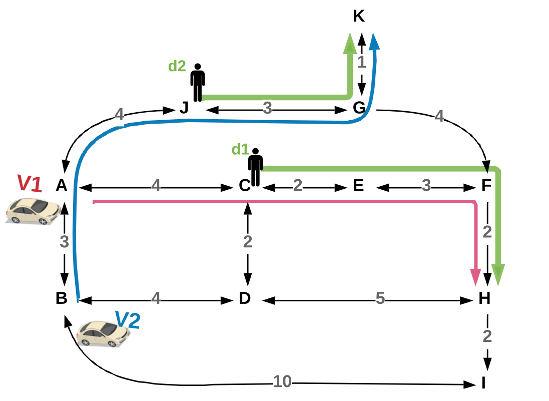

Figure 4. With no demand exchange, V_2 wins d_2 and V_1 keeps d_1

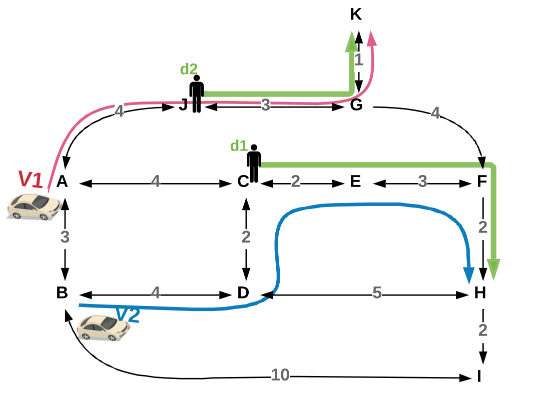

Figure 5. Global improvement when V_1 abandons d_1 to serve d_2

4.2 Abandoning Demands

In order to be able to make effective bids for new requests or to improve the solution, we propose that the vehicles exchange their planned requests.

Example 2 *Figure 4 shows a situation where the use of the bid-based insertion heuristic is very reactive but does not guarantee good scheduling. At the moment t_2 when the new request d_2 :< $1, (t_{15}, t_{40}), J, K$ > arrives, both vehicles are aware of it and place their possible bids. In the absence of any exchange capacity, V_1 still has d_1 in its schedule (with an initial cost of* 11*), so the best offer it can place is to serve both requests with a marginal cost of* 14*. While V_2 places the winning bid $Bid^{d_2}_{V_2}(t_{15}, +11)$, it adds d_2 to its schedule, and the overall cost becomes* 22*. Note that in this case, serving d_2 with V_1 and letting V_2 take care of d_1 results in an overall operational cost gain of* 21*, but this solution is never realized because d_1 is already scheduled on V_1.*

5 Improving Solution Quality

In the following we propose a local optimization protocol to improve the quality of the solution for cases like Figure 4 and the Example 4.

This protocol is based mainly on the k-exchange neighborhoods (k-opt) which is a path improvement algorithm, where at each planning phase, k steps from the current plan are replaced by k steps to get a cheaper path [12]. In the simplest settings k is set to 1 so the k-opt becomes 1-opt which means vehicle can exchange at most one demand a time

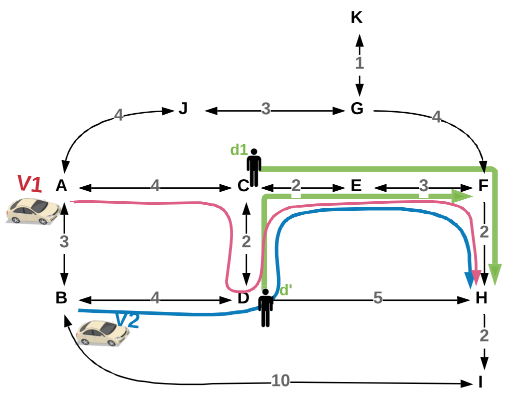

Figure 6. Even it can serve both demands V_2 can only bid for one demand at a time

Example 3 *Let's consider another case shown in Figure 6, where V_1 has d_1 in its schedule, V_2 has empty schedule, and a new demand is announced d' :< $(t_{10}, t_{20}), D, F$ >. Vehicle bids are thus $Bid^{d_1,d'}_{V_1}((t_{10}, t_{12}), +4)$, $Bid^{d'}_{V_1}(t_{10}, +15)$ (taking into account abandoning d_1), and $Bid^{d'}_{V_2}(t_{10}, +13)$. Obviously $Bid^{d_1,d'}_{V_1}$ is the winner and V_1 takes the charge of both demands with global cost $+15$, while the optimal solution in this case is that V_2 who takes both demands which make the global cost only $+13$, but using the auctions with abandon strategies lets V_2 in this case either bid for d' alone when it is announced, or bid for d_1 alone in response of the abandon suggestion by V_1. So in this case, the optimal solution is not achievable with the aforementioned abandon strategies.*

5.1 Pull-demand Optimization Bids

Similar to the *Rolling Horizon* strategy of [1], we propose an Optimization Protocol to improve our heuristics. In the *Rolling Horizon* strategy, all vehicle schedules are considered temporary and available to be scheduled by any vehicle, unless they are considered as *committed requests* by particular events (for example, v has started serving (moving towards) d, whose remaining time to serve it is below the horizon threshold).

The application of this strategy requires that all vehicle schedules are in shared memory, so that when a vehicle v_i offers to serve a request d, it knows if it is scheduled by another vehicle v_j, and therefore if it should send its offer cost to v_j. Then v_j will calculate the gain (or loss) in operating cost by abandoning d compared to the cost proposed by v_i. If there is a gain, it agrees to abandon d and then v_i updates its schedule with d, otherwise the bid is rejected.

In our protocol, we do not use the concept of *committed request*, but a vehicle can only bid on requests that it can satisfy, so requests that are rescheduled or that do not have enough time to be rescheduled are automatically ignored by the agent. Another difference here is that we don't have a shared memory. Agents exchange information about the context of the environment and about requests through information messages.

Protocol 1 Pullldemand Optimization Protocol

Step 1 A set of new demands enters the system based on announcement time order.

Step 2 Each new request is distributed to the connected set to which the sources belong. Each agent in this set can select his potential requests from the set of requests that includes new requests, scheduled and unscheduled requests that haven't yet reached their scheduled departure time.

Step 3 The agents enter the auction to serve their potential demands in similar auction criteria of the initial phase

Step 4 Each agent searches among its scheduled requests for the one to satisfy the next tick, this request is called d_{next}. If d_{next} exists, the agent broadcasts a "$clear\_demand$" message to inform other agents that it handles d_{next}. Each receiver deletes it from their potential and known sets of requests. In addition, each agent deletes any other requests that reach their time window upper-bound because staying any longer available for rescheduling would violate their time constraints.

Step 5 The scheduled and unscheduled requests that still have time remain announced by their sources (*Step 2*). This allows better planning in the next tick, if new requests are announced, or some new agents join the connected set.

In addition, the proposal of [1], where optimization is performed periodically at a predefined frequency, while the protocol we propose should be executed in parallel with the auction-based insertion strategy to have a fast rescheduling for continuous requests. Based on shared information of the current context, the optimization protocol is executed between connected set components when any change in this set context is detected (the set of vehicles in the connected set is changed or at least one of them is newly aware of some requests which are already scheduled by others). The protocol 5.1 details this strategy.

5.2 Discussion

Given the decentralized context, the use of the insertion heuristic is very efficient in terms of response time. The temporal complexity of the basic insertion heuristic for the Vehicle Routing Problem (VRP) is of $\mathcal{O}(n^3)$[3]. This type of heuristic is often used to solve DARPs, where new incoming requests have to be continuously processed in real time and integrated into the evolving schedules of vehicles. The usage of k-opt may add local improvements to the solution provided by the insertion heuristic based on the current context. However, the «*Pull-demand*» protocol can significantly improve the quality of the solution as illustrated in the following example.

Example 4 *Let's consider another case illustrated in Figure 6, where V_1 has d_1 in its schedule, V_2 has an empty schedule, and a new request $d' :< (t_{10}, t_{20}), D, F >$ is announced. The bids of the vehicles are thus $Bid^{d'}_{V_1}(t_{12}, +4)$, and $Bid^{d'}_{V_2}(t_{10}, +13)$. It is obvious that $Bid^{d_1,d'}_{V_1}$ is the winner, and V_1 will handle the two requests with an overall cost of* 15. *With the «*Pull-demand*» protocol, d_1 and d' enters in the set of candidate requests for V_1 and V_2, so that the vehicles can make combinatorial offers: $Bid^{d_1,d}_{V_1}((t_{10}, t_{12}), 0)$ and $Bid^{d',d_1}_{V_2}(-2)$. The cost of V_2 is* 13 *and the gain of V_1 is* 15*). V_1 has nothing to change in its schedule. V_2 wins and the solution is improved with an additional optimization round. Let's look at the applied protocol, step by step.*

Step 1 *d' enters the system at t_2 and both vehicles are aware of this, V_1 wins the auction with an overall cost of* 15

Step 2 *d_1 and d' are now in the set of requests known by both vehicles, V_2 calculates the costs to serve d_1 alone (*13*), d' alone (*9*) and both requests together making* 13. *It then selects the two requests as its potential requests. V_1 has no potential request because it already has the two requests in its schedule.*

Step 3 *V_2 places a bid $Pull\_Bid^{d',d_1}_{V_2}((t_{10}, t_{12}), 13)$. For V_1 the cost to serve both requests is* 15 *so it accepts $Pull\_Bid^{d',d_1}_{V_2}$ because it causes a gain of* 2.

Step 4 *None of the requests reach their scheduled service time or the upper-bound of their time window.*

Step 5 *All known requests remain announced and available for the next potential improvement.*

6 Experimental Evaluation

In this section, we experimentally evaluate the performance of our contributed pull-demand approach, using synthetic data and Open Street Map information.

6.1 Experimental Setup

The city map of Saint-Étienne was chosen for the simulation. The structure of the graph $G =< N, E >$ including nodes, edges and a set of sources of the $S \subset N$ request is extracted from OpenStreetMap (OSM[2]). In all the experiments, we set the number of sources $|S| = 20$, having a set $E_S \subset E$ of edges, such that $|E_S| = 75$ connecting the sources, each edge has a number of points which varies according to its length and the information extracted from OSM. The distance between two consecutive points is 40 meters. We used a discrete-time transport simulator available in the *Plateforme Territoire*[3] to evaluate the proposed strategy and analyze it in terms of quality of service and gain.

A fleet V of n vehicles is distributed randomly through S at the beginning of execution. Each vehicle $v \in V$ moves from one point to another on the same edge during each simulation cycle. In our test we consider that vehicles communicate via (DSRC) with a realistic communication range of 250m, so that a vehicle can send/receive messages to/from other entities located in its range. Each vehicle v can adopt two distinct travel behaviors: either *marauding* for requests or *going to* a destination:

- `going_to` defines the state of a vehicle when it has a specific destination, i.e. a request to serve. The vehicle is either *going_to* pickup location if the request is not yet picked up, or *going_to* delivery location otherwise.
- `marauding` defines the state of a vehicle when it does not have a request to serve, i.e. at the beginning of the simulation, each vehicle marauds until it decides to serve a request. Once a passenger is dropped off, the vehicle reverts to `marauding`. In this state, the vehicle randomly moves through its neighborhood to find requests to serve.

[2] https://www.openstreetmap.org/
[3] https://territoire.emse.fr/

At each simulation cycle, 0 or 1 request is generated in a uniformly random manner. For each request, the origin and destination points are randomly and uniformly generated from all sources. The time window for the requests is generated using two constant parameters l and u for the lower and upper limits as follows $[tw_{min}, tw_{max}]$ is initialized with two uniform random values where :

$$tw_{min} < tw_{max}$$

$$tw_{min} \geq t_{actual} + l$$

$$tw_{max} \leq tw_{min} + u$$

The evaluation criteria for these simulations are mainly the number of requests satisfied as a measure of Quality of Service (QoS), the simulated profit of the solution as a measure of Quality of Business (QoB). The profit is calculated in terms of the difference between the simulated travel price and the cost.

$$profit = total\_income - total\_cost$$

where

$$total\_income = \sum_{d \in D_s} P + p * distance(d)$$

$D_s \subseteq D$ is the set of all satisfied requests, P is a fixed price (service fee) per request, p is a pricing factor per unit of distance travelled, $distance(d)$ is the total travel distance for a request d and

$$total\_cost = \sum_{v \in V} cpd(v) * total\_distance(v)$$

In our tests, we consider the vehicles to be identical in terms of $cpd(\cdot)$ travel cost and p pricing factor. We set $P = 1.5$, $p = 2$ and $cpd(v) = 1$ for all v.

6.2 Experimental Results

To assess the feasibility of the «*Pull-demand*» Heuristic , we compare it to a greedy approach (programming a single request in advance) which has been mentioned by [17] as the best strategy for dynamic settings, following genetic algorithm selection. The two compared approaches use the same request selection strategy (request priority function) during each scenario. The results presented in the Figures 7 and 8 concern a scenario in which vehicles select the cheapest request considering the $costDemand(d, v)$ is the marginal cost mentioned in section 4; the same is applicable to the greedy algorithm, since the vehicle schedule can only contain one demand. We have executed several instances of problems that vary with the size of the fleet n. Each instance of these tests is executed 10 times with different probability seeds. First, we evaluate with a fixed fleet size $n = 16$ to track the quality of the solution over time, then with a variable fleet size $n \in [4..36]$ over 200 cycles for each scenario.

Note that it is impossible to achieve a 100% quality of service rate in these scenarios, because there will always be requests that will be generated until the last cycle, assuming that the scenario execution continues to serve them, while the execution stops at the last cycle, leaving them unserved. Although QoS values of our algorithm are close to those generated by greedy, they remain slightly ahead of them in most cases, as shown in Figure 9.

Starting from small fleet of 4 vehicles the *Pull-demand* heuristic allows to obtain solutions with the same QoS and QoB values as those of the greedy algorithm. By increasing the size of the fleet, more requests can be satisfied, which increases the QoS and QoB value.

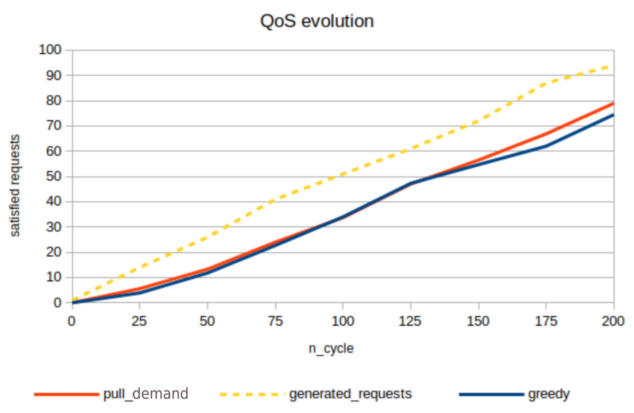

Figure 7. Quality of service for a fleet of 16 vehicles

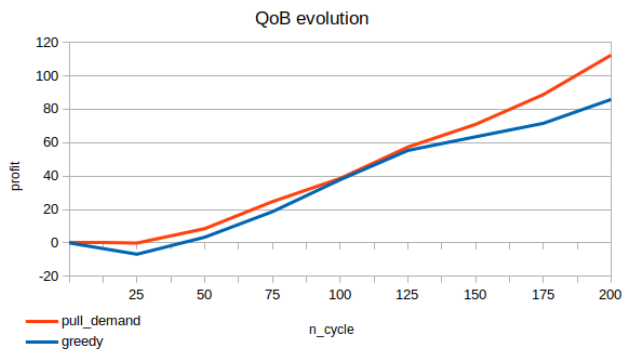

Figure 8. Quality of business for a fleet of 16 vehicles

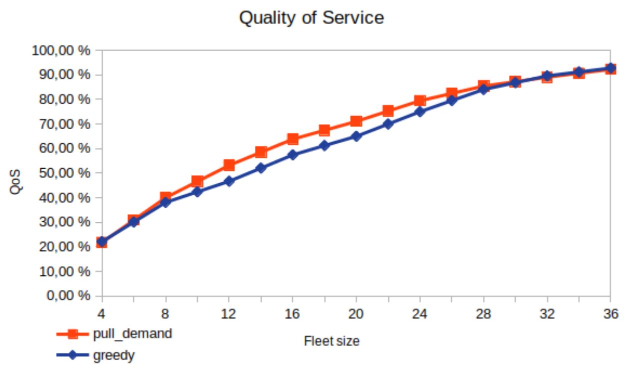

Figure 9. Quality of service with increasing fleet size

We find that the increase of these values for the heuristic is greater than that of the greedy approach. This can be explained by the fact that demand planning improves the system profit in the foreseeable future, as it reduces the process of roaming without a specific destination until a new demand is chosen, which is the behaviour of the vehicles in the greedy algorithm.

The value of QoS continues to grow as the fleet size increases, until it reaches a n_{QoS} threshold where the addition of new vehicles becomes unnecessary, as all requests received in the system now or in the future (except those received in the last moments of execution) can be served with the current fleet size. The same is true for QoB, but

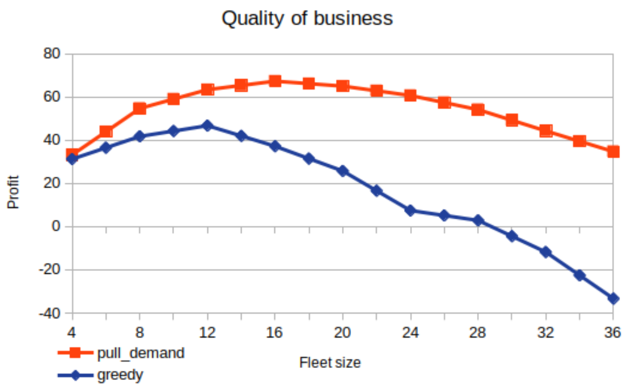

Figure 10. Quality of business with increasing fleet size

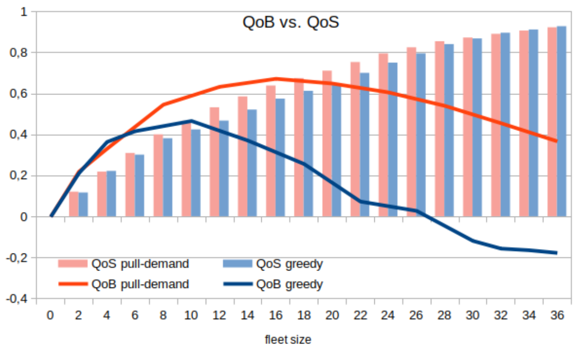

Figure 11. Quality of business vs. quality of solution evolution

the difference here is that the vehicles added will result in additional operational expenses, resulting in a loss of profit value after reaching its n_{QoB} growth threshold. In general, n_{QoB} is less than n_{QoS}, and we can see a trade-off between improving QoS or QoB.

According to the Figure 10, we see that the greedy fleet has $n_{QoB}^{\text{greedy}} = 12$ with a QoS of only about 60%, while $n_{QoB}^{\text{pull-demand}} = 16$ with a QoS of about 70%. Note that $n_{QoB}^{\text{pull-demand}} < n_{QoB}^{\text{pull-demand}}$ and even if it decreases afterwards, the value of $n_{QoB}^{\text{pull-demand}}$ remains higher than $n_{QoB}^{\text{pull-demand}}$. This gives a wider range of options for combining the enhancements of the provided solution (QoS and QoB). The above results show that in all cases, scheduling several requests in advance with the optimization protocol *Pull-demand* gives better results than scheduling a single request, with reduced computation time and no global information sharing.

7 Conclusion

In this paper, we proposed a decentralised protocol for the exchange of requests, based on an insertion heuristic, to allocate requests to vehicles in the context of dynamic transport on demand. We show through examples that the request exchange protocol can be a promising improvement in the quality of the solutions. In order to assess the feasibility of the proposed protocol, we evaluated the results of our technique on synthetic data for taxis operating in the city of Saint-Étienne, and showed that it outperforms a classical greedy approach. In future work, we plan to evaluate the efficiency, performance, robustness and optimality of this heuristic with respect to different approaches, by simulating different parameters on information distribution, decision criteria and different levels of problem dynamics.

REFERENCES

[1] Niels A.H. Agatz, Alan L. Erera, Martin W.P. Savelsbergh, and Xing Wang, 'Dynamic ride-sharing: A simulation study in metro atlanta', *Transportation Research Part B: Methodological*, **45**(9), 1450 – 1464, (2011).

[2] C Bellini, G Dellepiane, and C Quaglierini, 'The demand responsive transport services: Italian approach', *Transaction on the Built Environment, WIT Press , www.witpress.com, ISSN 1743-3509*, **64**, 10, (2003).

[3] Ann Melissa Campbell and Martin Savelsbergh, 'Efficient insertion heuristics for vehicle routing and scheduling problems', *Transportation science*, **38**(3), 369–378, (2004).

[4] Peter Cramton, Yoav Shoham, and Richard Steinberg, 'An overview of combinatorial auctions', *ACM SIGecom Exchanges*, **7**(1), 3–14, (December 2007).

[5] Panayiotis Danassis, Aris Filos-Ratsikas, and Boi Faltings, 'Anytime Heuristic for Weighted Matching Through Altruism-Inspired Behavior', in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pp. 215–222, Macao, China, (August 2019).

[6] Kakan Chandra Dey, Anjan Rayamajhi, Mashrur Chowdhury, Parth Bhavsar, and James Martin, 'Vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication in a heterogeneous wireless network – Performance evaluation', *Transportation Research Part C: Emerging Technologies*, **68**, 168–184, (July 2016).

[7] Malcolm Egan and Michal Jakob, 'Market Mechanism Design for Profitable On-Demand Transport Services', *arXiv:1501.01582 [cs]*, (January 2015). arXiv: 1501.01582.

[8] Mohamad El Falou, Mhamed Itmi, Salah El Falou, and Alain Cardon, 'On demand transport system's approach as a multi-agent planning problem', in *2014 International Conference on Advanced Logistics and Transport (ICALT)*, pp. 53–58. IEEE, (2014).

[9] Andrey Glaschenko, Anton Ivaschenko, George Rzevski, and Petr Skobelev, 'Multi-Agent Real Time Scheduling System for Taxi Companies', *AAMAS*, 8, (2009).

[10] Josep Maria Salanova Grau and Miquel Angel Estrada Romeu, 'Agent Based Modelling for Simulating Taxi Services', *Procedia Computer Science*, **52**, 902–907, (2015).

[11] Josep Maria Salanova Grau, Miquel Angel Estrada Romeu, Evangelos Mitsakis, and Iraklis Stamos, 'Agent based modeling for simulation of taxi services', *Journal of Traffic and Logistics Engineering*, **1**(2), 159–163, (2013).

[12] Keld Helsgaun, 'General k-opt submoves for the Lin–Kernighan TSP heuristic', *Mathematical Programming Computation*, **1**(2), 119–163, (October 2009).

[13] KFH-Group, Urbitran Associates, McCollom Management Consulting, Cambridge Systematics, Transit Cooperative Research Program, United States. Federal Transit Administration, and Transit Development Corporation, *Guidebook for measuring, assessing, and improving performance of demand-response transportation*, volume 124, Transportation Research Board, Washington, DC, 2008.

[14] Gauthier Picard, Flavien Balbo, and Olivier Boissier, 'Approches multiagents pour l'allocation de courses à une flotte de taxis autonomes', in *RIA2018*, number 2 in Revue d'intelligence artificielle, pp. 223–247, (2018).

[15] Wen Shen and Cristina Lopes, 'Managing Autonomous Mobility on Demand Systems for Better Passenger Experience', *arXiv:1507.02563 [cs]*, **9387**, 20–35, (2015). arXiv: 1507.02563.

[16] Marius M. Solomon, 'Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints', *Operations Research*, **35**(2), 254–265, (1987).

[17] Rinde R.S. van Lon, Tom Holvoet, Greet Vanden Berghe, Tom Wenseleers, and Juergen Branke, 'Evolutionary synthesis of multi-agent systems for dynamic dial-a-ride problems', in *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference companion - GECCO Companion '12*, p. 331, Philadelphia, Pennsylvania, USA, (2012). ACM Press.