Dispatching Requests for Agent-Based Online Vehicle Routing Problems with Time Windows

Mahdi Zargayouna, Besma Zeddini

CIT. Journal of Computing and Information Technology, Vol. 28, No. 1, March 2020, 59–72
doi: 10.20532/cit.2020.1004374

Dispatching Requests for Agent-Based Online Vehicle Routing Problems with Time Windows

Mahdi Zargayouna[1] and Besma Zeddini[2]

[1]Gustave Eiffel University, IFSTTAR, COSYS, GRETTIA, Champs sur Marne, France
[2]SATIE, UMR CNRS 8029 ENS Cachan, CY Tech, Cergy-Pontoise, France

Vehicle routing problems are highly complex problems. The proposals to solve them traditionally concern the optimization of conventional criteria, such as the number of mobilized vehicles and the total costs. However, in online vehicle routing problems, the optimization of the response time to the connected travelers is at least as important as the optimization of the classical criteria. Multi-agent systems on the one hand and greedy insertion heuristics on the other are among the most promising approaches to this end. In this paper, we propose a multi-agent system coupled with a regret insertion heuristic. We focus on the real-time dispatching of the travelers' requests to the vehicles and its efficiency. A dispatching protocol determines which agents perform the computation to answer the travelers' requests. We evaluate three dispatching protocols: centralized, decentralized and hybrid. We compare them experimentally based on their response time to online travelers. Two computational types are implemented: a sequential implementation and a distributed implementation. The results show the superiority of the centralized dispatching protocol in the sequential implementation (32.80% improvement in average compared to the distributed dispatching protocol) and the superiority of the hybrid dispatching protocol in the distributed implementation (59.66% improvement in average, compared with the centralized dispatching protocol).

ACM CCS (2012) Classification: Theory of computation → Design and analysis of algorithms → Online algorithms → Online learning algorithms → Scheduling algorithms

Computing methodologies → Modeling and simulation → Simulation types and techniques → Agent / discrete models

Computing methodologies → Artificial intelligence → Distributed artificial intelligence → Multi-agent systems

Keywords: vehicle routing problems, Multi-agent systems, Insertion heuristics

1. Introduction

Many of the most successful mass-market modern transportation applications, such as dynamic ridesharing or online food delivery, are instantiations of a theoretical problem called the vehicle routing problem (VRP). In a VRP, a number of nodes have to be visited only once by a number of capacitated vehicles. The objectives of the problem are, generally, first to minimize the number of mobilized vehicles, then to minimize the total incurred costs. Solving these problems has high practical usefulness and they are challenging optimization problems with stimulating issues. The problem with time constraints is one of the most widely studied variants of VRP (vehicle routing problem with time windows, VRPTW henceforth [1]). In this variant, the nodes must be visited inside time windows.

Vehicle routing problems can be divided in two categories: static problems and dynamic problems. In the static problems, all the problem data are available before the start of the optimization process. In the dynamic problems, the problem data are incomplete before the start of execution, and they are gradually discovered while the optimization is progressing. The incomplete data may concern any element of the problem, such as the traffic data or the available vehicles. However, the dynamic aspect usually refers to the travelers to be transported, which are unknown before execution (like in ridesharing and dial-a-ride systems). Operational problems are never completely static and it is

reasonable to assume that a static system does not meet current operational configurations. Indeed, in real vehicle routing problems, and even when all travelers are known in advance (with a reservation system for example), there is always an element that makes the problem dynamic. These elements may include no-shows, delays, breakdowns, *etc.*

Online vehicle routing problems could be considered as an extreme case of dynamic vehicle routing problems. Indeed, not only the problem data are not completely known before the start of the optimization, but the travelers connect in real time to the system and expect almost immediate responses to their requests. The system response time in this type of problems is therefore vital. If the optimization system needs two additional minutes to improve its current solution, it should immediately provide the current solution to the traveler, since the latter will likely not wait that long to get an answer to his request.

To meet the requirement for short response times, we rely on the multi-agent paradigm to solve online vehicle routing problems. An agent is an intelligent entity, located in an environment and that applies autonomous actions to achieve its objectives [2, 3]. Multi-agent modeling of the online VRPTW is relevant for the following reasons. On the one hand, the choice of a model allowing the distribution of calculations should make it possible to shorten the response times to travelers' requests. On the other hand, nowadays, vehicles are more and more connected and have on-board computing capacities. In this context, the transportation system is *de facto* distributed and requires appropriate modeling to take advantage of these facilities. The multi-agent system (MAS) that we propose in this article is composed of vehicle agents, traveler agents, interface agents and planner agents. The MAS simulates a distributed version of the so-called "insertion heuristics". Insertion heuristics are methods that involve inserting travelers individually in the vehicle routes. Each traveler is inserted in the route of the vehicle with the minimum marginal cost (the cost could refer to the incurred detour, for example). This is the fastest known heuristic, since there is no reconsideration of previous insertion decisions. When coupling insertion heuristics and multi-agent systems, there is a choice to be made regarding the location of the calculations of the best routes. In this context, we propose three classic dispatching protocols and compare them according to their ability to provide better response times to travelers. The dispatching protocol determines which agents perform the calculation to insert the travelers in the vehicle routes. In the centralized protocol, the planner performs most of the calculation. In the decentralized protocol, the vehicle agents perform most of the calculation in a collaborative way. Finally, in the hybrid protocol, work is shared between traveler agents and vehicle agents.

The remainder of this paper is structured as follows. In Section 2, we discuss previous proposals for the dynamic VRPTW. The multi-agent system and the three dispatching protocols of the MAS are presented in Section 3. The regret insertion heuristic is described in Section 4. We provide our experimental results in Section 5 and then conclude with a few remarks in Section 6.

2. Related Work

Operational settings of VRPTW are hard to be met using purely exact approaches, and it is hard to find optimal solutions to all of 56 benchmarking problems proposed in [26] for instance. A formal definition of the VRPTW problem can be found in [25] and interested readers of optimization approaches can refer to, *e.g.* [4] for a survey.

In fact, most of the proposed solution methods are heuristic or metaheuristic methods which provide good results in non-exponential times. These approaches have presented good results with benchmark problems. For instance, large-neighborhood local search [5], simulated annealing [6], evolutive strategies [7] and ant colonies [8, 9] present excellent performances with static problems.

When dealing with dynamic VRPTW, most of the approaches are more or less direct adaptations of static methods. For example, local research in large neighbourhoods is adapted to a dynamic context in [10] and genetic algorithms are adapted to dynamic settings in [11]. Insertion heuristics are the most widely adapted ap-

proach for the dynamic VRPTW (*e.g.* [12, 13, 14, 15]). Insertion heuristics are, in their original version, greedy algorithms, in the sense that the decision to insert a given traveler into a vehicle's itinerary is not reconsidered. Insertion heuristics are usually associated with metaheuristics to improve the quality of solutions. The advantage of using insertion heuristics is that they are both intuitive and fast. However, their resolution process is said to be short-sighted. Indeed, by definition, the system does not know which travelers will appear once it has assigned known travelers to the vehicles. Therefore, even if it happens to have a current optimal assignment of known travelers, the appearance of a new traveler could make the old assignment sub-optimal.

A vast majority of agent-based approaches in the literature are based, at least partially, on insertion heuristics. In [16] for instance, the authors propose a multi-agent architecture to solve a VRP and a multi-depot VRP. In [17], the authors propose a multi-agent architecture to solve a dial-a-ride problem. The principle of these two proposals is the same: distribute an insertion heuristic, followed by a post-optimization step. In [16], travelers are processed sequentially. They are distributed to all vehicles, which in turn offer insertion offers and the best offer is chosen by the traveler. In the second step, the vehicles exchange travelers to improve their solutions, each vehicle knowing all the other agents of the system. Since the vehicles operate in parallel, the authors plan to apply different heuristics for the vehicles. In-Time [17] is a system composed of traveler agents and vehicle agents. The traveler agent advertises himself and all vehicle agents calculate his insertion price in their itineraries. The traveler agent chooses the cheapest offer. The authors propose a distributed local search method to improve the solutions. They allow a traveler to stochastically request to cancel his current assignment and re-enter the system, hoping to get a better offer from another vehicle. MARS [18] models cooperative planning in a shipping company as a multi-agent system. The solution to the global scheduling problem emerges from local decisions. The system benefits from an a priori structuring of agents, since each vehicle is associated with a particular company and can only serve the travelers of that company.

For the reasons that we have given in the introduction, we choose a multi-agent modeling to solve the dynamic VRPTW. We opted for a solution based on insertion heuristics for their fast execution times and their adaptation to dynamic settings. As far as we are aware, none of the previous proposals have focused on the response time of the system to online travelers. In this paper, we propose three dispatching protocols and compare them based on their response time.

3. Dispatching Protocols

The three protocols for dispatching travelers' requests that we propose in this article are defined within the framework of a multi-agent system. Three main types of agents are defined in the system: traveler agents, vehicle agents, and interface agents. When a human user logs into the system, an interface agent creates a traveler agent representing him. Also, exclusively for the centralized protocol, we defined an additional agent, called the planner agent, which is responsible for executing all routings.

In online problems, system response time is essential, and only very fast approaches can compete. The fastest and most popular approach is the greedy insertion approach, initially proposed by Marius M. Solomon [26] in this context. The principle is to gradually insert travelers, one by one, into the vehicle's itineraries. To do this, the price of inserting a traveler into vehicle's itineraries is calculated, and the vehicle with the minimum insertion cost is chosen to insert the traveler. To calculate this insertion price for a vehicle, we compute the marginal cost between his current route and his new route.

When solving this problem with a multi-agent system, there are several alternatives regarding the entities processing the request. Each alternative is called a "dispatching protocol". In this section, we describe three possible dispatching protocols that we have designed, implemented and compared to model the online VRPTW: centralized dispatching, decentralized dispatching and hybrid dispatching. In the centralized protocol, the planner agent performs most of the computation. In the decentralized protocol, the vehicle agents perform most of the computation in a collaborative way. Finally, in the hy-

brid protocol the work is split between travelers and vehicles. Our objective is to verify which dispatching protocol is the most effective, in terms of response time to travelers' requests. The comparison of the different protocols does not take into account the traditional optimization criteria (number of vehicles mobilized and total costs incurred). Indeed, in terms of optimization, the three dispatching protocols follow the same algorithm and use the same insertion price (described in the next section). The only difference concerns the response time, *i.e.* the time that takes the system to decide which vehicle will visit the traveler. The three dispatching protocols are described in the following subsections.

3.1. Centralized Dispatching Protocol

In the centralized protocol, all processing is performed by a central entity which creates the vehicle routes. One of the main advantages of this protocol is that it allows online optimization techniques to be used in a centralized fashion. Online optimization (*e.g.* in [19]) provides benefits of the exact optimization techniques while also reducing response times. The principle is to discretize the processing time into time intervals. During each interval, an optimization is carried out with known travelers. New travelers are queued up, waiting for the next interval. Known travelers who could not be served and new travelers are submitted for the new optimization round.

Our objective is to maintain the same resolution approach while comparing response times; the centralized approach therefore mimics the same insertion heuristic as the other two dispatching protocols. In our proposal (see Figure 1), all travelers' requests are processed by the same planner agent. The planner agent has all the necessary information on each vehicle and traveler, as well as their current status. With this information, it places the current traveler in the position incurring the minimum marginal cost.

The scenario is as follows. A user appears and an interface agent creates a traveler agent to represent him. The traveler immediately sends a request to the planner agent, who tries to insert it into the route of each vehicle in the system, in all possible positions. To do so, it sequentially performs, for each vehicle, a procedure for calculating the insertion price of the vehicle, and chooses the vehicle and the insertion position with the minimum price. If no vehicle can insert the traveler, a new vehicle agent is created, with an empty route, and the traveler is inserted in the only possible position. Finally, the planner informs the traveler and the vehicle of the outcome of the procedure. Vehicle agents in the centralized dispatching protocol do not perform any calculations and only acknowledge the updates in their routes, decided by the planner agent.

Centralized dispatching has two main disadvantages. On the one hand, it is not possible to distribute the execution over several calculation units in order to limit the system's re-

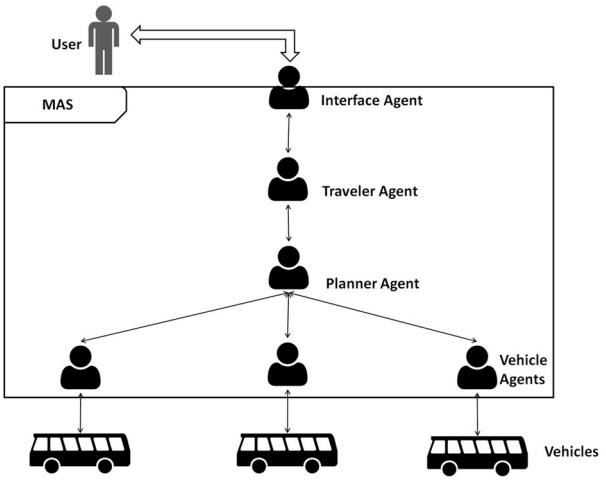

Figure 1. Centralized dispatching protocol.

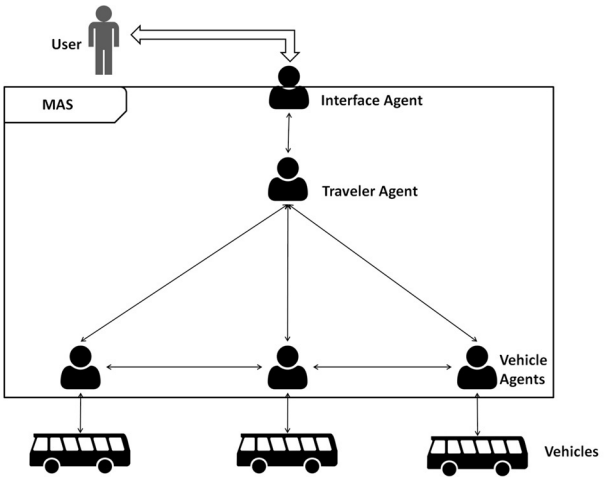

Figure 2. Decentralized dispatching protocol.

sponse time, which is the main concern in the online VRPTW. On the other hand, a failure of the planner would result in a complete system failure. Nevertheless, the centralized dispatching protocol has the advantage of minimizing communications between the agents, which are limited to the notification of the travelers and vehicles of the computation outcome. The number of messages is $N(1 + V)$, with N being the number of travelers and V being the number of vehicles.

3.2. Decentralized Dispatching

Decentralized dispatching is illustrated in Figure 2. Following this protocol, there is no bottleneck for route calculation. Following the principle of the greedy insertion heuristic, each vehicle agent tries to insert the new traveler in his route, and proposes an insertion price, corresponding to the "most economical" position where he can insert the traveler. The chosen vehicle will be the one with the minimum insertion price to transport the traveler.

In this dispatching protocol, the choice of the vehicle with the minimum price, the calculation of the price and the choice of the vehicle that will serve the traveler, are all carried out in a distributed way. Indeed, the scenario is as follows. When a new traveler appears, he sends his request to all the vehicles in the system. Upon receipt of the request, each vehicle calculates an insertion price for him. When it has completed his calculation, each vehicle issues a message to all vehicles with his ID and price.

For the processing of these messages and determination of the winning agent vehicle, we propose the following process. Each vehicle agent broadcasts his own calculated price to other vehicle agents. When he receives a new message containing a price calculated by another agent, he sorts the received offers, including his own offer, following their associated prices. When all the other vehicle agents have offered their prices, the vehicle agent checks to see which one is the winner. Then he updates his itinerary with the new inserted traveler and informs the concerned traveler agent accordingly.

This scheduling has the advantage of fully distributing the processing and being fault-tolerant. With a decentralized dispatching protocol, the entire system is not blocked following an agent's failure, as it would be the case with the centralized dispatching. In this protocol, for each new traveler, vehicles must cooperate to choose the one that is most appropriate to serve him. However, the number of exchanged messages could increase considerably, which is usually the price to pay for a processing distribution. The number of messages exchanged between vehicles with this dispatching protocol is equal to $N \times V^2$. The overall number of messages between all the agents is equal to $N(1 + V(1 + V))$.

3.3. Hybrid Dispatching

Hybrid dispatching is a compromise between the centralized and decentralized approach. In the hybrid approaches (see Figure 3), the traveler agent acts as a dispatcher. The traveler agent disseminates the request, collects the insertion prices of the vehicle agents and chooses the one who proposes the minimum price.

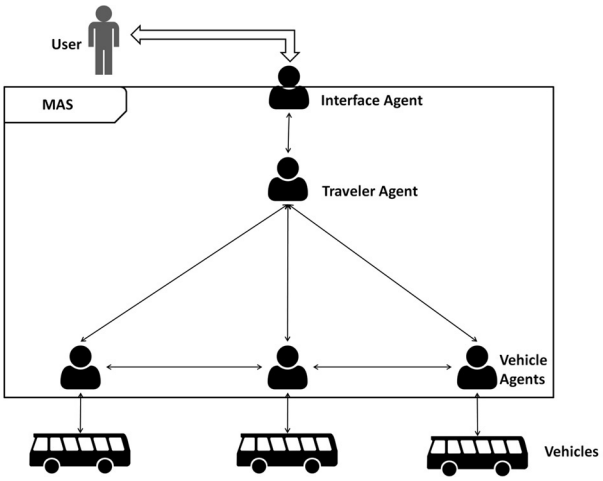

Figure 3. Hybrid dispatching protocol.

The hybrid approach applies the following protocol. A new human user provides the interface agent with the information about his transport request. The interface agent creates a traveler agent who represents him. Then, the new traveler agent sends a message to all the vehicles in the system. Each vehicle agent checks if he can insert the traveler in his route. The vehicle agent then sends his insertion price to the traveler agent. The traveler agent collects the answers from the vehicles and chooses the vehicle that

offers the minimum price. Once he has chosen the best vehicle that can serve him (if there is at least one that can insert the traveler), he issues a new message to the vehicles to inform them of his decision and asks the winning vehicle agent to insert him in his route and subsequently serve him. When the vehicle agent receives the traveler's message informing him that he is the winner, he updates his route and inserts the traveler.

Thus, the objective of the hybrid approach is to relax the planner of all calculations, and to limit the communication between vehicles. The total number of messages in the hybrid dispatching protocol is equal to $3VN$.

In the three dispatching protocols presented in this section, the planner agent, or vehicle agent, calculates a price for the insertion of a given traveler. The price calculation is the same for all three protocols. It is an original measure and can be qualified as a kind of regret heuristic that tries to overcome the disadvantages of traditional insertion price measures. The heuristic is described in the following section.

4. Space-Time Regret Insertion Heuristic

In the heuristics and multi-agent methods referred to in the literature, the hierarchical objective of minimizing the number of mobilized vehicles is considered as priority w.r.t the general costs (including the distance traveled by all the vehicles). A majority of the literature heuristics are, as a consequence, based on a two-phase approach: minimization of the number of vehicles followed by the minimization of the traveled distance [20]. The model that we propose in this section has the objective of minimizing the number of used vehicles in priority, while keeping the use of a "pure" insertion heuristics, *i.e.* without any further improvements to meet response time requirements. To this end, our heuristic encourages the vehicle agents to cover a maximal space-time area of the transportation network, avoiding the mobilization of a new vehicle if a new traveler appears in an uncovered zone.

A space-time pair $<i, t>$ – with i being a node and t being a time – is said to be "covered" by a vehicle agent v if v can be in i at t. The set of space-time nodes that are covered by the vehicle agent is called "action zone of the vehicle". In the context of the online VRPTW, the maximization of the vehicle agents' action zones gives them the maximum chance to satisfy the demand of a future (unknown) traveler. Through the modeling of vehicle agents' space-time action zones, we propose a new method to compute the traveler's insertion price in the route of a vehicle. This proposal is a kind of regret insertion heuristic. Regret insertion heuristics, instead of choosing the vehicle that has the minimal marginal cost, choose the vehicle and the traveler with the largest "regret". The regret is a measure of the potential price to be paid if a given traveler were not immediately inserted in the route of a given vehicle. There are several methods to compute the regret, such as the the sum of the differences between all the available prices and the minimum price (cf. [21] for instance).

4.1. Environment Modeling

Provided the network spatial graph G, we build the MAS environment in the form of a space-time network, inferred from the spatial graph. For each node of the graph we create pairs $<space, time>$, each representing the "state" of a node at a discrete time period. The space-time network is made of several spatial subgraphs. Each subgraph is a copy of G and corresponds to the state of the G at a certain period of time (cf. Figure 4). We index the nodes of the subgraphs as follows: $<0, t>, ..., <N, t>$, with $t \in \{1, ..., h\}$, where $0, ..., N$ are the nodes of the network and h is the number of considered discrete time periods. The total number of nodes in the space-time network is then equal to $h \times N$. The edges linking the nodes of a subgraph are those of the spatial graph, and the costs are the travel times.

4.2. Intuition of the Space-Time Action Zones

Consider a vehicle agent v that has an empty route. Consider also a new traveler c described by: n a node, $[e, l]$ a time window, s a service time, and q a quantity. In order for v to be able to insert c in his schedule, l has to be big enough to allow v to be in n without violating his time constraints (if e is too small, v will simply have

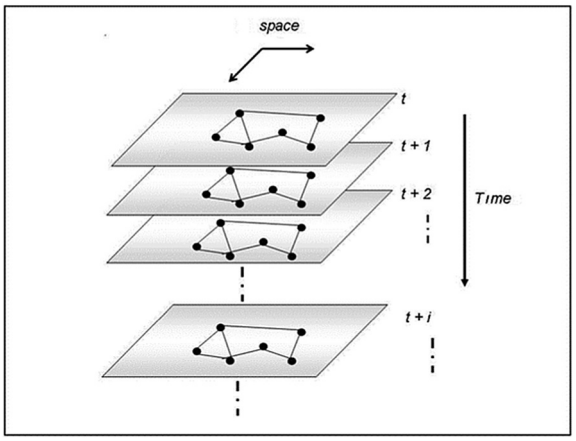

Figure 4. Space-time network.

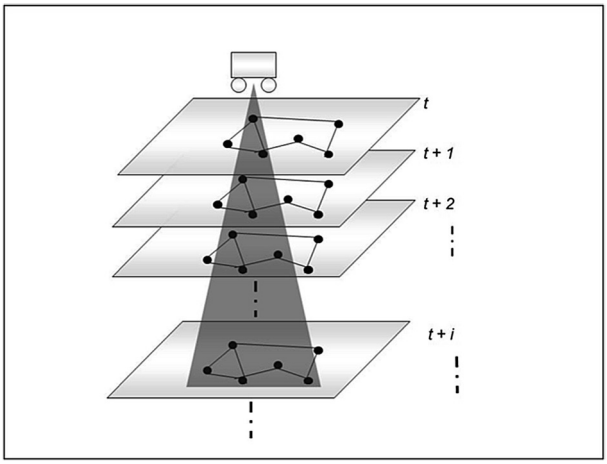

Figure 5. Initial space-time action zone.

to wait until *e*). More precisely, the current time *t*, plus the travel time between the depot and *n* has to be less than or equal to *l*. Starting from this observation, we define the action zone of a vehicle agent as the set of pairs <*n*, *t*> of the space-time network that remain valid given his current route (*n* can be visited by the vehicle at *t*). The action zone of a vehicle agent with an empty route is illustrated by the conic shadow in Figure 5.

When a vehicle agent inserts a traveler in his route, his action zone is recomputed, since some <*node*, *time*> pairs become not feasible. In Figure 6, a new traveler is inserted in the route of the vehicle. The action zone of the vehicle agent after inserting the traveler is represented by the interior of the contour of the bold lines which represent the space-time nodes that remain feasible after the insertion of the traveler.

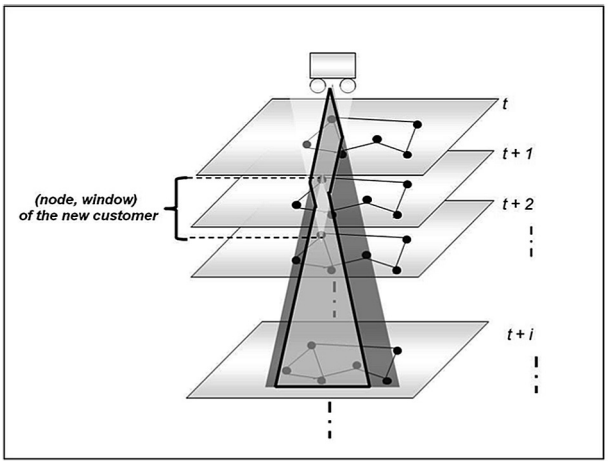

Figure 6. Action zone after the insertion of a traveler.

The insertion price sent from a vehicle agent *v* to a traveler agent *c* corresponds to the hypothetical decrease of the action zone of *v* following the insertion of *c* in his route, *i.e.* the number of space-time nodes that would not be feasible anymore.

The idea is that the chosen vehicle for the insertion of a traveler is the one that maintains the maximum chance to be candidate for the insertion of future travelers. Thus, the criterion that is maximized by the society of vehicle agents is the sum of their action zones, *i.e.* the capacity of the MAS to react to the appearance of traveler agents, without mobilizing new vehicles.

To illustrate the action zones and their dynamics, we present the version of the measure that is related to a Euclidean problem, *i.e.* where travel times are computed following the Euclidean metric. The following paragraphs detail the measure as well as its dynamics.

4.3. The Computation of Action Zones

In the Euclidean case, the transportation network is a plane, and the travel times between two points *i* (described by (x_i, y_i)) and *j* (described by (x_j, y_j)) is equal to $\sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2}$. Therefore, if a vehicle is in *i* at the moment t_i, it cannot be in *j* earlier than $t_i + \sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2}$.

We can compute at any time, from the current position of a vehicle, the set of triples (x, y, t) where it can be in the future. Indeed, considering a plane with an x-axis in $[x_{min}, x_{max}]$ and a y-axis in $[y_{min}, y_{max}]$, the set of space-time positions is the set of points in the cube delimited by $[x_{min}, x_{max}]$, $[y_{min}, y_{max}]$ and $[e_0, l_0]$ (e_0 and l_0 are the scheduling horizon and are the minimal and maximal values for the time windows). Consider a vehicle in the depot (x_0, y_0) at t_0. The set of points (x, y, t) that are accessible by this vehicle are described by the following inequality:

$$\sqrt{(x-x_0)^2+(y-y_0)^2} \le (t-t_0).$$

The (x, y, t) satisfying this inequality are those that are positioned inside the cone C of vertex (x_0, y_0, t_0) and with the equation $\sqrt{(x-x_0)^2+(y-y_0)^2} \le (t-t_0)$ (cf. Figure 7).

This cone represents the action zone of a vehicle agent, with an empty route, in the Euclidean case. It represents all possible space-time positions that this vehicle agent is able to have in the future.

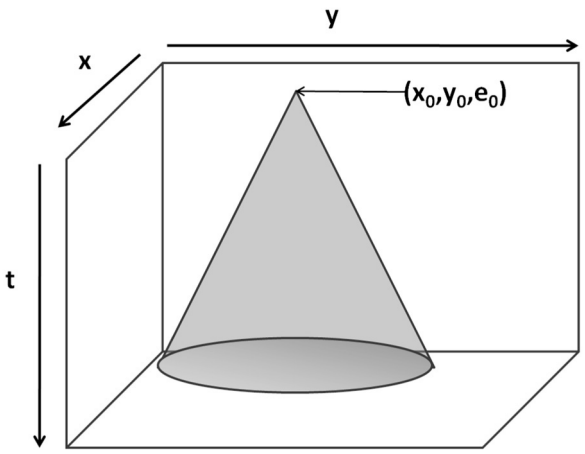

Figure 7. Initial action zone.

We use the action zone of the vehicle agents when a traveler agent has to choose between several vehicle agents for his insertion. We have to be able to compare the action zones of different vehicle agents. To do so, we propose to quantify it, by computing the volume of the cone C representing the future possible positions of the vehicle:

$$Volume(C) = \frac{1}{3} \times \pi \times (l_0 - e_0)^3.$$

This is the quantification of the initial action zone of any new vehicle agent joining the MAS ($volume_0$). When a new traveler agent appears, each vehicle agent computes his new action zone and its volume ($volume_1$). The price that it proposes to the traveler agent is the difference between his old action zone and his new one ($volume_0 - volume_1$). The new action zone computation is detailed in the following paragraph.

4.4. Dynamics of the Action Zones

Consider a traveler c_1 (of coordinates (x_1, y_1) and with a time window $[e_1, l_1]$) that joins the system, and suppose that v is currently the only available vehicle agent which has an empty route. The agent v has to infer his new space-time action zone, *i.e.* the space-time nodes that it can still reach without violating the time constraints of c_1. The new action zone answers the following questions: "if v had to be in (x_1, y_1) at l_1, where would it have been before? And if it had to be there at e_1, where would it be after $e_1 + s_1$?" (the service time is the time needed to load or unload the travelers). The triples (x, y, t) where the vehicle agent can be before visiting c_1 are described by the inequality (3), and the triples (x, y, t) where it can be after visiting c_1 are described by the inequality (4).

$$\sqrt{(x-x_1)^2-(y-y_1)^2} \le (l_1 - t) \qquad (3)$$

$$\sqrt{(x-x_1)^2-(y-y_1)^2} \le t - (e_1 + s_1) \qquad (4)$$

The new action zone is illustrated by Figure 8: the new measure consists of the volume of the intersection of the initial cone C with the union of the two new cones described by the inequalities (3) and (4) (denoted respectively by C_1 and C_2). The new measure of the action zone is equal to the volume of the intersection of C with the union of C_1 and C_2.

The insertion price of a traveler in the route of a vehicle is equal to the measure associated with the old action zone of the vehicle minus the measure of the new action zone, after the insertion of the traveler. The measured quantity represents the space-time positions that would not be feasible anymore for the vehicle, if it had to insert this traveler in his route. The winner

vehicle agent is the one with the minimum loss in his space-time action zone. Therefore, the new regret heuristic encourages to choose the vehicle with the highest marginal probability to be candidate for future travelers.

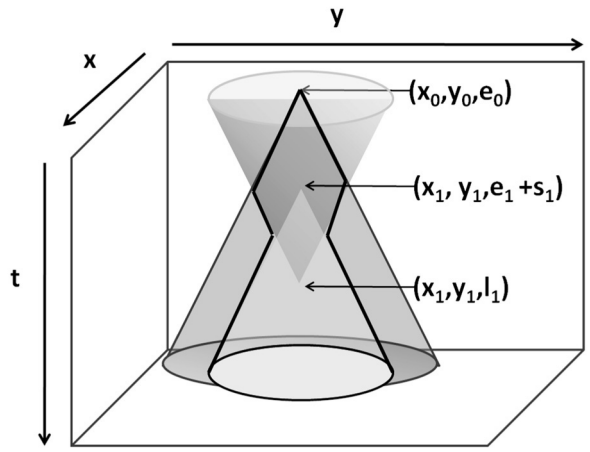

Figure 8. Space-time action zone after the insertion of c_1.

4.5. Coordination of Action Zones

The objective of the space-time organization model is to allow better space-time coverage of the transportation network. This improvement is materialized by a minimal mobilization of vehicles when confronted with the appearance of new travelers. With the mechanism described above, every vehicle agent tries to maximize his own action zone independently from the other agents. However, it would be more interesting that the agents cover the network in coordination. More precisely, for a vehicle, to lose space-time nodes that he is the only one to cover should be more costly than to lose nodes that are covered by other vehicle agents.

To this end, we associate with every node of the space-time network the list of vehicles covering it. Every vehicle notifies the space-time nodes that they are part of his action zone and every node continuously updates its list. Similarly, when the action zone of a vehicle agent loses a node, the node is notified and its list of vehicles is updated.

Now, when the insertion price of a traveler is computed, every vehicle agent starts by determining the space-time nodes that it would lose if it had to insert the new traveler. Then, it interrogates each of these nodes about the "price to pay" if it were not covering it anymore. This price is inversely proportional to the number of vehicles covering this node. More precisely, the price to pay is equal to

$$\frac{1}{\left|v_{<n,\,t>}\right|}$$

with $v_{<n,\,t>}$ denoting the vehicle agents covering the space-time node $<n, t>$ and $|v_{<n,\,t>}|$ the number of such vehicles.

This method associates a higher penalty with a decision to stop covering a node that is less covered by the others. Therefore, the vehicle agents are indirectly incited to cover the whole network in a coordinated way.

5. Experiments

In this section, we provide the experimental results of our simulations. First, we provide optimization costs for the three dispatching protocols, which are the same since they use the same regret heuristic. Then we compare the three dispatching protocols in terms of the response time.

Marius M. Solomon [26] has created a set of different static problems for the VRPTW. These challenging and diverse problems can be used as the benchmark examples for comparing different proposed vehicle routing methods. In Solomon's benchmarks, six different sets of problems have been defined: C1, C2, R1, R2, RC1 and RC2. From geographical point of view, the travelers are uniformly distributed in the problems of type R, clustered in the problems of type C, and a mix of travelers, uniformly distributed and clustered, is used in the problems of type RC. The problems of type 1 have narrow time windows (very few travelers can coexist in the same vehicle's route) and the problems of type 2 have wide time windows. Finally, a constant service time is associated with each traveler, which is equal to 10 in the problems of type R and RC, and to 90 in the problems of type C. Short service times would represent a problem where the loading and un-

loading of the transported entities is fast (transport of persons for instance). In every problem set, there are between 8 and 12 files, each containing 100 travelers.

We choose to use Solomon's benchmarks, while following the modification proposed by [22] to make the problem dynamic. To this end, let $[0, T]$ be the simulation time. All the time related data (time windows, service times and travel times) are multiplied by $\frac{T}{|l_0 - e_0|}$, with $[e_0, l_0]$ being the scheduling horizon of the problem. The authors divide the travelers set in two subsets, the first subset defines the travelers that are known in advance, and the second one the travelers who appear during execution. We do not make this distinction, since we consider no travelers known in advance. For each traveler, an occurrence time is associated, defining the moment when the traveler is recognised by the system. Given a traveler i, the occurrence time that is associated is generated randomly between $\left[0, \overline{e}_i\right]$, with:

$$\overline{e}_i = e_i \times \frac{T}{l_0 - e_0}. \quad (6)$$

It is known that the behavior of insertion heuristics is strongly sensitive to the appearance order of the travelers to the system. For this reason, we do not consider only one appearance order. We launch the process that we have just described ten times with every problem file, this way creating ten different versions of every problem file.

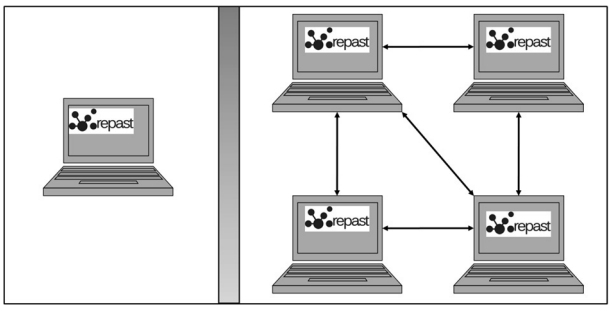

Figure 9. Sequential implementation (left) and distributed implementation (right).

We have implemented all the systems using the multi-agent Java-based platform Repast Simphony [23]. We have executed our experiments on a PC with an Intel Xeon E7-4820 processor, and 50 GB of RAM for the sequential versions, and a four PC network for the distributed versions, each with the same configuration (Intel Xeon E7-4820 processor, and 50 GB of RAM). Cf. Figure 9.

5.1. Optimization Cost

We have implemented two MAS with almost the same behavior, following the hybrid dispatching protocol. The only difference concerns the measure used by vehicle agents to compute the insertion price of a traveler. For the first implemented MAS, it relies on the Solomon measure (noted Δ Distance). The second relies on the space-time model (noted Δ Space-Time). We choose to run our experiments with the problems of class R and C, of type 1, which are the instances that are very constrained in time (narrow time windows). Recall that the objective of the problem is to minimize costs, materialized by a minimal number of mobilized vehicles, and then a minimal total traveled distance.

Table 1. Results summary (Criterion: Fleet Size).

Problem/Method	Δ Distance	Δ Space-Time
R1 25 travelers	64	53
C1 25 travelers	34	31
R1 50 travelers	107	92
C1 50 travelers	60	53
R1 100 travelers	181	150
C1 100 travelers	121	108

For each problem class and type, we have considered different numbers of travelers in order to verify the behavior of our models w.r.t. the problem size. To this end, we have considered successively the 25 first travelers, the 50 first travelers, and finally all the 100 travelers contained in each problem file. Table 1 summarizes the results. Each cell contains the best results obtained with each problem class (the sum of all problem files). The results show, with the two classes of problems, that the use of the space-time model mobilizes fewer vehicles than the classic model ($53 < 64$, $31 < 34$,

92 < 107, 53 < 60, 150 < 181, 108 < 121). The average improvement of the space-time model compared to the classic model is 13.26%. These results validate the intuition of the model, which implies maximizing the future insertion possibilities for a vehicle agent.

Once this result has been validated, it is interesting to check the results with respect to the total distance traveled by all the vehicles. Table 2 summarizes the results[1]. With respect to this criterion, the space-time model is more efficient for two problem classes (C1 25 travelers and R1 100 travelers), with 1.80% improvement on average. It is less efficient for four problem classes (R1 25 travelers, R1 50 travelers, C1 50 travelers and C1 100 travelers), with 1.39% improvement for the classic model. The fact remains that our results provide better results than the traditional heuristic, provided that the primary objective of the problem is to minimize the number of vehicles mobilized by the system.

Table 2. Results summary
(Criterion: Total Traveled Distance).

Problem/Method	Δ Distance	Δ Space-Time
R1 25 travelers	6372	6561
C1 25 travelers	3167	3152
R1 50 travelers	12036	12089
C1 50 travelers	6712	7093
R1 100 travelers	17907	17348
C1 100 travelers	16011	16512

The two following subsections provide the results in terms of response time for the three dispatching protocols.

5.2. Sequential Implementation Experiments

Table 3 provides the values in terms of average response times (in milliseconds) of every dispatching protocols in the centralized implementation. The response time for a traveler is the difference between the moment when the traveler agent is created and the moment when a vehicle is chosen by the traveler. The centralized protocol provides the best results (32.80% improvement, in average, compared to the distributed protocol), followed by the hybrid protocol (24.13% improvement, in average) and the decentralized protocol. This is due to the fact that the centralized approach does not generate communication between agents and does not assume any concurrency management. The hybrid approach provides results that are close to the centralized dispatching protocol. However, it provides results of worse quality for two reasons. On the one side, it generates more messages (linear with the number of vehicles) between the traveler agent and the vehicle agents. On the other side, the management of concurrent processes of the vehicles and travelers, and the fact that their contexts have to be restored every time the scheduler executes them, increases the exhibited response times for the travelers. Finally, the decentralized approach suffers from the two drawbacks: it generates a quadratic number of messages and it uses pseudo-parallelism which slows down the processing.

Table 3. Sequential implementation
(average response time (ms)).

Problem/Protocol	$Centralized_{seq}$	$Decentralized_{seq}$	$Hybrid_{seq}$
R1 25 travelers	35	54	36
C1 25 travelers	32	48	40
R1 50 travelers	36	59	44
C1 50 travelers	38	56	41
R1 100 travelers	43	63	49
C1 100 travelers	44	59	47

However, this round of experiments was executed on a single computer, therefore these results are not fair towards the decentralized dispatching strategy, and to a lesser extent, towards the hybrid approach. Indeed, to use the full capacity of these protocols, we have to execute our simulations on a mini-cloud.

[1]In Solomon's benchmarks, there is no unit associated with the distances.

5.3. Distributed Implementation Experiments

It is possible with Repast Simphony to distribute a simulation over a network, using relevant Java APIs. We report the corresponding results in Table 4. These results are interesting since they provide a new enlightenment concerning the most promising dispatching protocol in terms of response time to online users. Indeed, in the absence of slow-down due to single PC pseudo-parallelism, the hybrid protocol takes profit of the processing distribution, without suffering from a too big number of exchanged messages (59.66% improvement in average, compared with the centralized protocol). The decentralized protocol comes to the second position in terms of performance (39.31% improvement in average), taking profit from the distribution but suffering from its too big bandwidth consumption. The centralized protocol comes to the last position, since its gain in terms of exchanged messages does not counterbalance its sequentialization of processing. Observe that the centralized protocol yielded almost identical results in the distributed implementation, as well as in the sequential implementation. The small difference comes from the fact that vehicle agents are executed in other hosts than the planner agent, which results in a small additional cost in terms of communication.

Table 4. Distributed implementation (average response time (ms)).

Problem/Protocol	$Centralized_{dist}$	$Decentralized_{dist}$	$Hybrid_{dist}$
R1 25 travelers	36	23	14
C1 25 travelers	34	24	17
R1 50 travelers	37	24	16
C1 50 travelers	40	25	17
R1 100 travelers	44	23	15
C1 100 travelers	48	24	16

6. Conclusion

In this paper, we have proposed a multi-agent system implementing a regret insertion heuristic for the online vehicle routing problem with time windows. We propose three versions of the system, focusing on the travelers' dispatching protocols. The dispatching protocol decides which agents perform the computation to answer the travelers' requests. In the centralized protocol the planner agent performs most of the computation. In the decentralized protocol, the vehicle agents perform most of the computation in a collaborative way. Finally, in the hybrid protocol, the work is split between travelers and vehicles. We have compared these three approaches based on their response time to online users. We have considered two implementation types, sequential implementation and distributed implementation. The results have shown superiority of the centralized protocol in the first implementation (32.80% improvement, in average, compared to the distributed dispatching protocol) and superiority of the hybrid protocol in the second implementation (59.66% improvement, in average, compared with the centralized dispatching protocol). In our future works, we will consider more dynamic problems in which not only travelers, but also the traffic conditions are unknown before execution. To this end, we will integrate our vehicle routing system inside the multimodal traffic simulator SM4T [24].

References

[1] A. Expósito *et al.*, "Quality of Service Objectives for Vehicle Routing Problem with Time Windows", *Applied Soft Computing*, vol. 84, 2019. https://doi.org/10.1016/j.asoc.2019.105707

[2] T. Yang *et al.*, "A Survey of Distributed Optimization", *Annual Reviews in Control*, vol. 47, pp. 278–305, 2019. https://doi.org/10.1016/j.arcontrol.2019.05.006

[3] N. Bessghaier *et al.*, "Management of Urban Parking: an Agent-Based Approach", in *Proc. of the Int. Conference on Artificial Intelligence: Methodology, Systems, and Applications*, Springer, 2012, pp. 276–285. https://doi.org/10.1007/978-3-642-33185-5_31

[4] A. Mourad *et al.*, "A Survey of Models and Algorithms for Optimizing Shared Mobility", *Trans-*

portation Research Part B: Methodological, vol. 123, pp. 323–346, 2019. https://doi.org/10.1016/j.trb.2019.02.003

[5] J. H. Drake *et al.*, "Recent Advances in Selection Hyper-Heuristics", *European Journal of Operational Research*, 2019. https://doi.org/10.1016/j.ejor.2019.07.073

[6] J. Van Engeland *et al.*, "Literature Review: Strategic Network Optimization Models in Waste Reverse Supply Chains", *Omega*, vol. 91, 2020. https://doi.org/10.1016/j.omega.2018.12.001

[7] S. C. Ho *et al.*, "A Survey of Dial-a-Ride Problems: Literature Review and Recent Developments", *Transportation Research Part B: Methodological*, vol. 111, pp. 395–421, 2018. https://doi.org/10.1016/j.trb.2018.02.001

[8] H. Hu *et al.*, "Emergency Material Scheduling Optimization Model and Algorithms: A Review", *Journal of Traffic and Transportation Engineering (English Edition)*, vol. 6, issue 5, 2019. https://doi.org/10.1016/j.jtte.2019.07.001

[9] B. Barán and M. Schaerer, "A Multiobjective Ant Colony System for Vehicle Routing Problem with Time Windows", *Applied Informatics*, pp. 97–102, 2003.

[10] M. Gendreau *et al.*, "Neighborhood Search Heuristics for a Dynamic Vehicle Dispatching Problem with Pick-Ups and Deliveries", *Transportation Research Part C: Emerging Technologies* vol. 14, issue 3, pp. 157–174, 2006. https://doi.org/10.1016/j.trc.2006.03.002

[11] H. Housroum *et al.*, "A Hybrid GA Approach for Solving the Dynamic Vehicle Routing Problem with Time Windows", in *Proc. of the 2nd International Conference on Information & Communication Technologies*, 2006, pp. 787–792. https://doi.org/10.1109/ICTTA.2006.1684473

[12] O. B. Madsen *et al.*, "A Heuristic Algorithm for a Dial-a-Ride Problem with Time Windows, Multiple Capacities, and Multiple Objectives", *Annals of operations Research*, vol. 60, no.1, pp. 193–208, 1995. https://doi.org/10.1007/BF02031946

[13] L. Fu and S. Teply, "On-Line and Off-Line Routing and Scheduling of Dial-a-Ride Paratransit Vehicles", *Computer-Aided Civil and Infrastructure Engineering*, vol. 14, no. 5, pp. 309–319, 1999. https://doi.org/10.1111/0885-9507.00150

[14] M. E. Horn, "Fleet Scheduling and Dispatching for Demand-Responsive Passenger Services", *Transportation Research Part C: Emerging Technologies*, vol. 10, no. 1, pp. 35–63, 2002. https://doi.org/10.1016/S0968-090X(01)00003-1

[15] M. Diana, "The Importance of Information Flows Temporal Attributes for the Efficient Scheduling of Dynamic Demand Responsive Transport Services", *Journal of Advanced Transportation*, vol. 40, no. 1, pp. 23–46, 2006. https://doi.org/10.1002/atr.5670400103

[16] S. R. Thangiah *et al.*, "An Agent Architecture for Vehicle Routing Problems", in *Proc. of the 2001 ACM Symposium on Applied Computing*, 2001, pp. 517–521. https://doi.org/10.1145/372202.372445

[17] R. Kohout *et al.*, "In-Time Agent-Based Vehicle Routing with a Stochastic Improvement Heuristic", *AAAI/IAAI*, pp. 864–869, 1999.

[18] K. Fischer *et al.*, "A Model for Cooperative Transportation Scheduling", *ICMAS*, pp. 109–116, 1995.

[19] F. Grootenboers *et al.*, "Impact of Competition on Quality of Service in Demand Responsive Transit", in *Proc. of the German Conference on Multiagent System Technologies*, Springer, Berlin, Heidelberg, pp. 113–124, 2010. https://doi.org/10.1007/978-3-642-16178-0_12

[20] Y. Nagata *et al.*, "A Penalty-Based Edge Assembly Memetic Algorithm for the Vehicle Routing Problem with Time Windows", *Computers & Operations Research*, vol. 37, no. 4, pp. 724–737, 2010. https://doi.org/10.1016/j.cor.2009.06.022

[21] M. Diana and M. M. Dessouky, "A New Regret Insertion Heuristic for Solving Large-Scale Dial-a-Ride Problems with Time Windows", *Transportation Research Part B: Methodological*, vol. 38, no. 6, pp. 539–557, 2004. https://doi.org/10.1016/j.trb.2003.07.001

[22] M. Gendreau *et al.*, "Parallel Tabu Search for Real-Time VehicleRouting and Dispatching", *Transportation science*, vol. 33, no. 4, pp. 381–390, 1999. https://doi.org/10.1287/trsc.33.4.381

[23] M. J. North *et al.*, "The Repast Simphony Runtime System", in *Proc. of the Agent 2005 Conference on Generative Social Processes, Models, and Mechanisms*, vol. 10, pp. 13–15, 2005.

[24] M. Zargayouna *et al.*, "Multiagent Simulation of Real-Time Passenger Information on Transit Networks", *IEEE Intelligent Transportation Systems Magazine*, vol. 12, issue 2, pp. 50–63, 2020. https://doi.org/10.1109/MITS.2018.2879166

[25] M. Zargayouna and B. Zeddini, "Fleet Organization Models for Online Vehicle Routing Problems", *Transactions on Computational Collective Intelligence VII*, Springer, Berlin, Heidelberg, pp. 82–102, 2012. https://doi.org/10.1007/978-3-642-32066-8_4

[26] M. M. Solomon *et al.*, "Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints", *Operations Research*, vol. 35, no. 2, pp. 254–265, 1987.

Received: August 2018
Revised: October 2019
Accepted: May 2020

Contact addresses:
Mahdi Zargayouna
Gustave Eiffel University, IFSTTAR, COSYS, GRETTIA
Champs sur Marne
France
e-mail: mahdi.zargayouna@univ-eiffel.fr

Besma Zeddini
SATIE, UMR CNRS 8029 ENS Cachan, CY Tech
Cergy-Pontoise
France
e-mail: bzi@eisti.eu

Mahdi Zargayouna is a researcher at Gustave Eiffel University (France) and deputy director of the GRETTIA Laboratory. He is also head of the Intelligent Systems & Application MSc Program. He received his MSc, PhD and Habilitation degrees in computer science and artificial intelligence from the University of Paris Dauphine (France) in 2003, 2007, and 2019 respectively. He is mainly interested in multi-agent systems (languages, coordination models, simulation, optimization, *etc.*), and dynamic transportation applications (traveler information, crisis management, dial-a-ride, urban parking, *etc.*). He has published more than 70 papers in peer-reviewed journals and conference proceedings and is a member of the reviewer boards of several international journals and conferences.

Besma Zeddini is an assistant professor in computer science at CY Tech Engineering School and a researcher at CNRS-SATIE Laboratory (France). She is the Partnership and Valorization coordinator of CY Tech. She is the co-leader of the engineering program on Cybersecurity and Smart Systems and head of the IoT MSc program. Besma Zeddini received her MSc and PhD degrees in computer science from the University of Le Havre Normandie (France). Her work is mainly focused on artificial intelligence, complex systems, and simulation of intelligent transportation systems. She has authored more than 40 papers in international journals and conferences.