Dispatching Strategies for Dynamic Vehicle Routing Problems

Besma Zeddini, Mahdi Zargayouna

Dispatching Strategies for Dynamic Vehicle Routing Problems

Besma Zeddini[1], Mahdi Zargayouna[2]

[1] Quartz, EISTI
Avenue du Parc,
95000 Cergy Pontoise, France.
[2] Université Paris-Est, IFSTTAR, GRETTIA
Boulevard Newton, Champs sur Marne
F-77447 Marne la Vallée Cedex 2, France
bzi@eisti.eu hamza-mahdi.zargayouna@ifsttar.fr

Abstract. Online vehicle routing problems are highly complex problems for which several techniques have been successfully proposed. Traditionally, the solutions concern the optimization of conventional criteria (such as the number of mobilized vehicles and the total traveled distance). However, in online systems, the optimization of the response time to the connected users becomes at least as important as the optimization of the traditional criteria. Multi-agent systems and greedy insertion heuristics are the most promising approaches to optimize this criteria. To this end, we propose a multi-agent system and we focus on the clients dispatching strategy. The strategy decides which agents perform the computation to answer the clients requests. We propose three dispatching strategies: centralized, decentralized and hybrid. We compare these three approaches based on their response time to online users. We consider two experiments configuration, a centralized configuration and a network configuration. The results show the superiority of the centralized approach in the first configuration and the superiority of the hybrid approach in the second configuration.

1 Introduction

Several real-life distribution applications, such as the good deliveries to stores, the school buses routing, the newspapers and mail distribution, etc. are instantiations of vehicle routing problems (VRP). In its original version, a VRP is a multi-vehicle traveling salesman problem. A number of nodes have to be visited only one time by a number of vehicles. The problem objective is generally to find a set of routes for the vehicles that optimize the number of mobilized vehicles and the total traveled distance. Solving these problems has high practical usefulness and they are challenging optimization problems with stimulating issues. The problem variant with time (and capacity) constraints is one of the most widely studied variants of VRP (vehicle routing problem with time windows, VRPTW henceforth) [1]. In this variant, the requests to be handled are not simple nodes,

but clients who define a quantity to be transported, a node to be visited and two temporal bounds between which it has to be visited by a vehicle. Vehicles have limited capacities and the quantities associated with the clients in the same route must not be bigger than the capacity of the concerned vehicle.

Vehicle routing problems can be divided in two categories: static problems and dynamic problems. In the static problems, the system knows all the problem data before execution. In the dynamic problems, the problem data reveals as the optimization is being performed. The data may concern any entity of the problem, such as the traffic data or the available vehicles, but the dynamism usually refers to the clients to be served. The operational problems are never completely static and we can say that a static system cannot meet nowadays operational configurations anymore. Indeed, in real-life vehicle routing problems, and even when all the clients are known in advance (with a reservation system for instance), there always exists some element that makes the problem actually dynamic. These elements might concern no-shows, delays, breakdowns, etc. Online vehicle routing problems could be seen as an extreme case of dynamic vehicle routing problems. Indeed, not only the problem data, and specifically the clients, are not known before the optimization starts, but the clients connect in real-time to the system and expect quasi-immediate answers to their requests. The response time of the system in this configuration is then vital. If the system needs, say, two more minutes to gain one or two kilometers in its routes, it is not worth it in online problems, since the client will not wait that long to have an answer to its request.

To meet the requirement of short response times, we rely on the multi-agent paradigm for solving the online vehicle routing problems. An agent is an intelligent entity that is situated in an environment and that applies autonomous actions to satisfy its objectives [2, 3]. A multi-agent modeling of the online VRP is relevant for the following reasons. On the one side, choosing a design allowing for computing distribution should provide shorter response times to clients requests. On the other side, nowadays vehicles are more and more connected, and have onboard computers. In this context, the transport system is, *de facto*, distributed and necessitates an adapted modeling to take profit of these equipments. The multi-agent system (MAS) that we propose in this paper simulates a distributed version of the so-called "insertion heuristics". These are methods that consist in inserting the clients following their appearance order in the routes of the vehicles. The vehicle chosen to insert the considered client is the one that would have the minimal additional cost to visit it (the incurred detour for instance). This is the fastest known heuristic, since there is no reconsideration of previous insertion decisions. In this context, there is still a choice to perform with respect to the dispatching of clients requests to the vehicle agents of the multi-agent systems. We propose three dispatching strategies and we compare them following their ability to provide better response times to the clients. The dispatching strategy decides which agents perform the computation to answer the clients requests. In the centralized strategy, the planner agent performs most of the computation. In the decentralized strategy, the vehicle agents perform

most of the computation in a collaborative way. Finally, in the hybrid strategy the work is split between clients and vehicles.

The remainder of this paper is structured as follows. In section 2, we discuss previous proposals for the dynamic VRP w.r.t our approach. The multi-agent system and the three dispatching strategies architecture of the MAS are presented in section 3. We provide our experimental results in section 4 and then conclude with a few remarks in section 5.

2 Related Work

The majority of the proposed solution methods to vehicle routing problems are heuristic or metaheuristic methods, which provide good results in non-exponential times, and which have presented good results with benchmark problems. Generally speaking, most of the works dealing with the dynamic VRP are more or less direct adaptations of static methods. Among the static methods, insertion heuristics are the most widely adapted in a dynamic environment (e.g. [4]). Insertion heuristics are, in their original version, greedy algorithms, in the sense that the decision to insert a given client in the route of a vehicle is definitive. The advantage of using insertion heuristics is that they are intuitive and fast.

In their vast majority, multi-agent approaches of the literature rely, at least partially, on insertion heuristics. In [5], Thangiah *et al.* propose a multi-agent architecture to solve a VRP and a multi-depot VRP. In [6], Kohout and Erol 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 [5], the clients are handled sequentially. They are broadcasted to all the vehicles, which in turn propose insertion offers and the best proposal is retained by the client. In the second step, the vehicles exchange clients to improve their solutions, each vehicle knowing the other agents of the system. Since vehicles are running in parallel, the authors envision to apply different heuristics for each vehicle, without changing the architecture.

For the reasons that we have given in the introduction, we choose a multi-agent modeling to solve the dynamic VRP. For their fast execution times and their adaptation to dynamic settings, we privilege a solving grounded on insertion heuristics. Thus, from a protocol and an architecture point of view, our system sticks with the multi-agent systems we have just described, since we propose a distributed version of insertion heuristics. However, in these proposals, none have focused on the response time of the system to online clients. In our previous works (e.g. [7–9], we have addressed the optimization criteria of the VRPTW. In this paper, we do not focus on the optimization problem for itself. Our focus here is on the three dispatching strategies, and our result indicate which one is the best, with respect to the chosen implementation configuration.

3 Dispatching strategies

Each solution for a given vehicle routing problem instance is a set of vehicles with a specific route. Each vehicle's route is composed of a sequence of clients, together with their corresponding visit time. The three requests dispatching strategies that we propose in this paper are defined in the framework of a multi-agent system. Three categories of agents are defined in the system. The client agents represent users of the system (persons or goods, depending on the problem). The vehicle agents represent vehicles and the interface agents represent the interlocutor with the external world (GUI, simulator, etc.). When a user logs to the system, the interface agent create a representing client agent, representing the human user. A fourth agent type is defined for the only centralized dispatching strategy, which is the planner agent and is responsible of performing all the routing.

In online problems, the response time of the system is key, and only very fast approaches can compete in this configuration. The fastest approach, and the most popular one is the greedy insertion approach, originally proposed by Marius M. Solomon [1]. The principle is to insert clients progressively in the vehicles routes. To do so, the insertion price of inserting a client in the route of a vehicle is calculated, and the vehicle with the minimal price is chosen for inserting the client. To compute this insertion price, the cost of the current itinerary (the total traveled distance) and the cost of the new itinerary are compared. The difference between the two quantities is the additional effort or insertion price for the new client's insertion. Determining the chosen vehicle consists in selecting the vehicle with the minimal insertion price.

When the solving system is a MAS, there are several alternatives regarding who handles the request. Each alternative is called a "dispatching strategy". In this section, we describe and compare three possible dispatching strategies that we have designed, implemented and compared to model the dynamic VRP: a centralized dispatching, a decentralized dispatching and a hybrid dispatching. The objective is to check which dispatching strategy is the most effective, in terms of response time to clients requests. The evaluation of the different strategies does not consider the traditional optimization criteria (number of mobilized vehicles, total traveled distance and total waiting time). Indeed, in terms of optimization, the three dispatching strategies follow the same algorithm, the only difference concerns the response time, i.e. the time that the system takes to decide about which vehicle will serve the customer.

3.1 Centralized dispatching

In the centralized approach, all the treatments are performed by a central entity, which create vehicle plans and schedules. One of the main advantages of this approach is that it allows for central online optimization techniques. Online optimization (e.g. in [10]) allows to profit from optimization techniques, while reducing response times. The principle is to discretize the processing time into time intervals. During each interval an optimization is performed with the known

clients. The new clients are kept in a queue, waiting for the next interval. The known clients that could not be served and the new clients are submitted for the new optimization round. As we said in the previous section, our objective is to compare the same solving approaches while comparing response times, our centralized approach then mimics insertion heuristics. In our proposal (see Fig. 1 (left)), all client requests are treated by the same planner agent. The planner agent has all the necessary information about each vehicle and each client and their current status. With these information, it assigns the current client in the least costly position between all the possible vehicles.

The scenario is the following. A user appears and interacts with an interface agent who creates a client agent representing him. When created, the client sends a request to the planner agent, who tries to insert it in the route of every vehicle of the system, in every feasible position. To this end, it executes sequentially, for each vehicle, a procedure computing the insertion price for the vehicle, and chooses the vehicle and the insertion position with the minimal price. If no vehicle can insert the client a new vehicle agent is created and the client is inserted in the only possible position in its route. Finally, the planner informs the client and the vehicle of the outcome of the procedure. Vehicle agents in the centralized dispatching strategy do not perform any calculation and only acknowledge the updates in their routes.

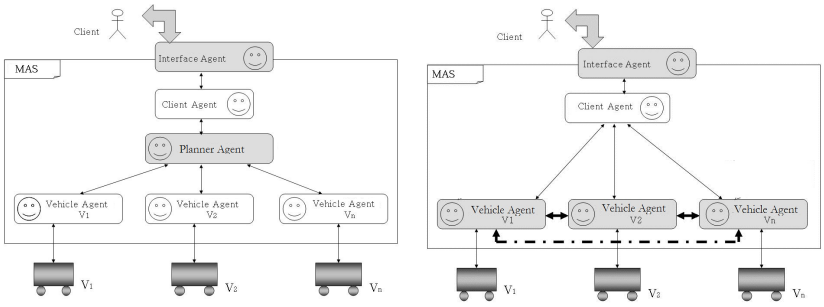

Fig. 1. Centralized architecture (left) and decentralized architecture (right)

The centralized dispatching poses two main problems. On the one side, it is not possible to distribute the execution over several hosts in order to limit the response time of the system, which is the primary concern in online systems. On the other side, the breakdown of the planner agent would result in a complete breakdown of the system. Nevertheless, the centralized dispatching strategy offers the advantage of minimizing the communications between agents, which are restricted to the notifications of the computing outcome to the clients and vehicles ($N(1+V)$), with N the number of clients and V the number of vehicles.

3.2 Decentralized dispatching

The decentralized dispatching is illustrated in Fig. 1(right). Following this method, there is no bottleneck for routes calculations. Following the principle of the greedy insertion heuristics, every vehicle agent tries to insert the new client in its route, and proposes an insertion price, corresponding to the "cheapest" position where it can insert the client. The chosen vehicle will be the one having the minimal insertion price to transport the client.

In this dispatching strategy, the choice of the vehicle with the minimal price, the computation of costs, and the choice of the vehicle, all these steps are performed in a distributed way. Indeed, the scenario is the following. When a new client shows up, it broadcasts its request to all the vehicles of the system. When the request is received, every vehicle computes its insertion price. When it finishes its computation, the vehicle broadcasts a message to all the vehicles with its identifier and its price. For the processing of these messages and the inference of the winner vehicle agent, we propose the following process.

Every vehicle agent broadcasts its own computed price to the other vehicle agents. When he receives a new message containing a price that was computed by another agent, he sorts the received offers, including its own offer, following their prices. When all the other vehicle agents have proposed a price, the vehicle agent either checks if it is the best vehicle. If so, it updates its route with the new inserted client.

This dispatching strategy offers the advantage of completely distributing the processing and to be fault-tolerant. Indeed, breakdowns might occur for the agents, which would block the whole system (cf. centralized dispatching). In this approach in the contrary, for each new client request, the vehicles have to negotiate to choose which one is the most appropriate to serve the client, instead of a central entity that would decide for them. However, the number of exchanged messages might increase dramatically, which is generally the price to pay for a distribution of the processing. The number of exchanged messages between vehicles with this dispatching strategy is equal to $N \times V^2$. The overall number of messages is equal to $N(1+V(1+V))$

3.3 Hybrid dispatching

The hybrid dispatching is a compromise between the centralized approach and the decentralized approach. In the hybrid approach (cf. Fig. 2), the client agent plays the role of a dispatcher. The client agent broadcasts the client request, collects the offers of the vehicle agents and chooses the one proposing the minimal price.

The hybrid approach follows the following protocol. A new user provides the interface agent the information concerning his transport request. The interface agent creates a client agent representing him. Then, the new client agent broadcasts a message to all the vehicles. Every vehicle agent verifies if it can insert the client in its route. The vehicle agent then sends its price to the client agent. The client agent collects the answers of the vehicles and chooses the vehicle that

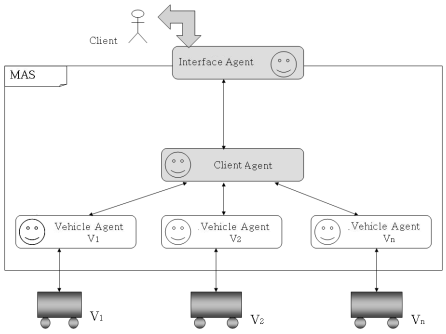

Fig. 2. Hybrid architecture

proposes the minimal price. Once it has chosen the best vehicle that can answer the new request (if there is at least one that can insert the client), it broadcasts a new message to the vehicles informing them about its decision and asking the winner vehicle agent to insert it in its route and to serve it. When the vehicle receives the message from the client informing it that it is the winner, it updates its route and inserts the client.

Thus, the objective of the hybrid approach is to relax the planner from all the calculation, and to limit the communications between vehicles. The overall number of messages in the hybrid dispatching strategy is equal to $3VN$.

4 Experiments

Our objective is to verify the impact of dispatching strategies on the response time of time-constrained online vehicle routing systems. We have generated several client files with 100, 200, 300 et 400 clients, while varying the number of vehicles between 4 and 8. The spatial environment is a plane of 50×50 and the depot is in the center of the plane. The customers are geographically uniformly distributed with time windows varying between 10 minutes and one hour. The service time is set to 5 minutes. The quantities associated with clients requests are between 5 and 20 while vehicle capacity is set to 400. The scheduling horizon is set to 10 hours. For each client, we have also to define its appearance time, i.e. the moment when it becomes known by the system. We have used the Gendreau [11] method for the definition of these moments. Clients appear between 30 minutes and 1 minute before the start of their time window. Provided the high level of randomness, we have executed each type of simulation 50 times and we report the average result values.

4.1 Centralized experiments

We have implemented the three dispatching strategies using the multi-agent platform REPAST Simphony [12]. The simulation is made of 7200 discrete simulation

ticks. Each tick corresponds to 5 seconds in the real world. Clients appearance times are transformed into "appearance ticks" and continuously feed the simulation at the computed ticks. We have executed our experiments on a PC with an Intel Xeon E7-4820 processor, and 50 GB of RAM. Since we use the same deterministic algorithm for all dispatching strategies, which is a distributed version of insertion heuristics, the results for the three architectures in terms of optimization costs are the same and not reported here.

Parameters / Approaches	Nb vehicles	Nb clients	Average response time (ms)
Centralized	4	100	33
	4	200	43
	8	300	50
	8	400	62
Hybrid	4	100	38
	4	200	51
	8	300	59
	8	400	72
Decentralized	4	100	46
	4	200	63
	8	300	94
	8	400	113

Table 1. Centralized configuration

The TABLE 1 provides the values in terms of average response times (in milliseconds) of every dispatching strategies. The response time for a client is the difference between the moment when the client agent is created and the moment when a vehicle is chosen by the client. The centralized architecture provides the best results, followed by the hybrid architecture and the decentralized architecture. This is due to the fact that the centralized approach does not generate communications between agents and does not suppose any concurrency management. The hybrid approach provides results that are close to the centralized dispatching strategy. 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 client agent and the vehicle agents. On the other side, the management of concurrent processes of the vehicles and clients, and the fact that their contexts have to be restored every time the scheduler executes them, increases the exhibited response times for the clients. Finally, the distributed approach suffers from the two drawbacks: it generates a quadratic number of messages and it uses pseudo-parallelism which slows down the processing.

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

4.2 Network experiments

It is possible with Repast Simphony to distribute a simulation on a network. We have deployed our three systems (one for each dispatching strategy) on a four PC network, each with the same configuration (Intel Xeon E7-4820 processor, and 50 GB of RAM). We report the new obtained results in Table 2.

Parameters / Approaches	Nb vehicles	Nb clients	Average response time (ms)
Hybrid	4	100	18
	4	200	26
	8	300	29
	8	400	32
Decentralized	4	100	23
	4	200	33
	8	300	42
	8	400	53
Centralized	4	100	35
	4	200	45
	8	300	53
	8	400	64

Table 2. Networked configuration

These results are interesting since they provide a new enlightenment concerning the most promising dispatching strategy in terms of response time to online users. Indeed, in the absence of slow-down due to single PC pseudo-parallelism, the hybrid architecture takes profit of the processing distribution, without suffering from a too big number of exchanged messages. The distributed architecture comes in the second position in terms of performances, taking profit from the distribution but suffering from their too big bandwidth consumption. The centralized architecture comes in the last position, since its gain in terms of exchanged messages does not counterbalance its sequentialization of processing. Anyway, this architecture provides results that are practically equivalent to a centralized implementation. The small difference comes from the fact that vehicle agents are executed in different hosts than the planner agent, which result in a small additional cost in terms of communication.

5 Conclusion

In this paper, we have proposed a multi-agent system with three versions, focusing on clients dispatching strategies. The dispatching strategy decides which agents perform the computation to answer the clients requests. In the centralized strategy the planner agent performs most of the computation. In the decentralized strategy, the vehicle agents perform most of the computation in a collaborative way. Finally, in the hybrid strategy the work is split between clients and

vehicles. We have compared these three approaches based on their response time to online users. We have considered two experiments configuration, a centralized configuration and a network configuration. The results have shown the superiority of the centralized approach in the first configuration and the superiority of the hybrid approach in the second configuration. In our future works, we will consider more dynamic problems, in which, not only clients are not known before execution, but also traffic conditions. To this end, we will integrate our vehicle routing system inside the multimodal traffic simulator SM4T [13].

References

1. Solomon, M.: Algorithms for the vehicle routing and scheduling with time window constraints. Operations Research **15** (1987) 254–265
2. Wooldridge, M., Jennings, N.R.: Intelligent agents: Theory and practice. Knowledge Engineering Review **10**(2) (1995) 115–152
3. Bessghaier, N., Zargayouna, M., Balbo, F.: Management of urban parking: an agent-based approach. In: International Conference on Artificial Intelligence: Methodology, Systems, and Applications, Springer (2012) 276–285
4. Diana, M.: The importance of information flows temporal attributes for the efficient scheduling of dynamic demand responsive transport services. Journal of advanced Transportation **40**(1) (2006) 23–46
5. Thangiah, S.R., Shmygelska, O., Mennell, W.: An agent architecture for vehicle routing problems. In: Proceedings of the 2001 ACM symposium on Applied computing (SAC '01), New York, NY (USA), ACM Press (2001) 517–521
6. Kohout, R., Erol, K.: In-Time agent-based vehicle routing with a stochastic improvement heuristic. In: Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence (AAAI'99/IAAI'99), Menlo Park, CA (USA), AAAI Press (1999) 864–869
7. Zeddini, B., Temani, M., Yassine, A., Ghedira, K.: An agent-oriented approach for the dynamic vehicle routing problem. In: IWAISE'08, IEEE (2008) 70–76
8. Zargayouna, M., Balbo, F., Scemama, G.: A multi-agent approach for the dynamic vrptw. In: ESAW 08. (2008)
9. Zargayouna, M., Zeddini, B.: Fleet organization models for online vehicle routing problems. In: Transactions on Computational Collective Intelligence VII. Springer (2012) 82–102
10. Grootenboers, F., de Weerdt, M., Zargayouna, M.: Impact of competition on quality of service in demand responsive transit. In Dix, J., Witteveen, C., eds.: MATES 2010. Volume 6251 of Lecture Notes in Computer Science., Springer (2010) 113–124
11. Gendreau, M., Guertin, F., Potvin, J.Y., Taillard, E.D.: Parallel tabu search for real-time vehicle routing and dispatching. Transportation Science **33**(4) (1999) 381–390
12. North, M.J., Howe, T.R., Collier, N.T., Vos, R.J.: The repast simphony runtime system. Agent 2005 Conference on Generative Social Processes, Models, and Mechanisms (2005)
13. Zargayouna, M., Zeddini, B., Scemama, G., Othman, A.: Simulating the impact of future internet on multimodal mobility. In: AICCSA'2014, IEEE Computer Society (2014)