A Multi-agent Approach to the Dynamic Vehicle Routing Problem with Time Windows

Dariusz Barbucha

Department of Information Systems
Gdynia Maritime University
Morska 83, 81-225 Gdynia, Poland
d.barbucha@wpit.am.gdynia.pl

Abstract. The term Dynamic Vehicle Routing refers to a wide range of transportation problems where the required information is not given a priori to the decision maker but is revealed concurrently with the decision-making process, where the goal of such process is to provide the required transportation and at the same time minimize service cost subject to various constraints including vehicle and fleet capacities. The most common source of dynamism in vehicle routing problem is the online arrival of customer during the operations, which increases the complexity of decisions and introduces new challenges while finding the optimal route plan. The paper proposes a new agent-based approach to the Dynamic Vehicle Routing Problem with Time Windows, in which two different dynamic order dispatching strategies are considered. Their influence on the results are investigated and identified in the computational experiment.

Keywords: Dynamic Vehicle Routing Problem with Time Windows, Multi-agent Systems, Cooperative Problem Solving.

1 Introduction

One of the most important problem in the contemporary transport companies is the Vehicle Routing Problem where a set of vehicles have to deliver (or pickup) goods or persons to (from) locations situated in a given area. While customer requests can either be known in advance or appear dynamically during the day, vehicles have to be dispatched and routed in the real time, possibly, by taking into account changing traffic conditions, uncertain demands, or varying service times [9].

Recent advances in information and communication technologies can help such companies to manage vehicle fleets in the real-time. When jointly used, technologies like Geographic Information Systems (GIS), Global Positioning Systems (GPS), traffic flow sensors and cellular telephones are able to provide relevant real-time data, such as current vehicle locations, and periodic estimates of road travel times [7].

There exist several important dynamic routing problems that are being solved in the real-time. Most representative examples, including transport of goods,

C. Bădică, N.T. Nguyen, and M. Brezovan (eds.): ICCCI 2013, LNAI 8083, pp. 467–476, 2013.

arise in dynamic fleet management and courier services. Others, focused on personal transportation services, include dial-a-ride problems, emergency and taxi cab services [7].

The paper aims at proposing a new multi-agent approach for solving the Dynamic Vehicle Routing Problem with Time Windows (DVRPTW), where the set of customer requests has to be served by the set of available vehicles in order to minimize the vehicle fleet size and the total distance needed to pass by these vehicles, and satisfying several customers and vehicles constraints. According to Pillac [9] and Ghiani [7] classifications, the considered problem belongs to *dynamic* and *deterministic* routing class, where all data are known in advance, and the input orders are revealed dynamically and unpredictably during the execution of orders.

In contrast to its static counterpart, dynamic VRPTW involves new features that increase the complexity of decisions and introduces new challenges while finding the optimal route plan. Because of the fact, that new customer orders can continuously arrive over time, at any moment of time, there may exist customers already under servicing and new customers which need to be serviced. As a consequence, each newly arriving dynamic request, needs to be incorporated into the existing vehicles tours and the current solution may need to be reconfigured to minimize the goal functions.

The rest of the paper is organized as follows. Section 2 includes formulation of the VRPTW. An overview and the main elements of the proposed agent-based approach for VRPTW are presented in Section 3. Section 4 presents results of computational experiment, and finally, Section 5 concludes the paper and suggests the directions of future research.

2 Formulation of the Vehicle Routing Problem with Time Windows

The *static* Vehicle Routing Problem with Time Windows (VRPTW) can be formulated as the problem of determining optimal routes passing through a given set of locations (customers) and defined on the undirected graph $G = (V, E)$, where $V = \{0, 1, \ldots, N\}$ is the set of nodes and $E = \{(i, j)|i, j \in V\}$ is the set of edges. Node 0 is a central depot with NV identical vehicles of capacity W. Each node $i \in V \setminus \{0\}$ denotes a customer characterized by a non-negative demand d_i, and a service time s_i. Moreover, with each customer $i \in V$, a time window $[e_i, l_i]$ wherein the customer has to be supplied, is associated. Here e_i is the earliest possible departure (ready time), and l_i - the latest time the customer request has to be started to be served. The time window at the depot ($[e_0, l_0]$) is called the scheduling horizon.

Each link $(i, j) \in E$ denotes the shortest path from customer i to j and is described by the cost c_{ij} of travel from i to j by shortest path ($i, j \in V$). It is assumed that $c_{ij} = c_{ji} (i, j \in V)$. It is also often assumed that c_{ij} is equal to travel time t_{ij}.

The goal is to minimize the vehicle fleet size and the total distance needed to pass by vehicles in order to supply all customers in their required time windows

(minimization of the fleet size is often considered to be the primary objective of the VRPTW), satisfying the following constraints:

- each route starts and ends at the depot,
- each customer $i \in V \setminus \{0\}$ is serviced exactly once by a single vehicle,
- the total load on any vehicle associated with a given route does not exceed vehicle capacity,
- each customer $i \in V$ has to be supplied whithin the time window $[e_i, l_i]$ associated with it (the vehicle arriving before the lower limit of the time window causes additional waiting time on the route),
- each route must start and end within the time window associated with the depot.

The *dynamic* VRPTW considered in the paper can be seen as an extension of the static VRPTW where all customer requests are dynamic (they arrive while the system is already running) and hence the optimisation process has to take place in the real-time. Let the planning horizon starts at time 0 and ends at time T. Let $t_i \in [0, T]$ $(i = 1, \ldots, N)$ denotes the time when the $i-th$ customer request is submitted.

Whereas during the recent years there have been many important advances in the field of static versions of different variants of VRP (the reader can find a review of different methods proposed for solving VRPTW for example in two papers of Braysy and Gendreau [4,5]), definitely much less has been done with respect to solving their dynamic versions. Dynamic vehicle routing problems, their variants, methods of solving them and examples of practical application can be found in one of the last paper of Pillac [9].

In recent years only few approaches based on using intelligent agents for solving some transportation problems have been proposed. The survey of existing research on agent-based approaches to transportation management presented by Davidson et al. [6] shows that this field is still promising and worth further exploration.

3 Multi-agent Approach for VRPTW

3.1 Overview

Technically, the proposed approach extends the platform for solving dynamic VRP proposed in [1,2]. Architecture of the approach is based on JADE (Java Agent Development Framework), a software framework for the development and run-time execution of peer-to-peer applications [3].

Within the proposed platform several types of autonomous agents are used. They are `GlobalManager`, `OrderGenerator`, `OrderManager`, and `Vehicle`. Each agent encapsulates a particular abilities and during the process of solving the problem agents play their roles cooperating in order to achieve the common goal. The list of agents and their characteristics (description, attributes and communication with other agents) are included in Tab. 1.

Table 1. Agents and their characteristics

`GlobalManager`	An agent which runs first and initializes all others agents.
Atributes:	

Communication:	`GlobalManager` $\longrightarrow$ `OrderGenerator`
	`GlobalManager` $\longrightarrow$ `OrderManager`
	`GlobalManager` $\longrightarrow$ `Vehicle`
	It initializes all agents.

`OrderGenerator`	An agent which generates (or reads) new orders.
Atributes:	l_R - list of all dynamic requests, l_dR - list of actual dynamic requests.
Communication:	`OrderGenerator` $\longrightarrow$ `OrderManager` It sends new orders to the `OrderManager` agent.
`OrderManager`	An agent which manages the list of requests.
Atributes:	$R = \{R_1, R_2, \ldots, R_{NV}\}$ - list of all routes,

Communication:	`OrderManager` $\longleftarrow$ `OrderManager`
	`OrderManager` $\longrightarrow$ `Vehicle`
	After receiving the new request from `OrderGenerator`, `OrderManager` inserts it to the list of dynamic orders and allocates it to the available `Vehicle` agents.

`Vehicle`	An agent that represents a vehicle and serves the customers' orders.
Atributes:	$(x0, y0)$ - depot's coordinates on the plane, W - the capacity of the vehicle, Ri - actual route assigned to this vehicle, $cost(Ri)$ - actual cost of the route, Wr - actual available space, v - speed, sV - vehicle state (*waiting*, *driving*, *stopped*), tsV - total time spend by vehicle in the system, twV - vehicle's total waiting time.
Communication:	`OrderManager` $\longleftrightarrow$ `Vehicle` Most of its lifetime, a vehicle spends serving requests. It starts after receiving and accepting the first request. After reaching the nearest customer it proceeds to the next customer belonging to the route or waits for next order. If the vehicle reaches the last customer on the current route, it waits in the location until a new request arrives. When all requests are dispatched among the available vehicles, the waiting vehicle returns back to the depot. Periodically `Vehicle` receives customer's request from the `OrderManager` one at a time, and tries to assign it to the existing route in order to perform it.

Additionally, all the above agents operate on customer order represented by `Customer` class in the system. Each order is described by the set of attributes described above, and, additionally, by: ts_i - time in which an order is served, sC_i - state of the order (*available*, *blocked*, *finished*, *canceled*), and twC_i - customer's waiting time for servicing.

3.2 Process of Solving DVRPTW

The process of solving DVRPTW (performed in the loop until end of request is reached) is divided into general steps, presented as Algorithm 1.

Algorithm 1. MAS-DVRPTW

1: Initialize all agents: `GlobalManager`, `OrderGenerator`, `OrderManager`, and `Vehicle`
2: **while** (`end_of_orders` has not been reached) **do**
3: `OrderGenerator` agent generates (or reads) new order and sends it to the `OrderManager`,
4: `OrderManager` incorporates it into the list of orders to be served and allocates it to the available Vehicle agents, using predefined dispatching strategy.
5: **end while**

`GlobalManager` agent runs first and next it initializes all other agents. After receiving messages from all agents about their readiness to act, the system is waiting for events. Three kinds of events are genertaed in the proposed approach. The first group includes *system events* (among them the most important is the `end_of_orders` event). The second group (*order events*) includes main event observed in the system - `new_order` event. The last group of events (*vehicle events*) are mainly sent by `Vehicle` agents which reports their states during serving a set of requests assigned to them (`vehicle_drive`, `vehicle_stop`, `vehicle_wait`, `vehicle_reached_location(p)`, etc.). The above algorithm focuses on main steps of solving the problem without deeper insight into the simulation part, so details of the flow of messages are omitted here.

What is important, the above process of solving the problem is performed in a dynamically changing environment. It means that at the arrival of the new order, all vehicles are serving the customers already assigned to their routes, and introduction of a new request to the routes requires re-optimization procedure. In the proposed approach it has been decided to implement two strategies of dispatching dynamic requests to the available vehicles. They reflect the level at which the list of customer orders is maintained and are called *Decentralized* and *Centralized Dispatching Strategy*, respectively. The process of (re-)optimization is organized in different way in both of them.

3.3 Dispatching Strategies

Decentralized Dispatching Strategy. Using Decentralized Dispatching Strategy (DDS) the process of assigning a new customer order to the available vehicles is performed using the *Contract Net Protocol* (CNP) [11], which is composed of a sequence of five main steps, where agents, acting as *Initiator* or *Participant*, negotiate the contract. After receiving a new order the following steps are performed.

1. `OrderManager` initializes a session using the CNP and starts communication between `OrderManager` and `Vehicle` agents. As *Initiator* it announces the request to each `Vehicle` agent (moving and waiting vehicles) sending around the call for proposal (`cfp`) message. `Vehicle` (as *Participants* or *Contractors*) are viewed as potential contractors.
2. Each `Vehicle` agent after receiving the request (with customer data) from the `OrderManager`, calculates the cost of inserting a new customer into the existing route (using the Solomon's *I1* constructive heuristic for VRPTW [13]). If the insertion of a new customer into the existing route does not violate the problem constraints, the calculated cost of insertion is sent back (as the `propose` message) to the `OrderManager`. Otherwise, the `Vehicle` sends back the rejection (`reject`) message.
3. `OrderManager` after receiving proposals from all `Vehicle` agents, chooses the one with the lowest cost of insertion. Next, it sends the `accept-proposal` message to the `Vehicle` which is awarded and the `reject-proposal` to the others.
4. `Vehicle` which receives the `accept-proposal` message, inserts the customer into its current route and sends the `inform-done` message if the operation is performed successfully and `failure` message, otherwise.
5. If all `Vehicle` agents rejected the proposal, then new order is assigned to new `Vehicle` agent located at the depot.

Using DDS, each `Vehicle` agent is autonomous and maintains its own list of orders. Let $v(i)$ be a `Vehicle` agent and let $R^i = [r^i_1, r^i_2, \ldots, r^i_k, \ldots, r^i_{length(R^i)}]$ be a current route assigned to the vehicle $v(i)$ $(i = 1, \ldots, NV)$. Assume that r^i_k is a customer (location), the vehicle $v(i)$ is currently driving to. Thus, the part of route $[r^i_1, r^i_2, \ldots, r^i_k]$ is fixed, and the process of assigning a new order to the existing route is possible only on positions $k+1, \ldots, length(R^i)$ of route $v(i)$. It is easy to see that if a particular request is arrising close to the end of planning horizon (highly dynamic order), possibility of reoptimization of the route assigned to vehicle $v(i)$ is limited.

After reaching a location of the current customer, `Vehicle` sends a message to `OrderManager` informing about it and continues serving next requests according to its autonomous routing plan.

Centralized Dispatching Strategy. In Centralized Dispatching Strategy (CDS) the list of all orders is maintained by `OrderManager` agent, and `OrderManager` is responsible for planning each vehicle route. On the other hand, `Vehicle` agent maintains short-term list of requests including only the location of the next order.

Similarly, as in DDS, let $v(i)$ be a `Vehicle` agent and let R^i be a current route assigned to the vehicle $v(i)$ $(i = 1, \ldots, NV)$. Let $R = [R^1, R^2, \ldots, R^{NV}]$ be a list of all routes assigned to each `Vehicle` agent $v(i)$ $(i = 1, \ldots, NV)$. In fact, it can be viewed as a solution of the routing problem.

Opposite to DDS, here, after receiving a new request, the `OrderManager` does not announce it immediatelly to the `Vehicle` agents but it buffers the

order and tries to insert to one of the active vehicles routes on positions $k+1, \ldots, length(R^i)$ of each route $v(i)$ (r^i_k is a customer (location), the vehicle $v(i)$ is driving to).

After reaching a location of the current customer r^i_k, each `Vehicle` agent $v(i)$ sends the message to `OrderManager` informing it about readiness for serving next requests. According to the current global routes plan, `OrderManager` sends the next customer's order for serving to the `Vehicle` agent $v(i)$.

In order to improve the solution currently maintained by `OrderManager`, parallel to the dispatching new orders to the available vehicles, `OrderManager` performs a set of operations trying to improve the current solution taking into account orders which have not been yet assigned to the vehicles yet. The process of improvement is performed by the set of four local search heuristics based on the following moves:

- a single customer from randomly selected position of individual is moved to another, randomly selected position,
- two customers from randomly selected positions of individual are exchanged,
- two randomly selected routes are disconnected and the remaining parts from different routes are reconnected again,
- two edges from two randomly selected routes are exchanged.

Centralized dispatching strategy can be viewed as a kind of *bufferring strategy*, where new requests are not allocated immediately to the vehicles but assignment of some requests to vehicles is delayed (see for example [10]).

4 Computational Experiment

In order to validate the proposed approach, a computational experiment has been carried out. It aimed at evaluating the influence of the kind of dispatching strategies on the performance of the system measured by the number of vehicles needed to serve all requests in the predefined time and the total distance needed to pass by these vehicles.

The proposed agent-based approach was tested on the classical VRPTW benchmark datasets of Solomon [13] transformed into the dynamic version through revealing all requests dynamically. The experiment involved six datasets of instances (R1, R2, C1, C2, RC1, RC2) [12] including 100 customers, each. Best known solutions identified by different heuristics for solving static VRPTW averaged over all solutions belonging to each group are presented in Tab. 2.

It is assumed that all requests are dynamic, and they may arrive with various frequencies. In the experiment arrivals of the dynamic requests have been generated using the Poisson distribution with λ parameter denoting the mean number of requests occurring in the unit of time (1 hour in the experiment). For the purpose of experiment λ was set to 5, 10, 15, and 30. It is also assumed that all requests have to be served. Additionally, it has been assumed that the vehicle speed was set at 60 km/h.

Table 2. Best known solutions identified by heuristics [12] averaged for each group of instances

Instance	Vehicles	Distance	Instance	Vehicles	Distance
R1	11.92	1209.89	R2	2.73	951.91
C1	10.00	828.38	C2	3.00	589.86
RC1	11.50	1384.16	RC2	3.25	1119.35

Each instance was repeatedly solved five times and mean results from these runs were recorded. All simulations have been carried out on PC Intel Core i5-2540M CPU 2.60 GHz with 8 GB RAM running under MS Windows 7.

The experiment results for both dispatching strategies: decentralized (DDS) and centralized (CDS), are presented in Tab. 3 and 4, respectively. Beside the name of the instance set and the value of λ parameter, next columns include the average number of vehicles used and the average distance travelled by all vehicles.

Table 3. Results obtained by the proposed approach (DDS strategy)

Instance	λ	#Vehicles	Distance	Instance	λ	#Vehicles	Distance
R1	5	13.72	2231.99	R2	5	4.11	1808.62
	10	13.12	1954.31		10	3.69	1370.79
	15	13.28	1694.18		15	3.89	1234.29
	30	13.20	1288.40		30	3.64	1024.23
C1	5	11.59	1499.37	C2	5	3.38	1114.83
	10	11.25	1100.71		10	3.00	732.04
	15	10.00	949.38		15	3.00	728.85
	30	10.00	916.49		30	3.00	619.84
RC1	5	13.54	2394.60	RC2	5	4.48	2160.35
	10	13.76	1884.80		10	4.60	1977.57
	15	13.61	1585.51		15	4.11	1328.20
	30	13.00	1412.81		30	4.15	1136.27

Analysis of the results presented in both tables allows one to conclude that dynamization of the VRPTW almost always results in deterioration of the results in comparison with its static counterpart. For the datasets used in the experiment, it is especially observed for instances with tight time horizon (C1, R1, RC1). It is easy to observe, that such deterioration also often depends on level of dynamism of the instance. For instances with a low ratio of request arrivals ($\lambda = 5 - 10$), the results are worse in comparison with the cases where customer request are known in advance or they arrive at early stage of computation ($\lambda = 30$).

An interesting conclusions can be provided by focusing observation on dispatching strategies proposed in the paper. The influence of the kind of strategies on the performance of the system is observed for almost all tested cases. The centralized strategy (CDS) outperforms decentralized (DDS) one, taking into account both observed factors: the number of vehicles needed to serve requests and the total distance needed to pass by these vehicles while serving them, but not with the same strength.

Table 4. Results obtained by the proposed approach (CDS strategy)

Instance	λ	#Vehicles	Distance	Instance	λ	#Vehicles	Distance
R1	5	13.58	2114.09	R2	5	3.87	1827.66
	10	13.54	1736.71		10	3.50	1191.66
	15	13.32	1469.36		15	3.48	1171.94
	30	13.09	1261.63		30	3.81	982.37
C1	5	10.97	1532.50	C2	5	3.00	1103.04
	10	10.55	1272.35		10	3.00	899.72
	15	10.00	870.94		15	3.00	718.83
	30	10.00	853.50		30	3.00	627.72
RC1	5	12.42	2336.13	RC2	5	4.21	2137.96
	10	13.40	1810.14		10	4.21	1800.48
	15	13.00	1516.53		15	4.04	1276.20
	30	13.00	1489.81		30	4.24	1149.08

5 Conclusions

Although there exists several methods for solving the Vehicle Routing Problem with Time Windows, the majority of them are focused on the static case where all input data are known in advance. This paper proposes a multi-agent approach to solving the Dynamic Vehicle Routing Problem with Time Windows, which allows to observe how different scenario of dispatching dynamic orders to the fleet of vehicles can influence the total cost (and other parameters) of servicing customer orders. Computational experiment showed that centralized dispatching strategy outperforms decentralized one in both dimensions: number of vehicles and the total distance.

Future research may aim at observation and comparison the values of different factors allowing a decision maker to measure performance of the proposed approach. These include: the time a customer must wait before its request is completed, the time spent by each vehicle in the system, the total waiting time of each vehicle, etc.

Another direction of future research is extending the proposed approach to other problems like for example Dynamic Pickup and Delivery Routing Problem with Time Windows (DPDPTW) and incorporating waiting [8] and buffering [10] strategies into the system. The former strategy consists in deciding whether a vehicle should wait after servicing a request, before heading toward the next customer or planning a waiting period on a strategic location. The later one consists in delaying the assignment of some requests to vehicles in a priority buffer, so that more urgent requests can be handled first. Their positive impact on the results has been confirmed by the authors of them.

Acknowledgments. The research has been supported by the Ministry of Science and Higher Education grant no. N N519 576438 (2010-2013).

References

1. Barbucha, D., Jędrzejowicz, P.: Multi-agent platform for solving the dynamic vehicle routing problem. In: Proc. of the 11th IEEE International Conference on Intelligent Transportation Systems (ITSC 2008), pp. 517–522. IEEE Press (2008)
2. Barbucha, D., Jędrzejowicz, P.: Agent-based approach to the dynamic vehicle routing problem. In: Demazeau, Y., Pavón, J., Corchado, J.M., Bajo, J. (eds.) 7th International Conference on PAAMS 2009. AISC, vol. 55, pp. 169–178. Springer, Heidelberg (2009)
3. Bellifemine, F., Caire, G., Greenwood, D.: Developing Multi-Agent Systems with JADE. John Wiley & Sons, Chichester (2007)
4. Braysy, O., Gendreau, M.: Vehicle routing problem with time windows, part i: Route construction and local search algorithms. Transportation Science 39, 104–118 (2005)
5. Braysy, O., Gendreau, M.: Vehicle routing problem with time windows, part ii: Metaheuristics. Transportation Science 39, 119–139 (2005)
6. Davidson, P., Henesey, L., Ramstedt, L., Tornquist, J., Wernstedt, F.: An analysis of agent-based approaches to transport logistics. Transportation Research Part C 13, 255–271 (2005)
7. Ghiani, G., Guerriero, F., Laporte, G., Musmanno, R.: Real-time vehicle routing: Solution concepts, algorithms and parallel computing strategies. European Journal of Operational Research 151, 1–11 (2003)
8. Mitrovic-Minic, S., Laporte, G.: Waiting strategies for the dynamic pickup and delivery problem with time windows. Transportation Research Part B: Methodological 38(7), 635–655 (2004)
9. Pillac, V., Gendreau, M., Guéret, C., Medaglia, A.: A review of dynamic vehicle routing problems. European Journal of Operational Research 225, 1–11 (2013)
10. Pureza, V., Laporte, G.: Waiting and buffering strategies for the dynamic pickup and delivery problem with time windows. INFOR 46(3), 165–175 (2008)
11. Smith, R.: The contract net protocol: High level communication and control in a distributed problem solver. IEEE Transactions on Computers 29(12), 1104–1113 (1980)
12. Solomon, M.: Vrptw benchmark problems, `http://w.cba.neu.edu/~msolomon/problems.htm`
13. Solomon, M.: Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations Research 35, 254–265 (1987)