Agent-Based Approach to the Dynamic Vehicle Routing Problem

Dariusz Barbucha and Piotr Jędrzejowicz

Dept. of Information Systems, Gdynia Maritime University, Morska 83, 81-225 Gdynia, Poland
{barbucha, pj}@am.gdynia.pl

Abstract. The term dynamic transportation problems refers to a wide range of problems where the required information is not given a priori to the decision maker but is revealed concurrently with the decision-making process. Among the most important problems belonging to this group are routing problems, which involve dynamic decision making with respect to vehicle routing in response to the flow of customer demands. The goal of such routing is to provide the required transportation with minimal service cost subject to various constraints. The paper proposes an approach to the dynamic vehicle routing problem based on multi-agent paradigm.

1 Introduction

One of the important group of transportation problems are vehicles routing problems (VRP), where a set of customers is to be served by the fleet of capacited vehicles in order to minimize the service cost and satisfying the set of given constraints. The VRP is said to be *static* if all its input data do not depend explicitly on time, otherwise it is *dynamic*.

The goal of the paper is to present a multi-agent approach to solving the dynamic vehicle routing problem (DVRP). Since DVRP can be viewed as a distributed problem it is proposed to search for solutions using intelligent software agents. To support search process a multi-agent platform simulating activities of the transportation company has been implemented.

The paper is organized as follows. Section 2 includes a formulation of the dynamic vehicle routing problem and introduces measures of the degree of its dynamism. Section 3 describes main features of the multi-agent approach proposed. Section 4 reports on the results of the computational experiment. Finally, Section 5 contains conclusions and suggestions for future research.

2 Formulation of the Dynamic Vehicle Routing Problem

The problem considered in the paper is modelled as an undirected graph $G = (V, E)$, where $V = \{0, 1, \dots, N\}$ is the set of nodes and E is a set of edges. Node 0 is a central depot with NV identical vehicles of capacity W and each other node $i \in V \setminus \{0\}$ represents customer (with its request). Each customer (denoted as $cust(i)$) is characterized by coordinates in Euclidean space $(x(i), y(i))$ and a non-negative demand $d(i)$ $(i = 0 \dots, N)$. Each link (i, j) between two customers denotes the shortest path from customer i to j and is described by the cost $c(i, j)$ of travel from i to j by shortest path $(i, j = 1 \dots, N)$ and $t(i, j)$ $(i, j = 1 \dots, N)$ - the travel time for each edge (i, j). It is assumed that $c(i, j) = c(j, i)$ and $t(i, j) = t(j, i)$.

Let $R = \{R(1), R(2), \ldots, R(NV)\}$ be a partition of V into NV routes of vehicles that cover all customers. Denote a length of the route $R(i)$ by $len(R(i))$ and a cost (or travel distance) by $cost(R(i))$, where $i = 1, \ldots, NV$.

The goal is to find vehicle routes which minimize the total cost of travel - $cost(R) = \sum_{i=1}^{NV} cost(R(i))$ and such that each route starts and ends at the depot, each customer is serviced exactly once by a single vehicle, and the total load on any vehicle associated with a given route does not exceed the vehicle capacity.

In the dynamic version of the problem defined above, it is also assumed that certain number of customers' requests are available in advance and the remaining requests arrive in sequence while the system is already running. Let us assume that the planning horizon starts at time 0 and ends at time T. Let $t(i) \in [0, T]$, where $i = 1, \ldots, N$ denotes the time when the $i - th$ customer request is submitted. Let N_s denotes the number of *static* (i.e. submitted in advance) requests available in $t(i) = 0$, where $i = 1, \ldots, N_s$ and N_d - the number of *dynamic* requests arriving within the $(0, T]$ interval. Of course $N_s + N_d = N$.

There are a few measures of degree of dynamism. In the model considered in the paper it has been used the one given by Lund et al. [8] and Larsen [7] who defined the *degree of dynamism (dod)* as a proportion of the number of dynamic requests to the number of all requests ($dod = N_d/N$). It is easy to see that $dod \in [0, 1]$. According to the formula, the problem is more dynamic, if the above proportion is much closer to 1. If $dod = 0$, then the problem is static, and if $dod = 1$, the problem is fully dynamic.

During the recent years there have been many important advances in the field of static VRP. Because of the fact that this problem is computationally difficult, most of them are based on heuristics or metaheuristics [6]. Definitely much less works have been done with respect to solving dynamic VRP (see for example [4]).

3 Multi-agent Approach for DVRP

Among the methods for solving DVRP, the approaches based on intelligent software agents seems to be promising. In recent years only few approaches based on using intelligent agents for solving some transportation problems have been proposed. Some of them, rather simple, refer direct to the solving one of the problem from VRP class. Such approaches are presented for example in [10] where agent-based architecture is proposed for solving classical VRP, and in [5], where the authors consider Dynamic Pickup and Delivery Problem with Time Windows and propose an agent-based approach to solve it. Much complex multi-agent system developed to simulate planning and scheduling in a shipping company is presented for example in [2]. It solves the dynamic scheduling problem using a set of heterogeneous agents (drivers, trucks, trailers, containers) represented as holonic agents. Each holonic agent (or holon) consists of a set of subagents and co-ordinates and controls the activities of its subagents.

A multi-agent approach for solving DVRP presented in this paper is based on a specially designed multi-agent platform developed to simulate a transportation company activities. The platform is based on JABAT middleware, originally developed for solving difficult combinatorial optimization problems [1].

3.1 Main Features of the JABAT Middleware

JABAT is a middleware supporting design and development of the population-based applications intended to solve different computational problems. It produces solutions to combinatorial optimization problems using a set of intelligent optimising agents, each representing an improvement algorithm. The process of solving of a single task (i.e. a problem instance) in JABAT consists of several steps. At first an initial population of solutions is generated. Then, the individuals from the population are, at the subsequent computation stages, improved by independently acting optimization agents, thus increasing chances for reaching a global optimum. Finally, when the stopping criterion is met, the best solution in the population is taken as the result.

This functionality is realized mainly by two types of agents: *SolutionManagers* and *OptiAgents*. Each *SolutionManager* maintains a population of solutions and is responsible for finding the best solution of a single instance of the problem. *OptiAgents*, each representing a single optimizing algorithm, are used in process of finding/improving the solution of the problem. The agents of both types act in parallel and communicate with each other exchanging solutions that are either to be improved (when solutions are sent to *OptiAgent*) or stored back (when solutions are sent to *SolutionManager*).

Apart from *OptiAgents* and *SolutionManagers* there are also other agents working within the system which are responsible for initialising, organising the process of migrations between different platforms, writing down the results during the process of searching the best solution, and monitoring unexpected behaviors of the system.

More detailed description of JABAT environment were described in [1].

3.2 JABAT Implementation of the DVRP Simulator

The proposed implementation is based on the assumptions that population of solutions consist of a single individual, and only one optimization agent is responsible for improving a solution.

The structure of the *OptiAgent* designed for solving DVRP, viewed as a part of JABAT environment, is presented in Fig. 1. The specialized *OptiAgent*, called *DVRP-OptiAgent*, which reflects a transport company and typical elements of it, is itself a set of the following types of agents:

- *ACompany (AC)* - an agent which runs first and initializes all others agents.
- *ARequestGenerator (ARG)* - an agent which generates (or reads) new orders and sends them to the *ARequestManager* agent.
- *ARequestManager (ARM)* - an agent which manages the list of requests received from *ARG*. After receiving the new request, *ARM* announces it to each *AVehicle* agent and chooses the best from offers returned by *AVehicle* agents.
- *AVehicle (AV)* - an agent that represents a vehicle and is described by the capacity of the vehicle, actual route assigned to this vehicle, actual cost of the route, actual available space and the vehicle state. Periodically *AV* receives customer's request from the *ARM* one at a time, tries to assign it to the existing route in order to minimize the cost, and sends back its offer (i.e. calculated cost of insertion) to the *ARM*. If the offer turns out to be the best, the respective request is added to the actual route. Most

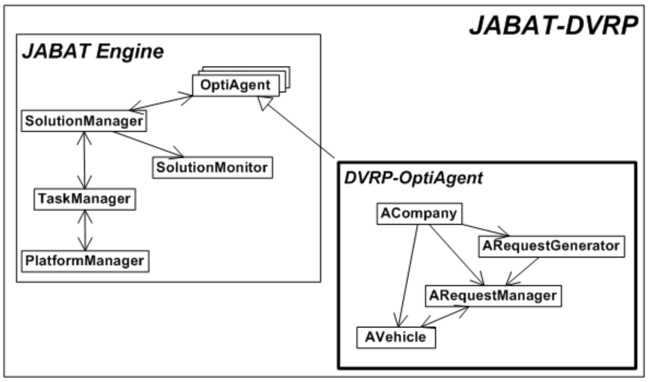

Fig. 1. Structure of the DVRP implementation based on JABAT

of its lifetime, a vehicle spends serving requests. It starts after receiving and accepting the first request. After reaching the nearest customer it goes to the next customer belonging to the route. In the model considered in the paper it is assumed that if the vehicle reaches the last customer on the current route, it waits in this location until a new arriving request is assigned to it or until the end of pool of requests is reached. In the first case the vehicle breaks waiting and moves to the new assigned customer, in the second case the waiting vehicle returns back to the depot.

Although the development of software agents representing elements of such company is not a new idea (similar approach is presented for example in [5], where three types of agents are proposed: agent-customer, agent-company and agent-vehicle), it seems to be very natural, taking into account a distributed nature of the problem.

The process of solving DVRP is divided into three phases:

1. Allocation of the pool of static requests to the available vehicles,
2. Allocation of the new dynamic requests to the fleet of vehicles,
3. Improvement of the current solution by *intra-route* and *inter-routes* operations.

The **first phase** is in fact the process of solving the static VRP for N_s requests. The solution obtained by procedure used in the system and described below gives the initial solution to the DVRP.

The initial solution is generated basing on polar representation of each vertex (customer with its request) in graph G, which uses the idea originated from *split* phase of the *sweep* algorithm of Gillett and Miller [3]. First, each vertex (customer) $i \in V$ is transformed from cartesian coordinates to polar coordinates $(\theta_i; \rho_i)$, where θ_i is the angle and ρ_i is the ray length.

Generation of individual (solution) starts from randomly choosing an arbitrary vertex i^* and assigning a value θ_{i^*} to it. Next, the remaining angles centered at 0 from the initial ray $(0; \rho_{i^*})$ are computed and the vertices (customers) are ranked in increasing order of their θ_i value. Resulting ranking determines the order in which requests are assigned to the available vehicles.

The process of assignment vertices to vehicles starts from the first unrouted vertex having the smallest angle, and next assigning vertices to the first vehicle as long as its capacity is not exceeded. If the capacity of the vehicle is exhausted, the vertices are assigned to the next vehicle. The whole process is repeated until the end of pool of static requests is reached.

Assignment of requests to vehicles is carried out in the form of messages exchange between *ARequestManager* and *AVehicle* agents.

After assigning all static requests to the available vehicles, all vehicles with the requests assigned to them start moving, and in loop the system is waiting for an event. Taking into account the objective of the problem, the most important event is a new request event.

The **second phase** includes assigning new dynamic requests to available vehicles and is realized in dynamically changing environment, where all vehicles are serving the customers already assigned to their routes.

The main steps of this phase are:

1. *ARG* reads (or generates) a new dynamic request and sends it to the *ARM*.
2. After receiving a new request from the *ARG*, *ARM* initializes a session using the *Contract Net Protocol (CNP)* [9] and starts communication between *ARM* and *AV* agents. As *Initiator* it announces the request to each *AV* agent (moving and waiting vehicles) sending around a call for proposal (`cfp`) message. *AV* (as *Participants* or *Contractors*) are viewed as potential contractors.
3. Each *AV* agent after receiving the request (with customer data) from the *ARM*, calculates the cost of inserting a new customer into the existing route. If an insertion of a new customer into the existing route does not violate the vehicle's capacity, the calculated cost of insertion is sent back (as `propose` message) to the *ARM*. Otherwise, the *AV* sends back the rejection (`reject`) message.
4. *ARM* after receiving proposals from all *AV* agents, chooses the one with the lowest cost of insertion. Next, it sends the `accept-proposal` message to the *AV* which is awarded and the `reject-proposal` to the others.
5. *AV* which receives the `accept-proposal` message, inserts the customer into the current route and sends the `inform-done` message if the operation is performed successfully and `failure` message, otherwise.
6. The above process is repeated for each new request.

Additionally, two kinds of improvement procedures are defined for *ARM* and *AV* agents and performed in the **third phase** of the process of solving DVRP. Each *AV* agent executes a set of operations that aim at improving the cost of its route (*intra-route* operations which operate on one selected route). In addition, the *ARM* agent also periodically performs global moves that aim at improving the global solution (*inter-routes* operations which operate on at least two selected routes).

Three *intra-route* operations include:

- *v1_2opt* - The implementation of *2-opt* algorithm where the sequence of customers visited by the vehicle on route $R(p)$ ($p \in \{1,\dots,NV\}$) is changed by eliminating two edges and reconnecting the two resulting paths in a different way to obtain a new route.
- *v1_relocate* - The sequence of customers visited by the vehicle on route $R(p)$ is changed by moving one customer $cust(i) \in R(p)$, $i = 1,\dots,len(R(p))$, $p \in \{1,\dots,NV\}$ from its current position to another one.

- *v1_exchange* - Two selected customers from current route $R(p)$ are swapped, i.e. for each pair of customers $cust(i), cust(j) \in R(p)$ $(i \neq j,\ i,j = 1,\ldots,len(R(p))$, $p \in \{1,\ldots,NV\})$, $cust(i)$ moves to position occupied by $cust(j)$ and $cust(j)$ moves to the position occupied by $cust(i)$.

All possible moves are considered in above operations and moves that shorten the current route are accepted. The resulting route with the greatest reduction of the total cost is accepted as a new tour of the vehicle.

The above *intra-route* operations could be initialized by the *ARM* agent and next performed by *AV* agent or directly performed by *AV* agent. In the first case, *ARM* decides whether and when use the operation and sends the proper message to the particular *AV*. In the second case, *AV* autonomically decides about performing the operation.

Two *inter-routes* operations are proposed and implemented in the system:

- *v2_relocate* - One selected customer $cust(i)$ from one route $R(p)$ is moved to the second route $R(q)$ $(i = 1,\ldots,len(R(p)),\ p,q \in \{1,\ldots,NV\})$,
- *v2_exchange* - Two selected customers from two different routes $(cust(i) \in R(p)$ and $cust(j) \in R(q))$ are selected and swapped $(i = 1,\ldots,len(R(p)),\ j = 1,\ldots,len(R(q))$, $p,q \in \{1,\ldots,NV\})$.

As it is easy to see, during the process of assigning each new request to the available vehicles, each *AV* agent competes with others in order to get the request to servicing but during the execution improvement operations, agents cooperate in order to improve the current solution.

To sum up, the whole algorithm based on multiple agents for solving DVRP is presented in form of pseudocode as follows.

```
1. Allocate static requests
2. All vehicles with the assigned requests start moving
3. In loop system is waiting for an event
  IF (request_event)
    CASE "new request":
      allocate request to available vehicles
    CASE "end of requests":
      waiting vehicles return to the depot
  ELSE IF (vehicle_event)
    CASE "vehicle $v(i)$ reached the location $p$":
      IF (NOT location $p$ is the last location)
        vehicle $v(i)$ proceeds to the next location
      ELSE
        IF (NOT end_of_requests)
          vehicle $v(i)$ waits in location p
        ELSE
          vehicle $v(i)$ is moving to the depot
  ELSE
    between events do intra-route and inter-routes operations
4. If all vehicles returned to the depot then STOP.
```

4 Computational Experiment

To validate effectiveness of the approach computational experiment has been carried out. For static cases, solutions (global cost of serving all request) obtained by the proposed approach were compared with the best known solutions using the mean relative error (MRE) from the best known solution. Additionally, for dynamic cases, the influence of the degree of dynamism of the problem and frequency of the customer requests arrivals on the solution have been observed.

The proposed agent-based approach was tested on classical VRP dataset transformed into its dynamic version, in which not all requests are known in advance. The experiment involved 5 benchmark instances (*vrnpc1 - vrnpc5*) available from *OR-Library* benchmark set [11]. Each benchmark set includes information about number of customers, capacity of vehicles, coordinates of depot and coordinates and demands of customers. The selected problems contain 50-199 customers located randomly over the plane and have only capacity restriction.

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 our experiment).

The proposed simulation model was run for the number of dynamic requests (N_d) varying from 0% (pure static problem) to 100% of all requests with step equal 20%. Additionally, for each positive value of the degree of dynamism, it has been assumed that dynamic requests may arrive with various frequencies. For the purpose of experiment λ was set to 3, 4, 5, 6, 10, 15, 20.

Additionally, it has been assumed that the vehicle speed was set at 60 km/h.

The above assumptions produced 36 test instances (1 static and 35 dynamic) for each dataset, giving in total 180 test instances. Moreover, each test problem was repeatedly executed five times and mean results from these runs were recorded.

The experiment results are presented in Tables 1-2.

Table 1 shows mean relative errors averages over all runs for each tested static instance of VRP from the OR-Library. Together with the problem names the header of the table includes the number of customers in the brackets. In rows of the table the average (Avg), minimum (Min) and maximum (Max) values of errors for each instance are shown.

Table 2 shows values of the percentage increase in cost of allocating all dynamic requests for selected instances of the problem as compared to the cost of the best known solution to the static instance. The first two columns of the table include degree of dynamism (in %) and mean number of requests per hour. The remaining five columns

Table 1. Mean relative error from the best known solution for selected instances of DVRP with all static requests

	vrpnc1 (50)	vrpnc2 (75)	vrpnc3 (100)	vrpnc4 (150)	vrpnc5 (199)
Avg	0,53%	3,83%	3,86%	5,07%	4,63%
Min	0,00%	2,25%	1,63%	2,82%	3,91%
Max	2,20%	7,91%	5,77%	8,09%	5,78%

Table 2. The performance of the proposed agent-based approach (measured as dynamic/best known static cost) for selected instances of the DVRP

Degree of dynamism	Mean number of requests per hour	vrpnc1 (50)	vrpnc2 (75)	vrpnc3 (100)	vrpnc4 (150)	vrpnc5 (199)
	3	34,8%	12,3%	17,8%	22,4%	14,3%
	4	36,4%	12,1%	17,8%	21,3%	14,7%
	5	14,1%	11,2%	10,7%	12,8%	14,8%
20%	6	15,3%	10,4%	10,6%	10,1%	9,2%
	10	12,7%	9,5%	9,3%	9,9%	8,8%
	15	5,1%	9,4%	7,9%	3,9%	8,6%
	20	2,2%	7,9%	4,5%	2,6%	7,7%
	3	45,2%	37,0%	52,4%	53,7%	39,2%
	4	52,9%	24,4%	30,8%	45,6%	35,8%
	5	53,5%	22,3%	29,8%	39,8%	27,2%
40%	6	48,8%	20,9%	27,4%	34,2%	20,6%
	10	10,9%	17,6%	23,7%	30,7%	18,7%
	15	3,2%	15,9%	16,7%	18,2%	15,8%
	20	2,8%	14,3%	10,6%	16,3%	8,4%
	3	80,6%	61,7%	58,6%	90,5%	78,1%
	4	41,7%	45,7%	54,7%	82,1%	56,2%
	5	55,9%	27,8%	40,8%	80,4%	45,8%
60%	6	29,1%	34,6%	38,9%	73,2%	44,8%
	10	36%	37,9%	46,8%	48,7%	38,7%
	15	47,5%	33,4%	34,1%	47,7%	38,7%
	20	18,5%	25,2%	23,9%	44,2%	30,6%
	3	105,6%	84,7%	71,8%	111,1%	117,3%
	4	83,7%	40,1%	59,3%	88,6%	103,1%
	5	60%	42,7%	54,3%	68,0%	78,4%
80%	6	65,3%	44,6%	50,1%	66,9%	70,6%
	10	24,4%	39,7%	46,6%	65,7%	63,6%
	15	41,3%	40,1%	42,8%	53,1%	62,3%
	20	13,4%	38,1%	33,5%	51,4%	43,6%
	3	101,6%	98,5%	80,9%	128,7%	121,7%
	4	77,4%	89,5%	80,2%	94,8%	113,2%
	5	82,4%	65,6%	77,5%	82,2%	99,8%
100%	6	73,6%	68,8%	77,1%	73,2%	96,2%
	10	38,5%	60,9%	72,4%	56,1%	75,3%
	15	23,9%	60,7%	46,4%	53,6%	75,2%
	20	22,4%	51,0%	49,8%	50,4%	56,5%

show calculated values obtained by proposed agent-based approach for each tested instance of the problem.

Results obtained during the experiment and presented in Table 1 show that the proposed agent-based approach produces quite good solutions in case of static requests only. The average value of MRE is not greater than 5%, but it depends on the instance and for most instances is smaller. Minimal value of MRE observed during the experiment is equal to 0% for *vrpnc1* instance or close to 2-3% for most of the instances, which is only slightly worse than the results produced by other methods (see, for example [6]).

By analyzing the results presented in Table 2 it is easy to see, that overall cost depends strongly on the degree of dynamism and frequency of dynamic request arrivals for all tested instances. In most cases the total cost may be substantially higher than for the static case.

It should be noted that comparisons of obtained results to dynamic cases can not be directly compared with the approaches proposed by other authors mentioned in this paper. This is mainly due to differences in problem formulation and differences in datasets used for evaluating the results. However, in case of static instances the results are fully comparable.

5 Conclusions

The paper proposes a multi-agent approach to solving the dynamic vehicle routing problem. The approach is based on a multi-agent platform which can be used to simulate activities of the transport company and analyze various scenarios with respect to the dynamic routing of the fleet of vehicles.

Computational experiment proved that presented approach can offer good quality solutions for the static case (as compared with state of the art approaches). It also shows how dynamic nature of the problem can influence the total cost of realizing customer requests. The overall evaluation of the presented approach is positive thanks to several features typical for multiple agent systems, like autonomy of agents, ability to increase computational efficiency through parallelization and possibility of using distributed environment.

References

1. Barbucha, D., Czarnowski, I., Jędrzejowicz, P., Ratajczak, E., Wierzbowska, I.: An Implementation of the JADE-based A-Team Environment. International Transactions on Systems Science and Applications 3(4), 319–328 (2008)
2. Burckert, H.J., Fischer, K., Vierke, G.: Holonic Transport Scheduling with TeleTruck. Journal of Applied Artificial Intelligence 14, 697–725 (2000)
3. Gillett, B.E., Miller, L.R.: A heuristic algorithm for the vehicle dispatch problem. Operations Research 22, 240–349 (1974)
4. van Hentenryck, P.: Online Stochastic Combinatorial Optimization. MIT Press, Cambridge (2006)
5. Kozlak, J., Creput, J.C., Hilaire, V., Koukam, A.: Multi-agent environment for dynamic transport planning and scheduling. In: Bubak, M., van Albada, G.D., Sloot, P.M.A., Dongarra, J. (eds.) ICCS 2004. LNCS, vol. 3038, pp. 638–645. Springer, Heidelberg (2004)
6. Laporte, G., Gendreau, M., Potvin, J., Semet, F.: Classical and modern heuristics for the vehicle routing problem. International Transactions in Operational Research 7, 285–300 (2000)
7. Larsen, A.: The on-line vehicle routing problem. Ph.D. Thesis, Institute of Mathematical Modelling, Technical University of Denmark (2001)

8. Lund, K., Madsen, O.B.G., Rygaard, J.M.: Vehicle routing problems with varying degrees of dynamism. Technical report, Institute of Mathematical Modelling, Technical University of Denmark (1996)
9. Smith, R.G.: The Contract Net Protocol: High Level Communication and Control in a Distributed Problem Solver. IEEE Transactions on Computers 29(12), 1104–1113 (1980)
10. Thangiah, S.R., Shmygelska, O., Mennell, W.: An agent architecture for vehicle routing problem. In: Proc. of the ACM Symposium on Applied Computing (SAC 2001), Las Vegas, pp. 517–521 (2001)
11. OR-Library, `http://people.brunel.ac.uk/~mastjjb/jeb/orlib/vrpinfo.html`