2012 IEEE International Conference on Systems, Man, and Cybernetics
October 14-17, 2012, COEX, Seoul, Korea

Parallel Solver for Vehicle Routing and Pickup and Delivery Problems with Time Windows Based on Agent Negotiation

Petr Kalina
Dept. of Cybernetics
Czech Technical University in Prague
petrkali@gmail.com

Jiří Vokřínek
Agent Technology Center
Czech Technical University in Prague
jiri.vokrinek@fel.cvut.cz

***Abstract*—In this paper we present a parallel solver for the vehicle routing problem with time windows (VRPTW) and the pickup and delivery problem with time windows (PDPTW). The solver is based on parallel competition of particular solvers solving the given problem instance. The particular solvers are based on negotiation between a fleet of agents representing individual vehicles using a cost structure corresponding to the well-known travel time savings insertion heuristic. The performance of the solver is assessed on the Homberger-Gehring and Li-Lim benchmark sets for the VRPTW and PDPTW case respectively. While both sets are widely used across the routing community, they were not addressed by previous agent-based studies. Thus the achieved average solution quality of 95% and 87% for the VRPTW and PDPTW cases represents a new best known result for these benchmark sets for agent-based approaches. An analysis of the solver's convergence, runtime and parameter sensitivity is provided within the experimental evaluation section, that has also not been provided by previous agent-based studies.**

***Index Terms*—VRPTW, PDPTW, Agents, Parallelization**

I. Introduction

The family of Vehicle Routing Problems (VRP) has been studied well over 50 years with the original Dantzig [1] article dating back to 1959. The time and capacity constrained variants of the problem — the vehicle routing problem with time windows (VRPTW) and the closely related pickup and delivery problem with time windows (PDPTW) — are amongst the most widely studied problems in the area of operations research. Both problems are very similar and consist of constructing a set of routes starting and ending at a single depot that satisfy the demands of all the customers while satisfying the time window and vehicle capacity constraints. The exact problem semantics will be described later within the Section III.

Real world applications of routing algorithms are typically complex, with dynamic, heterogenous and potentially non-cooperative environments being captured and processed. The multi-agent systems are an emerging choice for modeling systems with similar attributes [2].

Within this paper we present a parallel solver (Section IV) that is based on competition of particular solvers being run in parallel. The individual competing solvers are based on the negotiation between a fleet of agents corresponding to the individual vehicles being routed. A rich solver configuration space is presented including several specific negotiation methods used to improve the particular solution under construction. Thus a relatively big number of solvers can be spawned returning an improving sequence of results over time.

A comprehensive analysis of the solver's performance and algorithmic features is provided (Section V), based on the widely used benchmark sets of Homberger-Gehring [3] (VRPTW) and Li-Lim [4] (PDPTW). Such an information has been missing from previous agent based studies, either because the presented algorithms couldn't cope with the complexity of the extended datasets or because they were concerned mainly with the real-world applicability using ad-hoc problem instances.

II. Related Work

An excellent survey of VRPTW methods is provided by [5], [6], while a comprehensive survey on the PDPTW is presented by [7]. Both surveys concern the classical centralized approaches. Also, the list of references to the currently leading state-of-the-art solvers is well maintained at [8] for both benchmark sets.

A number of agent-based approaches have been suggested for solving the routing problems in general. On simple VRP good results have been reported by an agent based solver presented by [15].

An agent based algorithm for VRPTW is presented in [9], built around the concepts of a Shipping Company and underlying Shipping Company Truck. The planning is done dynamically and is based on the well known contract net protocol (CNP) accompanied by a "simulated trading" improvement strategy based on finding the optimal exchanges of customers between the trucks by solving a maximal pairing problem on a graph representing the proposed exchanges. Unfortunately, no relevant performance assessment is provided.

The algorithm for PDPTW presented by [10] is essentially a parallel insertion procedure based on CNP with subsequent improvement phase consisting of reallocating some randomly chosen tasks from each route. Used cost structure is based on the well known Solomon's I1 insertion heuristic [11]. The performance is assessed on an ad-hoc dataset.

978-1-4673-1714-6/12/$31.00 ©2012 IEEE

The algorithm for VRPTW presented by [12] is based on agents representing individual customers, individual routes and a central planner agent. A sequential insertion procedure based on Solomon's I1 heuristic is followed by an improvement phase in which the agents propose moves gathered in a "move pool" with the most advantageous move being selected and performed. Additionally, a route elimination routine is periodically invoked — which is not well described in the text. Experimental assessment is based on Solomon's instances [11] and does not include the extended datasets. Also, no runtime information is provided.

III. Problem Statement

The VRPTW is defined by a depot 0, vehicle capacity c and a set of vehicle routing (VR) tasks $vr_1 \dots vr_n$ to be served. Each task vr_i is defined by a customer i, demand d_i, service time s_i and service time window (e_i, l_i). The mutual travel times denoted as $t_{i,j}, i = 0..n, j = 0..n$ between the individual customers as well as the depot are known and correspond to Euclidian space.

A feasible solution corresponds to a set of m routes for a fleet of m vehicles each starting and finishing in the depot that together serve all VR tasks within the given time windows (time windows constraints). The sum of all customer demands for a single route must not exceed the vehicle capacity c (capacity constraint).

The PDPTW is an extension of the VRPTW given by a depot 0, vehicle capacity c and a set of pickup-delivery (PD) tasks $pd_1 \dots pd_k$ to be served. Each PD task $pd_i = (vr_p, vr_d)$ consists of two VR subtasks defined according to the previous problem definition, with $d_p = -d_d$. Both vr_p and vr_d have to be served within a single route, the pickup vr_p preceding the delivery vr_d (pairing constraints). The capacity constraint is modified so that at any point on the route the actual load of the vehicle must not exceed the vehicle capacity c.

For both problems, the primary optimization criteria is to find the minimal number of routes serving all customers (*vehicles count*, VC). The secondary hierarchical objective is the minimization of the cumulative duration of all routes combined (*travel time*, TT). Thus both problems consist of solving the underlying multiple traveling salesman problem in the search space constrained by the capacity, time window and the pairing constraints (PDPTW).

For both cases a lower bound on the number of routes can be computed based on the mutual incompatibilities of the tasks. Two tasks are incompatible if they cannot be served by a single vehicle, that is when it is impossible to start the service at either of the customers at the earliest possible moment and reach the other customer prior to the end of the corresponding time window. The minimal number of vehicles necessary for solving a given problem instance is bound by the size of the maximal set of mutually incompatible tasks (TWLB). Using a graph with edges corresponding to the mutually incompatible tasks, this corresponds to solving a maximal clique problem — a NP-hard problem in it's own right. Therefore, within this work we use a greedy max-clique heuristic based on [13] with a complexity of $O(n^3)$, n being the number of tasks.

Also, for the VRPTW, the cumulative demand of all customers has to be lower than the cumulative capacity of all the vehicles combined providing for a capacity based lower bound on the number of routes (CLB). As the upper bound on the number of routes is the number of tasks in the problem instance for both problems, we define the *upper bound number of routes* (UB) as

$$UB = n \tag{1}$$

and the *lower bound number of routes* (LB) as

$$LB = max(CLB, TWLB) \tag{2}$$

where n is the number of tasks and CLB and TWLB are the capacity based and task incompatibility based lower bounds defined above, given that $CLB = 1$ for the PDPTW.

IV. Parallel Multi-Agent Solver

As mentioned above, the presented solver is based on parallel competition of solvers based on agent negotiation in an approach similar to [14].

A. Agent Based Negotiation Mechanism

The individual competing particular solvers are based on an agent negotiation based algorithm. The basic algorithm envelope is generic [15] and provides a base for formalizing agent negotiation based solving approaches to routing problems in general.

The algorithm is based on a three layer agent architecture featuring a top layer represented by a Task Agent, middle layer represented by an Allocation Agent and a fleet of Vehicle Agents present at the bottom level of the architecture.

- **Task Agent** acts as an interface between the algorithm's computational core and the surrounding infrastructure. It is responsible for registering the tasks and submitting them to the underlying Allocation Agent.
- **Allocation Agent** instruments the actual solving process by negotiating with the Vehicle Agents. The negotiation is conducted based upon task commitment and decommitment cost estimates provided by the Vehicle Agents.
- **Vehicle Agent** represents an individual vehicle serving a route. It provides the Allocation Agent with the above mentioned inputs. These are computed based on local Vehicle Agent's plan processing.

The negotiation algorithm is outlined by Figure 1. The algorithm is initialized by applying a particular ordering to the set of tasks by the Task Agent (line 1). A fleet of Vehicle Agents is instantiated with a size corresponding to the upper/lower bound for the number of routes, according to Equations 1 and 2 (line 2).

Follows the main allocation loop (lines 3 – 7). The Vehicle Agent with the lowest commitment cost estimate is identified in an auction process conducted on behalf of the Allocation Agent, based on the locally computed commitment cost estimates provided by the Vehicle Agents. For both presented

Procedure $negotiate(problemInstance)$
begin
1: apply *ordering* to the set of instance tasks T;
2: instantiate *initial fleet* of Vehicle Agents;
3: **for each** task $t \in T$
4: find Vehicle Agent v with the lowest commitment cost estimate;
5: let agent v *commit* to the task t;
6: apply *iterative improvement* step;
7: **end for each**
8: apply *final improvement* step;
end

Fig. 1. The negotiation based allocation algorithm.

problems the commitment cost estimates are based on the implementation of known travel time savings insertion heuristic. In a distributed environment the auction process is carried out using the well known CNP protocol. The agent with the lowest commitment cost commits to the task (line 5).

Lines 6 and 8 correspond to an improvement step being applied e.g. *ReallocateAll* or *ReallocateWorst* described later in the text. An improvement step corresponds to a negotiation being conducted between the Vehicle Agents in which the partial solution being constructed is improved by a series of task *reallocations*, in which some agents drop some of their commitments while some other agents with better cost estimates commit to the corresponding tasks. Based on the particular solver configuration, the improvement steps can be executed *iteratively* (line 6) or only at the end of the allocation process (line 8).

In case no Vehicle Agent can feasibly commit to the task on line 4 (due to the time window or capacity constraints), the process is terminated and restarted with an increased initial Vehicle Agents fleet size. Obviously this happens only with the LB initial fleet size setting, as with the UB setting a baseline solution with each task being served by a single vehicle is always feasible.

B. Particular Solver Configuration Space

As mentioned above, the presented parallel solver is based on a wide competition of particular solvers being spawned and run in parallel. The solvers are based on the general negotiation algorithm defined above. Thus a particular solver is instantiated based on (i) the *initial task ordering*, (ii) the *initial fleet size*, (iii) the *iterative improvement method* and (iv) the *final improvement method* to be used.

Following initial task orderings were considered:

- **First In First Out** (FIFO): tasks processed in the order they enter the system (from the original set).
- **Most Demand First** (MDF): tasks are ordered decreasingly by the volume of their demands.
- **Least Demand First** (LDF): tasks are ordered in a reverse order than in case of the MDF ordering.
- **Tightest Time window First** (TTF): tasks are ordered increasingly by the duration of their time window (VRPTW) or by the sum of the durations of the time windows of their pickup and delivery subtasks (PDPTW).
- **Widest Time window First** (WTF): tasks are ordered in a reverse order than in case of the TTF ordering.
- **Earliest First** (EF): tasks are ordered increasingly by the beginning time of their time window (VRPTW only).
- **Latest First** (LF): tasks are ordered in a reverse order than in case of the EF ordering (VRPTW only).
- **Earliest First Pickup** (EFP): tasks are ordered increasingly by the beginning time of the time window of their pickup subtask (PDPTW only).
- **Earliest First Delivery** (EFD): tasks are ordered increasingly by the beginning time of the time window of their delivery subtask (PDPTW only).
- **Latest First Pickup** (LFP): tasks are ordered in a reverse order than in case of the EFP ordering (PDPTW only).
- **Latest First Delivery** (LFD): tasks are ordered in a reverse order than in case of the EFD ordering (PDPTW only).

Note that the MDF and TTF orderings correspond to the well known *most constrained first* greedy heuristic for the underlying multiple bin packing problem and the time windows constraints respectively. The EF and LF orderings and their PDPTW variants are motivated by having the competing tasks from the time windows point of view being allocated in close succession potentially preventing future bottlenecks.

With respect to the initial fleet size, the two previously mentioned settings were considered — the LB and the UB settings corresponding to Equations 1 and 2.

We considered the three following improvement step negotiation methods:

- ***DelegateWorst*** (DLGW): each Vehicle Agent identifies it's worst task based on the locally computed decommitment gain estimate and delegates it to a *different* agent with a better commitment cost estimate. In case no such agent exists, the original commitment is kept.
- ***DelegateAll*** (DLGA): each Vehicle Agent processes all of it's tasks in the order in which they appear within it's route. For each processed task it delegates it to a *different* agent with a better commitment cost estimate. In case no such agent exists, the original commitment is kept.
- ***ReallocateAll*** (RLCA): each Vehicle Agent processes all of it's tasks in the order in which they appear within it's route and reallocates each task to an agent with a better commitment cost estimate. The difference from the *Delegate* methods is, that the task can be reallocated also to a different place in the current agent's route.

For the sake of simplicity we considered particular solvers with (i) only the final improvement step being employed (denoted as DLGW, DLGA and RLCA particular solvers) and (ii) solvers with an identical negotiation method used for the iterative and the final improvement steps (denoted as DLGW-I, DLGA-I and RLCA-I solvers). A baseline solver not employing either of the improvement steps is denoted as CNP.

Thus the configuration space for the particular solvers

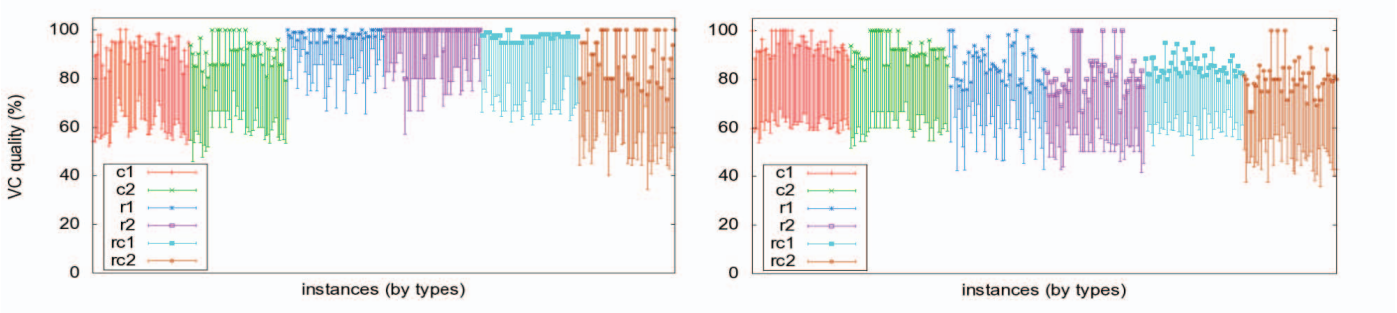

Fig. 2. Worst to best quality across the particular solver competition for all the VRPTW (left) and the PDPTW (right) instances

consists of 98 configurations in the VRPTW case (7 orderings $\times$ 2 initial fleet sizes $\times$ 7 improvement method configurations) and 126 configurations in the PDPTW case (2 more orderings).

C. Vehicle Agent Cost Estimates

In previous sections we mentioned several times, that the negotiation core of the presented solver is based on the commitment cost estimates. Also, in case of the improvement step methods, a decommitment cost estimate is used. These estimates are computed locally by the Vehicle Agents and thus can be modified to reflect the semantics of a particular real-world problem being solved.

For the presented solver these estimates are computed based on the well known travel time savings (TTS) insertion heuristic. The heuristic — including the time windows and capacity constraint checking — is efficiently implemented according to [16]. For the PDPTW, the heuristic was further modified to find the cheapest composite insertion of both the pickup and delivery tasks.

V. Experimental Evaluation

Within the presented study, we evaluate the solver on the PDPTW and VRPTW instances presented by [3] and [4] available at SINTEF TOP Portal [8]. For each of the problems there are 300 instances with sizes of 200, 400, 600, 800 and 1000 tasks. For each size there are 6 instance types provided denoted as R1, R2, RC1, RC2, C1, and C2 type. For C1 and C2 types the customer locations are grouped in clusters, unlike the R1 and R2 classes where the customers are randomly placed. The RC1 and RC2 types provide a mix of both random and clustered customer locations. The C1, R1 and RC1 also differ from C2, R2 and RC2 in terms of the scheduling horizon, the former having a shorter horizon resulting in routes of about 10 customers on the average, the latter having a longer horizon with routes of around 50 customers.

The solver is implemented in JAVA programming language. All experiments were performed using a 4G RAM AMD Athlon 2G Gentoo system running the 64-bit Sun JRE 1.6.0_22.

The results are presented in percentage relative quality metric for both the primary vehicles count (VC) criteria and the secondary travel time (TT) criteria computed based on the best known results available at the SINTEF portal as:

$$quality = \frac{best\ known\ value}{measured\ value} \times 100\ [\%] \qquad (3)$$

A. Solution Quality

The achieved quality for the primary VC optimization criteria for both problems is illustrated by Figure 2. The graphs show bars corresponding to the worst and the best achieved result for individual instances for the respective problems. The results are grouped by individual instance types.

For the VRPTW case the solver achieved an average VC quality of 94.5% with a 100% quality being achieved for 30.3% of the instances. For these instances, the average achieved TT quality was 72%, while the overall TT quality over all instances was 86%. In 3.3% of the cases the solver was able to find solutions with shorter routes than the best known, on the expense of lower VC quality. The solver achieved particularly good results on the randomized R1 and R2 instance types with a 98.8% VC quality and 63% of the solutions equalling the best known solutions.

For the PDPTW case the average VC quality was 86.7% compared to the best known solutions, with a 100% quality being achieved for 11.7% of the instances with an average TT quality of 90.5%. Interestingly, given the weaker performance compared to the VRPTW case, in 3% of the cases the solver was able to equal the best known solutions in both criteria. In 15.1% of the cases the solver was able to find solutions with shorter routes than the best known, again on the expense of lower VC quality.

Given the fact that the solver (i) is computationally bounded and (ii) traverses only feasible solution space, the results arguably prove that the agent negotiation is a sound algorithmic approach to the time-constrained routing problems variants, the weaker solution quality being offset by the inherent flexibility of the multi-agent approach.

We argue that there are several opportunities to further improve the solution quality. As is discussed later within the Section V-D, an improvement could be achieved by modifying the used insertion heuristic to exploit the time window constraints in an approach similar to [13]. Also, a further improvement could be achieved by (i) enabling more complex moves within the allocation process than the simple single task reallocations and (ii) enabling traversing also the unfeasible

Particular Solvers	VRPTW	PDPTW
CNP	73.3% (0%)	67.1% (0%)
DLGW	78.8% (1.3%)	71.2% (0%)
DLGW-I	81.2% (4.6%)	75.3% (3.5%)
DLGA	85.0% (7.5%)	76.2% (0.9%)
DLGA-I	86.4% (31.0%)	79.5% (28.2%)
RLCA	84.7% (8.8%)	76.7% (1.6%)
RLCA-I	87.3 (46.7%)	81.0% (65.8%)

TABLE I
NEGOTIATION METHODS COMPARISON

solution space by employing some backtracking strategy in a fashion similar to the capacity backtracking presented by [14].

B. Negotiation Methods

We further analyzed the average achieved quality based upon the particular negotiation method used for the improvement steps. Table I summarizes the results, listing the average achieved quality and the percent of wins in the solver competition (in brackets) aggregated over all particular solvers using the corresponding negotiation method.

The results show a correlation between the complexity and the achieved average quality of the individual configurations, with the most complex RLCA-I configuration providing the best results for both problems. Thus the results suggest a significant positive effect of the negotiation methods on the overall convergence, in particular when applied iteratively throughout the allocation process.

C. Initial Fleet Size

Within this experiment we evaluated the influence of the initial fleet size setting on the convergence of the solver. Table II summarizes the results for the two problems together with a baseline VRP experiment, consisting of running the whole VRPTW experiment while ignoring the time window constraints. The results correspond to the percent of solver configurations where (i) the LB setting outperformed the corresponding UB setting in the VC criteria, (ii) the UB setting outperformed the LB setting in the TT criteria and (iii) both settings achieved identical results.

The results illustrate the impact the time window constraints have on the solving process. Given a route and a task to be inserted, in the baseline VRP case all possible insertion positions are feasible, provided that the capacity constraint is not violated. In the constrained cases the number of feasible insertions is limited to positions with fitting vehicle arrival time, given that the insertion will not cause a delay in the schedule rendering any of the subsequent customer visits unfeasible. With a limited number of vehicles for the LB setting and thus correspondingly limited number of feasible insertion positions the solver is forced to perform insertions that cause big travel time increases. The UB setting avoids such insertions by using more routes to cover the customers and thus the UB and the LB solutions diverge.

Thus the allocation process based on the TTS heuristic fails to successfully address the primary optimization criteria in the UB setting. We argue, that the only way how to achieve

Problem	LB Better in VC	UB Better in TT	Identical
VRPTW	50.3%	37.1%	49.0%
PDPTW	57.3%	32.4%	40.9%
VRP	8.9%	6%	90.9%

TABLE II
INITIAL FLEET SIZE SETTINGS COMPARISON

UB setting convergence is by devising a route elimination mechanism based on a completely different cost structure taking into account both the spatial (travel time) and the temporal (time windows) aspects of the problem.

D. Initial Task Orderings

Within this experiment we aimed at assessing the influence of the initial task ordering on the success of the subsequent allocation process. Such an assessment is relevant namely because (i) it illustrates the baseline applicability of the solver for the dynamic problem variants and (ii) because a minimal influence was reported by previous agent-based studies on the simple VRP case [14].

Table III lists the quality achieved by the individual orderings corresponding to the best achieved results for a competition of solvers using only the specified ordering. The results prove that the success of the allocation process is sensitive to the initial task ordering. Interestingly, the best results were achieved by the orderings based on the temporal aspects of the problems, rather than the orderings based on the capacity aspects. Arguably this is due to the fact that for the presented benchmark sets, the temporal aspects of the problems induced by the time windows constraints are dominant, while the capacity aspect relevant to the underlying bin-packing problem are far less pronounced. Therefore, as already mentioned, we argue that the results could be further improved by introducing an insertion heuristic directly addressing the temporal aspects of the problem rather than indirectly via the travel time increases.

Also interestingly, the greedy nature of the allocation algorithm is illustrated, with the TTF ordering significantly outperforming the WTF, corresponding to success of the the well known *most constrained first* greedy heuristic applied to the time window constraints.

E. Runtime and Convergence

Within this experiment we evaluated the runtime features of the solver. As explained in Section IV the solver architecture enables a solving approach with a big number of competing particular solvers being run in parallel. The results are then extracted as a quality-increasing sub-sequence from the sequence of results of individual solvers as they finish over time.

Considering the biggest problems only (1000 nodes), the average runtime of the parallel composite solver before it found its best solution was approximately 2 and 4 minutes in the VRPTW and PDPTW cases respectively. On the other hand, the single-threaded composite runtime (sum of all solver runtimes within the full blown composite solver competition) is much higher, with the maximum of 210 and 318 minutes

	FIFO	MDF	LDF	EF	EFP	EFD	LF	LFP	LFD	TTF	WTF
VRPTW Quality	88.0%	88.6%	87.5%	93.0%	–	–	91.3%	–	–	91.5%	87.2%
% Wins	3.3%	17.9%	2.0%	35.0%	–	–	11.0%	–	–	29.1%	1.8%
PDPTW Quality	80.4%	79.0%	79.8%	–	82.3%	81.3%	–	82.7%	81.7%	85.0%	78.2%
% Wins	3.6%	0.9%	3.4%	–	25.7%	12.8%	–	9.5%	9.8%	32.0%	2.2%

TABLE III
INITIAL TASK ORDERINGS COMPARISON — AVERAGE OVERALL QUALITY AND THE PERCENT OF WINS IN THE SOLVER COMPETITION

for the VRPTW and PDPTW cases respectively. In 25.2% (VRPTW) and 4.3% (PDPTW) of the cases the best solution found by the composite solver was actually achieved using a non-iterative improvement strategy resulting in a runtime of under 10 seconds.

Thus the results suggest, that given a fitting ordering and a good initial fleet size estimate, the convergence of the approach is actually very good. No effort was made with respect to pruning the space of solvers competing within the solver competition, which might be an interesting opportunity for future research. In overall, the runtime and convergence features provided by the the agent based approach show promise in terms of providing for a potentially very flexible and fast parallel implementation with respect to the two studied problems.

VI. CONCLUSIONS

In this paper we present a parallel solver implementation based on agent negotiation for the widely studied VRPTW and PDPTW problems. The solver is based on parallel execution of particular solvers returning an improving sequence of results over time. The individual particular solvers are based on a configuration space given by a set of 7 (VRPTW) or 9 (PDPTW) task orderings, 7 improvement strategies based on agent negotiation and 2 initial fleet size settings.

An analysis of the solver performance, convergence and runtime is presented based on the widely used Homberger-Gehring and Li-Lim benchmark sets, which have been missing from previous agent-based works. With respect to the primary optimization criteria, the solver equals the best known results in 30.3% and 11.7% for the VRPTW and PDPTW, with an overall quality of 94.5% and 86.7% respectively. Considering the biggest 1000 customer instances, the time before the parallel solver found it's best solution was 2 and 4 minutes on average for the two respective problems.

Opportunities for future research are identified in: (i) pruning the space of particular solvers to boost the overall efficiency, (ii) improving convergence by adopting a more fitting heuristic than the basic travel time savings heuristic and (iii) employing more complex moves than the simple feasible single task reallocations throughout the allocation algorithm and the improvement steps.

This work thus represents an initial rigorous inquiry into applicability of agent based solving approach to the two studied problems missing from previous agent based studies. Even using quite simple local planning the solver benefits from strong agent-to-agent interactions providing for a good solution quality. Further research in both directions of fine-tuning the solver for the current problem variants or introducing a more realistic problem settings exploiting the inherent flexibility of the multi-agent problem decomposition is thus very relevant.

ACKNOWLEDGMENT

This work was supported by the Ministry of Education, Youth and Sports of Czech Republic within the Research program MSM6840770038: Decision Making and Control for Manufacturing III and also within the grant no. LD12044.

REFERENCES

[1] G. B. Dantzig and J. H. Ramser, "The truck dispatching problem," *Management Science*, vol. 6, no. 1, pp. 80–91, 1959.
[2] P. Skobelev, "Multi-agent systems for real time resource allocation, scheduling, optimization and controlling: Industrial applications," in *Holonic and Multi-Agent Systems for Manufacturing*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2011, vol. 6867, pp. 1–14.
[3] J. Homberger and H. Gehring, "A two-phase hybrid metaheuristic for the vehicle routing problem with time windows," *European Journal of Operational Research*, vol. 162, no. 1, pp. 220–238, 2005.
[4] H. Li and A. Lim, "A metaheuristic for the pickup and delivery problem with time windows," 2001.
[5] O. Bräysy and M. Gendreau, "Vehicle routing problem with time windows, part I route construction and local search algorithms," *Transportation Science*, vol. 39, no. 1, pp. 104–118, 2005.
[6] ——, "Vehicle routing problem with time windows, part II metaheuristics," *Transportation Science*, vol. 39, no. 1, pp. 119–139, 2005.
[7] S. N. Parragh, K. F. Doerner, and R. F. Hartl, "A survey on pickup and delivery problems," *Journal fr Betriebswirtschaft*, vol. 58, no. 2, pp. 81–117, 2008.
[8] Sintef. Top — http://www.sintef.no/projectweb/top/problems/.
[9] K. Fischer, J. P. Mller, and M. Pischel, "Cooperative transportation scheduling: an application domain for dai," *Journal of Applied Artificial Intelligence*, vol. 10, pp. 1–33, 1995.
[10] R. Kohout and K. Erol, "In-time agent-based vehicle routing with a stochastic improvement heuristic," in *11th Conference on Innovative Applications of Artificial Intelligence*. AAAI/MIT Press, 1999.
[11] M. M. Solomon, "Algorithms for the vehicle routing and scheduling problems with time window constraints," *Operations Research*, vol. 35, pp. 254–265, 1987.
[12] H. W. Leong and M. Liu, *A multi-agent algorithm for vehicle routing problem with time window*. ACM, 2006, p. 106111.
[13] Q. Lu and M. M. Dessouky, "A new insertion-based construction heuristic for solving the pickup and delivery problem with hard time windows," *European Journal of Operational Research*, vol. 175, pp. 672–687, 2005.
[14] J. Vokřínek, A. Komenda, and M. Pěchouček, "Agents towards vehicle routing problems," in *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1 - Volume 1*, ser. AAMAS '10. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2010, pp. 773–780.
[15] ——, "Abstract architecture for task-oriented multi-agent problem solving," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 41, no. 1, pp. 31 –40, January 2011.
[16] A. M. Campbell and M. Savelsbergh, "Efficient insertion heuristics for vehicle routing and scheduling problems," *Transportation Science*, vol. 38, pp. 369–378, August 2004.