An Efficient Route Minimization Algorithm for the Vehicle Routing Problem with Time Windows Based on Agent Negotiation

Petr Kalina[1], Jiří Vokřínek[2], and Vladimír Mařík[3]

[1] Intelligent Systems Group, Czech Technical University in Prague, Czech Republic
[2] Agent Technology Center, Czech Technical University in Prague, Czech Republic
[3] Department of Cybernetics, Czech Technical University in Prague, Czech Republic
{petr.kalina,jiri.vokrinek,vladimir.marik}@fel.cvut.cz

Abstract. We present an efficient route minimization algorithm for the vehicle routing problem with time windows. The algorithm uses a generic agent decomposition of the problem featuring a clear separation between the local planning performed by the individual vehicles and the abstract global coordination achieved by negotiation — differentiating the presented algorithm from the traditional centralized algorithms. Novel negotiation semantics is introduced on the global coordination planning level allowing customers to be temporarily ejected from the emerging solution being constructed. This allows the algorithm to efficiently backtrack in situations when the currently processed customer cannot be feasibly allocated to the emerging solution. Over the relevant widely-used benchmarks the algorithm equals the best known solutions achieved by the centralized algorithms in 90.7% of the cases with an overall relative error of 0.3%, outperforming the previous comparable agent-based algorithms.

Keywords: multi-agent systems, logistics, optimization, VRPTW.

1 Introduction

The vehicle routing problem with time windows (VRPTW) is one of the most important and widely studied problems within the operations research (OR) domain featuring in many distribution and logistic systems. It is a problem of finding a set of routes starting and ending at a single depot serving a set of geographically scattered customers, each within a specific time-window and with a specific demand of goods to be delivered. The primary objective of the VRPTW is to find the minimal number of routes serving all customers.

Real world applications of VRPTW and routing problems in general are often very complex and semantically rich, featuring heterogenous fleets with varied operational costs [20], specific constraints e.g. loading constraints [29], driver shift times [26] reflecting also the typical real-world challenges e.g. traffic congestions, breakdowns, customers rescheduling etc.

G. Boella et al. (Eds.): PRIMA 2013, LNAI 8291, pp. 149–164, 2013.

The multi-agent (MA) systems are an emerging architecture with respect to modeling new-generation systems based on smart actors and their intelligent coordination, promoting the autonomy of the individual actors. The outstanding attribute inherent to most MA planning techniques is thus the clear separation of local planning of the actors and the global coordination of their individual plans [2,15]. This enables for the problem specific constraints — such as the time-windows and capacity constraints inherent to VRPTW or those mentioned above — to be incorporated by developing fitting local planning strategies for the individual actors (vehicles). The global coordination mechanism, on the other hand, remains abstract and can be applied over a wide range of problems similar in nature e.g. general task allocation problems [28].

Within this work we present a significant extension to the global coordination framework extending the previous similar works [28,13]. The extension is based on introducing a specific backtracking strategy to the abstract allocation process based on the *ejection* principle [19], denoted here as *rotations*. The contribution is significant for two reasons: (i) it enables the resulting VRPTW algorithm to significantly outperform the previous comparable agent-based algorithms with a performance closely matching the established centralized algorithms and (ii) similar concepts within the coordination framework can be applied in a number of other similar problems — namely in highly constrained task allocation problems with scheduling/sequencing aspects e.g. general scheduling and temporal resource allocation problems and their extensions.

Thus a VRPTW algorithm is presented and the solving process is outlined in detail. The semantic of the negotiation based coordination process is analyzed and compared to the previous similar methods. In order to provide a relevant comparison of the presented algorithm to the best known centralized algorithms the experimental evaluation is based on the two most widely used benchmarks known from the OR literature. The comparison to the previous similar agent-based algorithms is provided as well.

As mentioned above, the presented algorithm differentiates itself from the traditional algorithms by featuring the clear separation between the problem specific agents' local decision making and the abstract global coordination, representing the core feature inherent to cooperative multi-agent planning [2,15]. On the other hand the finer details inherent to agency e.g. alternative agent behavioral patterns and commitment semantics [17], the complexity of the underlying network communication [6] or semantically rich problem extensions [10] are not discussed in detail, presenting further opportunities for future research.

2 Problem Statement and Notations

As mentioned above, the VRPTW consists of finding a set of routes starting and ending at a single depot serving a set of geographically scattered customers. The primary optimization criteria is to find a minimal set of such routes serving all the customers within the given time windows.

Let $\{1..N\}$ represent the set of customers with the depot denoted as 0. For each customer c_i let (e_i, l_i, s_i, d_i) denote the earliest/latest service start times

(*time window*), service time and the demand. Let a sequence of customers $\langle c_0, c_1, ..c_m, c_{m+1}\rangle$ denote a single route with c_0 and c_{m+1} denoting the depot. Let D denote the vehicle capacity and let $t_{i,j}$ correspond to the travel time between customers c_i and c_j with all mutual travel times being known.

The objective of the VRPTW is finding a minimal set of routes serving all customers. For each route $\langle c_0, c_1, ..c_m, c_{m+1}\rangle$ the sum of corresponding customers' demands must be lower than the capacity of the vehicle serving the route $\sum_1^m d_i \leq D$ (capacity constraint) while the service at each customer c_i must begin within the interval given by (e_i, l_i) (time-windows constraints).

Given a route $\langle c_0, c_1, ..c_m, c_{m+1}\rangle$ let (E_i, L_i) correspond to the *earliest* and *latest possible service start* at customer c_i computed recursively according to:

$$\begin{aligned} E_1 &= max\,(e_1, t_{0,1}) \\ E_i &= max\,(e_i, E_{i-1} + s_{i-1} + t_{i-1,i}) \end{aligned} \tag{1}$$

and

$$\begin{aligned} L_m &= l_m \\ L_i &= min\,(l_i, L_{i+1} - t_{i,i+1} - s_i) \end{aligned} \tag{2}$$

As shown in [5], the time window constraints are satisfied when $E_i \leq L_i$ for all $i \in 1..m$. The capacity constraint is satisfied when $\sum_1^m d_i \leq D$.

3 Related Work

The VRPTW has been extensively studied for for almost thirty years with the classical Solomon's [27] article dating back to 1987. The full review of relevant OR literature is outside the scope of this study, however we refer the reader to the excellent surveys of recent (up to 2005) literature provided by [3,4]. Instead we briefly introduce the state-of-the-art traditional algorithms and concentrate on the relevant agent-based studies. The performance of the individual algorithms is evaluated using the *cumulative number of vehicles* (CNV) metric used widely within OR field, corresponding to the total number of routes/vehicles over all problem instances across the whole corresponding set of problem instances.

3.1 Traditional Algorithms

As already mentioned, the majority of successful VRPTW algorithms combine several methods in a multi-phase solving approach. Thus within the initial *construction* phase a set of initial solutions is generated. The *route minimization* can be part of this phase (as it is with the presented algorithm) or it can be embedded as a separate phase - addressing the primary objective. The secondary objectives are typically addressed in an additional phase within which the number of routes remain constant [3,4,24,19,25]. The route minimization is also considered as being the computationally hardest phase of solving the VRPTW [3,4].

Thus developing efficient route minimization heuristics as a key initial step in providing efficient VRPTW algorithms.

The algorithm introduced in [19] is based on the *ejection pools* (EP) principle, performing very good potentially unfeasible insertions of customers to individual routes and subsequently recovering the feasibility by ejecting some other customers from the unfeasible routes. The insertion-ejection phase is interleaved with a local search procedure dynamically improving the solution throughout the solving process.

The algorithm presented by [25] is based on embedding a branch-and-price algorithm based on a modification of an exact algorithm presented in [9] into a large neighborhood search procedure based on series of destructive and recreative steps. Thus a subset of (routed or unrouted) customers is selected based on one of the four various operators introduced to provide means of search diversification and the identified customers are removed from the partial solution. The neighborhood corresponding to the partial solutions obtained by reinserting the removed customers is traversed using a branch-and-price algorithm in an effort to identify the next best partial solution and the process continues from there.

A memetic algorithm [22] achieving the contemporary best CNV of 405 and 10304 over the Solomon's and Homberger's benchmark respectively is presented in [24]. Initial feasible solutions are generated using an adaptation of the EP principle used by [19]. The EP mechanism is accompanied by a powerful feasible insertion method denoted as *squeeze* as well as a search diversification *perturb* procedure. The *squeeze* method also employs a specific adaptive local search procedure used to repair potentially unfeasible intermediate solutions using a heuristic carried over from [19]. The route minimization phase is followed by a travel time minimization phase combining an evolutionary algorithm used for traversing the more distant parts of the search space (the *exploration* phase) with a local search based improvement phase providing for traversing the immediate neighborhoods of the examined solutions (referred to as the *exploitation* phase). The exploration algorithm is based on adaptation of the known EAX operator [23], while the exploitation phase combines two alternative hill-climbing methods aimed at (i) reducing the level of unfeasibility within the intermediate emerging solution and (ii) reducing the travel time.

3.2 Agent-Based Algorithms

A number of approaches and systems have been proposed addressing a variety of routing and logistic problems relying on decentralized agent-based planning techniques. A survey of some of the most interesting works is provided for example by [7]. Also a number of studies was presented addressing the VRPTW in particular. However, as discussed in detail within the Section 5 there are some common deficiencies shared by most of these works, namely that (i) a relevant comparison of the presented algorithms to the established traditional algorithms is missing and (ii) where provided, it reveals a relatively weak performance of these algorithms. Thus within this work we present an agent-based VRPTW algorithm which boast a vastly improved performance and provide support for

such a claim by pitting it against both the most relevant benchmarks known from the OR literature.

The algorithm presented in [10] is built around the concepts of Shipping Company Agents (SCA) representing individual shipping companies and their respective fleets of Truck Agents (TA). After registering a customer, the SCA negotiates with his fleet of TAs to estimate the price of serving the customer. The other SCAs are contacted as well and based on the corresponding cost estimates the SCA may assign the customer to one of its trucks or decide to cooperate with another company. The planning within each fleet is done dynamically and is based on the well known contract net protocol (CNP) [8] accompanied by a *simulated trading* improvement strategy based on finding the optimal customer exchanges by solving a modification of the maximal pairing problem on a graph representing the proposed exchanges originally presented in [1]. Both cooperative and competitive models are explored with respect to the cooperation of individual SCAs. Also a specific model for simulating traffic jams is presented.

The algorithm for the closely related pickup and delivery problem with time windows (PDPTW) presented by [14] is essentially a parallel insertion procedure with a subsequent improvement phase consisting of reallocating some randomly chosen tasks from each route. The used cost structure is based on the well known Solomon's I1 insertion heuristic [27].

The algorithm presented by [18] is based on agents representing individual customers, 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.

In [6] an algorithm is introduced based on Order agent — Scheduling agent — Vehicle agent hierarchy. The algorithm is based on a modified CNP insertion procedure limiting the negotiation to agents whose routes are in proximity of the customer being allocated in an effort to minimize the number of negotiations.

The algorithm presented in [12] features a similar agent architecture as the one used within this study inspired by the general agent architecture for task allocation problems presented in [28]. The negotiation process corresponds to a parallel customer insertion procedure with a specific *iterative* improvement method being periodically invoked after processing each customer. Also, an additional *final* improvement method is invoked at the end of the solving process after all customers have been allocated. Several alternative improvement methods are introduced based on relocating alternative sets of customers between the individual routes. The used local planning strategy corresponds to the well known *travel time savings* heuristics [27].

An improved version of the algorithm is presented in [13]. An alternative *slackness savings* based vehicles' local planning strategy is introduced based on [21] as well as the set of refined improvement methods to be used throughout the solving process. Also a specific parallel execution wrapper is introduced based on running in parallel the individual *algorithm instances* corresponding to the

previously discussed algorithm differentiated by the order in which the customers are processed as well as the choice of the perticular improvement methods being used. The algorithm achieves a CNV of 10949 over the two mentioned benchmarks, corresponding to 2.2% relative error compared to [24], representing the best comparable result for an agent-based algorithm.

4 Agent-Based Algorithm for VRPTW with Rotations

Within this work we present an extension to the global coordination framework used by the previous similar works [28,13]. The main contributions is the introduction of a novel negotiation semantics allowing customers to be temporarily ejected from the emerging solution being constructed by means of *rotations*, providing for an efficient backtracking strategy. This enables the algorithm to proceed in situations where the semantic used by the previously presented algorithms got stuck and the negotiation process failed.

The underlying abstract negotiation framework is based on [28] and features the clear separation between the local planning of individual vehicles and the abstract global coordination achieved by negotiation — differentiating the presented algorithm from the traditional centralized algorithms. Thus, as already mentioned, the abstract coordination mechanism is generic and can be applied to a variety of problems that are similar in nature [28] — by developing a fitting local planning strategy to be used by individual vehicles (agents/resources in general). This enables for transparent inclusion of specific extensions such as heterogeneities within the fleet [20], loading or cargo organization strategies [29], driver-relevant constraints [26]. Alternatively the autonomic nature of the system can be exploited by introducing autonomous trajectory/path planning strategies or reflecting the non-cooperative or tactical aspects of the modeled system [16].

4.1 Abstract Global Coordination Framework

The abstract negotiation-based framework featuring at the global coordination level of the overall planning process is a modification of a similar framework presented in [13]. The underlying agent architecture is carried over 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 cusotomers and submitting them to the underlying Allocation Agent.

Allocation Agent instruments the actual solving process by negotiating with the Vehicle Agents, corresponding to the global coordination phase of the overall planning process. The negotiation is conducted based upon the commitment/decommitment cost estimates provided by the Vehicle Agents.

```
Input: Stack of customers C, Fleet of empty vehicles — initial solution σ
Output: Solution σ — complete or partial based on success of the process

Procedure negotiate(C, σ)
 1:     Initialize rotation counters ∀c ∈ C, c.rts := 0;
 2:     while (C not empty and rotations < rotationLimit)
 3:       Pop c̄ ∈ C (LIFO strategy);
 4:       Cheapest := {v ∈ Feasible(σ, c̄), v.costCommit(c̄) is minimal};
 5:       if (Cheapest ≠ ∅) then
 6:         Randomly select v̄ ∈Cheapest;
 7:         v̄.commit(c̄);
 8:         SHAKE(σ);
 9:       else
10:         SQUEEZE(c̄, σ);
11:       endif
12:       if (c̄ ∉ σ) then
13:         C_ej := ROTATE(c̄, σ);
14:         Push C_ej to top of C
15:         c̄.rts++;
16:         rotations++;
17:       endif
18:     enwhile
19:     return σ;
End
```

Fig. 1. The Abstract Global Coordination Process

Vehicle Agent represents an individual vehicle serving a route. It provides the Allocation Agent with the above mentioned inputs, computed based on the Vehicle Agent's local (private) planning strategy.

Figure 1 illustrates the modified coordination process instrumented by the Allocation Agent. In essence it consists of a series of *negotiation* interactions between the Allocation Agent and the vehicles represented by the Vehicle Agents, being part of the initially empty emerging solution σ. The process is abstract and is based solely on the cost estimates computed locally by the vehicles based upon the particular local planning strategy being used.

The process is started by resetting the *rotation counters* for all allocated customers (line 1). As discussed later, the counters are used for determining the best rotation within the ROTATE method (line 13) in an effort to diversify the search. Follows an attempt to feasibly allocate the customer to the emerging partial solution σ (lines 3 – 11). Initially the vehicles that can feasibly serve the customer (the set *Feasible*(σ, $\bar{c}$)) at the lowest possible cost (the set *Cheapest* — a subset of *Feasible*(σ, $\bar{c}$)) are identified based on the cost estimates provided by the individual Vehicle Agents (line 4). In case *Cheapest* $\neq \emptyset$ a random vehicle from the set is chosen and commits to serving the customer. In such a case an attempt follows to further improve the emerging solution within the SHAKE

method (line 8). In the opposite case a further attempt is made to feasibly *squeeze* the customer into the emerging solution σ within the SQUEEZE method (line 10).

Both the SHAKE and the SQUEEZE methods are based on traversing the neighborhoods of the emerging solution σ in an effort to (i) improve the solution in a way that increases the chance for future customer allocations (the SHAKE method) and (ii) modify it in a way as to enable feasible insertion of $\bar{c}$ (the SQUEEZE method). The semantic of the corresponding negotiation is slightly different for the two methods and is discussed in detail in Section 4.2.

In case a feasible slot is not found for the customer $\bar{c}$ in any of the routes neither within the initial allocation effort nor within the SQUEEZE method the feasible allocation process has failed. Such a situation arises typically towards the end of the solving process as the individual routes get denser and all of the possible allocation slots are rendered unfeasible due to capacity or time-window constraints[1]. In such a situation an attempt is made to insert the customer to one of the routes at the expense of ejecting some other customers from that route. Thus, within the ROTATE method (line 13) each of the vehicles tries to identify a fitting set of customers in its schedule the ejection of which would enable the customer $\bar{c}$ to be feasibly inserted — an operation referred to as a *rotation* — and the most fitting rotation is identified and performed. As already mentioned the selection of the most fitting rotation is based on the values of the rotation counters with the cost of an ejection corresponding to $\sum_{i=1}^{k} c_{e_i}.rts$, with the $c_{e_i}, i = 1..k$ being the ejected customers. The criteria thus favors rotations that are (i) smaller and (ii) consist of customers that have caused the least number of rotations and thus are arguably easier to allocate feasibly. The ROATE method is discussed in detail within the Section 4.3.

The process continues with incrementing the rotation counter for the allocated customer $\bar{c}.rts$ and the global *rotations* counter. Thus either all customers are allocated or the *rotationLimit* is exceeded. In the latter case there still remain some unserved customers (solution σ is not complete). In such a case the process can be restarted with a different fleet size or an additional vehicle can be added to the fleet.

The initial size of the fleet — corresponding to the initial solution σ of empty vehicles — is thus a significant factor affecting the efficiency of the whole algorithm and should correspond to the theoretical lower bound on the number of vehicles for the solved problem instance. An estimate of this number can be computed based upon the total demand of all customers $d_t = \sum_{i=1}^{N} d_i$ and the

[1] The latter situation is actually much more frequent suggesting the temporal/schedulling aspects are the dominant constraining factor for the solved problem. However, we argue that this is also partly due to the fact that the problem instances from the used benchmarks present a rather simple variant of the underlying multiple bin-packing problem. Individual customer demands are rather small and distributed only across several specific values (e.g. 10, 20, 30 with the vehicle capacity being 200) — an observation that might prove to be interesting from the point of view of the OR community where the used benchmarks are widely used and considered de-facto standard.

vehicle capacity D as d_t/D. An alternative estimate uses the temporal mutual incompatibilities between the customers. Two customers are temporally incompatible if it's not possible to start the service in either of them at the earliest possible moment and then reach the other customer in time, given the corresponding time windows. Considering a graph with nodes corresponding to the customers and edges to their temporal incompatibilities, the minimal size of the fleet is given by the size of the maximal clique within this graph [21]. Thus the overall theoretical lower bound on the size of the fleet is the bigger of these two numbers. As the max-clique problem is a *NP*-hard problem in its own right, the latter number is computed using a well known $O(N^3)$ polynomial approximate algorithm presented in [21].

4.2 The Shake and the Squeeze Methods

As already mentioned, the two methods are based on traversing the closer neighborhoods of the emerging solution σ. With the Shake method an effort is made to improve the solution by performing *improving* relocations of customers within and between the routes. The term improving reflects the fact that only moves that increase the utility from the point of view of the cost structure provided by the used local planning strategy are executed. In case of the used travel time savings (TTS) strategy [13] this corresponds to *shaking* the customers in a way as to reduce the traveled distances (with vehicle speed being constant for all the vehicles, the travel times are proportional to traveled distances). Thus only relocations that result in decrease in the overall traveled distance of the whole solution are performed.

The Squeeze method uses a similar semantics, however with some key extensions reflecting the effort to allocate the customer $\bar{c}$ that has previously failed to be feasibly allocated. Each relocation effectively consists of removing a customer (denoted as c_r) from the original route v and then requiring all the vehicles to estimate the feasibility and cost of a possible insertion of c_r to their schedules. Within the Squeeze method, after the relocated customer c_r was removed from the underlying route the feasibility of inserting the customer $\bar{c}$ is within this route is examined. In case the insertion of $\bar{c}$ to v is feasible, it is performed and an effort is made to feasibly allocate the c_r to the thus modified solution. In case the attempt is successful, the $\bar{c}$ has been successfully squeezed within the merging solution σ. In the opposite case the $\bar{c}$ is removed and the relocation proceeds by reinserting c_r back to v (to either the original or a different slot within the schedule).

The importance of the Squeeze method is twofold. On one hand it presents an opportunity to examine some simple yet helpful moves during the phase where the emerging solution σ is improved, potentially avoiding the need to perform a full fledged rotation. On the other hand it marks the place within the algorithm where there is arguably the biggest potential for further improvement in terms of performance. The contemporary VRPTW algorithms e.g. [19,24] present strategies enabling to repair the unfeasible intermediate solutions within a specific local search procedure guided by an utility function corresponding

to the level of unfeasibility of the solution. As will be discussed later within the Section 5.2 we believe such an approach could significantly improve the convergence and efficiency of the algorithm.

4.3 The Rotate Method

As already mentioned above, the Rotate method enables the algorithm to proceed even when it is not possible to feasibly allocate the currently processed customer $\bar{c}$. In previous similar algorithms [12,13] such a situation caused the whole allocation process to fail, requiring another vehicle to be added to the fleet. The situation typically occurs towards the end of the solving process as the solution gets denser and the schedules of individual routes tighter. Based on previous findings [13], given the customers are processed in an unfitting order the situation is even likelier to occur. The resulting algorithms are therefore very sensitive to the initial customer ordering.

A single *rotation* thus consists of identifying a specific set of customers to be ejected from one of the routes such that it is possible to subsequently feasibly insert the currently processed customer $\bar{c}$ to the route. In order to identify the best possible rotation two criteria can be considered — the effects the ejecting of the specific identified set of customers will have on the solving process and also the fittingness of the subsequent insertion of $\bar{c}$ made possible by the ejection. Considering the first criterion, it is especially important to prevent the rotation mechanism from cycling by appropriately diversifying the ejections.

Within this work thus an approach is used that is an adaptation of the insertion-ejection mechanism presented in [23]. For each route the possible ejections are traversed in a specific way corresponding to the lexicographic ordering. Let $\langle c_{e_1}, c_{e_2}, .., c_{e_k} \rangle$ denote an ejection of size k from the route $\langle c_0, c_1, ..c_m, c_{m+1} \rangle$ with $e_i \in 1..m$. The maximal rotation size is bounded by the k_{max} parameter. Thus for example given $k_{max} = 3$ the rotations are traversed in the following order: $\langle c_1 \rangle$, $\langle c_1, c_2 \rangle$, $\langle c_1, c_2, c_3 \rangle$, $\langle c_1, c_2, c_4 \rangle$ ... $\langle c_1, c_2, c_m \rangle$, $\langle c_1, c_3, c_4 \rangle$ ... $\langle c_1, c_{m-1}, c_m \rangle$, $\langle c_1, c_m \rangle$, $\langle c_2 \rangle$, $\langle c_2, c_3 \rangle$ etc. The cost of an ejection is determined based on the values of the rotation counters as $\sum_{i=1}^{k} c_{e_i}.rts$ and the ejection with the minimal cost is chosen. In case of equality the following criteria is used (in following hierarchical order): (i) minimization of the size of the ejection and (ii) minimization of the travel time increase for the corresponding route after $\bar{c}$ has been inserted. These criteria have proved to be the most efficient based on our computational tests being out of scope of this study.

The ejected customers are then added to the set (stack) C using a LIFO strategy. This means that prior to proceeding to the customer following $\bar{c}$ in the original customer ordering, the $\bar{c}$ and all the ejected customers from the corresponding chain of rotations have to be feasibly allocated. Such an approach corresponds to the fact that if any of the customers cannot be feasibly allocated to the partial solution σ at this point, it is very unlikely it could be allocated later within the solving process as the solution becomes even more constrained. It is also supported by our computational experiments not being presented in detail within this study. Thus in effect the solution is *rotated* until $\bar{c}$ can be

Table 1. Results for alternative settings of the *rotationLimit* parameter

Set	CNV	Absolute (relative) error for alternative *rotationLimit* values			
		Presented Algorithm			
	Nagata [24]	2,000	5,000	20,000	1,000×N/5
100	405	5 (1.2%)	4 (1.0%)	3 (0.7%)	3 (0.7%)
200	694	5 (0.7%)	4 (0.6%)	2 (0.3%)	0 (0.0%)
400	1,381	18 (1.3%)	13 (0.9%)	12 (0.9%)	4 (0.3%)
600	2,067	35 (1.7%)	23 (1.1%)	11 (0.5%)	4 (0.2%)
800	2,738	63 (2.3%)	43 (1.6%)	16 (0.6%)	8 (0.3%)
1000	3,424	83 (2.4%)	49 (1.4%)	18 (0.5%)	8 (0.2%)
All	10,709	206 (1.9%)	134 (1.3%)	55 (0.5%)	27 (0.3%)

feasibly allocated, corresponding to an efficient backtracking strategy based on the introduced search diversification heuristic guiding the ejections.

The number of potential ejections is huge. Also testing the feasibility of insertion of $\bar{c}$ for a particular ejection requires the E_i, L_i values (see Equations 1 and 2) to be recomputed along the route [5]. Therefore in order to speed up the process several pruning strategies have been developed. The first strategy is trivial - by storing the contemporary best cost ejection all the ejections with costs higher than this ejection can be ignored. The second strategy is based on two trivial observations: (i) if a rotation is a subset of an unfeasible rotation (in terms of allowing $\bar{c}$ to be inserted after the ejection) it is also unfeasible and (ii) if a rotation is a superset of a feasible rotation it is feasible and has higher cost than that rotation. In both of these cases thus the corresponding rotations can be pruned. The mentioned strategies provide a significant speedup of the rotation process. However we argue that significant saving could be achieved by introducing concepts of interesting temporal/spatial neighborhoods to the overall negotiation process that would enable to limit the number of considered routes and insertion slots within this and also the other phases of the negotiation process to only the heuristically identified interesting neighborhoods.

5 Experimental Validation

The experiments were carried out using the two well known benchmarks of Homberger and Solomon [27,11]. The Solomon's set consists of 56 problem instances with 100 customers each, while the extended Homberger's provides for additional 5 sets of 60 instances with 200, 400, 600, 800 and 1000 customers respectively sharing otherwise the same basic attributes as the Solomon's problems. Thus for each instance size there are 6 instance types provided — the R1, R2, RC1, RC2, C1, and C2 type, each with a slightly different topology and time windows properties. For C1 and C2 types the customer locations are grouped in clusters, unlike the R1 and R2 classes where the customers are randomly placed.

Table 2. Comparison with previous agent-based VRPTW algorithms

Set	CNV	Absolute (relative) error				
	Nagata	Agents				
	[24]	*Fisher* [10]	*Leong* [18]	*Kalina* [12]	*Kalina* [13]	*Presented*
100	405	31 (7.7%)			25 (6.2%)	3 (0.7%)
100-R1	143		30 (21.0%)		11 (7.7%)	2 (1.4%)
200-1000	10,304			573 (5.6%)	331 (3.2%)	24 (0.2%)
All	10,709				240 (2.2%)	27 (0.3%)

The RC1 and RC2 instance types combine the previous two types with a mix of both random and clustered 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 providing for routes of around 50 customers.

The presented results correspond to the Travel Time Savings [13] local planning strategy being used by the individual vehicles and a $k_{max} = 3$ setting for the maximal rotation size.

5.1 Overall Solution Quality Analysis

The performance of the algorithm in terms of the primary optimization criteria is illustrated by Table 1. The results are listed for alternative subsets from the experimental data identified by the first column, expressed using the previously discussed cumulative number of vehicles (CNV) metric. The "100" subset corresponds to the Solomon's benchmark while the "200–1000" rows denote alternative sizes within the Homberger's benchmark. The "All" row lists the overall result over both benchmarks. The second column lists the CNV achieved by [24] representing the currently best known overall results. The rest of the columns correspond to the absolute and relative errors in terms of CNV for various settings for the *rotationLimit* parameter, with the N denoting the size of the instance for the setting within the last column.

The Table 2 further presents the comparison of the algorithm to the previous comparable agent-based algorithms. The notation is similar as with the previous table with "100-R1" corresponding to the R1-type problems from the Solomon's benchmark and the 200–1000 row denoting the complete Homberger's benchmark. Note that to our knowledge these are the only comparable results presented by the previous agent-based VRPTW studies addressing at least specific subsets of problems from the two benchmarks otherwise widely used within the OR community.

In the most complex setting the algorithm was able to match the best known solutions in 90.7% of all tested problem instances, resulting in an overall relative error of 0.3%. Also importantly, the algorithm significantly outperforms the

Table 3. Average runtimes for individual instance sizes

Size	Lim [19]	Prescott-Gagnon [25]	Nagata [24]	*Presented*
100	39 min	192.5 min	25 min	9 min
200	93 min	265 min	20.5 min	67 min
400	296 min	445 min	81 min	345 min
600	647 min	525 min	80.4 min	615 min
800	1269 min	645 min	138 min	1253 min
1000	1865 min	810 min	186 min	1765 min
CPU	P4-2.8G	OPT-2.4G	OPT-2.4G	K8-2.3G

comparable previously presented agent-based algorithms over the comparable problem subsets.

The effect of the proposed ejection based ROTATE method is clearly illustrated. The method is based on a powerful search diversification criteria using the rotation counters $c.rts, c \in C$ to express the measure of difficulty of feasibly allocating individual customers. Considering the most complex setting with the *rotationLimit* parameter being linearly proportional to the size of the problem instance the performance of the algorithm is consistent across alternative instance sizes. This further supports the efficiency of the proposed method with respect to the existing traditional centralized algorithms. The difference in performance over the Solomon's 100 customer instances is arguably due to the fact that over the smaller benchmark the competing algorithms (that are typically not computationally bound) often use a non-proportionally long running times, as also illustrated by Table 3.

5.2 Runtime and Convergence Analysis

The comparison in terms of runtime with the traditional VRPTW algorithms is presented by Table 3. The listed values correspond to the average runtime for individual instance sizes. The abbreviations in the "CPU" row correspond to AMD Opteron 2.4GHz, Pentium 4 2.8GHz and AMD K8 2.4GHz processors.

The results show that the convergence of the presented algorithm is significantly worse than in case of the best compared traditional algorithm [24] as outlined by the increasing gap between the runtimes for bigger instance sizes. Note also, that the runtimes listed for the compared algorithm include addressing also the secondary travel time optimization criteria.

As already mentioned, we argue that this is primarily due to the fact that the negotiation semantic adopted by the SQUEEZE and SHAKE methods is very simple. When the currently processed customer $\bar{c}$ cannot be feasible allocated to the emergent solution σ the solution is *rotated* until all affected customers can be *squeezed* in. The SQUEEZE method is based on performing simple customer relocations between the routes. However, at this stage of the solving process, the solution σ is already tightly constrained and the chance of modifying it

significantly using the simple negotiation semantic is correspondingly as most possible relocations are rendered unfeasible due the time-window or capacity constraints. Thus the number of necessary rotations grows dramatically.

In comparison, the *sqeeze* method employed by the compared traditional centralized algorithm [24] corresponds to a much more sophisticated local search procedure. Apart from customer relocations several other moves are considered. Most significantly, also the unfeasible moves are considered and a specific cost function is introduced driving the search process corresponding to the *level of unfeasibility* of the underlying intermediate solution. Thus the chance of allocating the customers affected by the individual rotations into the tightly constrained solution is higher, resulting in significantly less rotations being performed.

6 Conclusion

Within this paper we introduce an efficient agent-based algorithm for the VRPTW. The algorithm is based on coordinating the fleet of autonomously planning vehicles within an abstract global coordination framework based on agent negotiation. The main contribution of the paper is the extension of the used negotiation semantic. Core to the novel semantic is the introduction of the Rotate and Squeeze methods, enabling the algorithm to efficiently backtrack in situations where the previous algorithms failed. Due to the clear separation between the local planning and the global coordination the presented framework can be easily adopted to (i) incorporate typical real-word extensions of the problem and to (ii) solve a variety of task allocation and scheduling problems in general, further supporting the significance of this study.

The performance of the resulting algorithm is evaluated using the relevant widely-used benchmarks known from the OR literature. A comparison the traditional centralized algorithms is presented in terms of algorithm's performance and convergence. The algorithm equals the best-known solutions in 90.7% of all considered problem instances with an average relative error of 0.3%. Also, a comparison to the previous comparable agent-based algorithms is presented revealing that the algorithm significantly outperforms these algorithms.

The coordination solving process is discussed in detail. Promising research opportunities were identified in: (i) introducing more complex negotiation semantics within the Squeeze method enabling for recovering feasibility of intermediate infeasible solutions and (ii) exploiting the potential of the algorithm by adapting it to complex problem variants by introducing real-world relevant constraints to the local planning strategy — or pitting it against different task allocation and scheduling problems.

Acknowledgements. This effort was supported by the Grant Agency of the CTU in Prague, grant No. SGS12/188/OHK3/3T/13, the Ministry of Education, Youth and Sports of the Czech Republic, grant No. LD12044 and the Centre of Applied Cybernetics III, Competence Centre funded by TACR, grant No. TE01020197.

References

1. Bachem, A., Hochstättler, W., Malich, M.: The simulated trading heuristic for solving vehicle routing problems. Technical report, Discrete Applied Mathenatics (1996)
2. Brafman, R.I., Domshlak, C.: From one to many: Planning for loosely coupled multi-agent systems. In: Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS), pp. 28–35 (2008)
3. Bräysy, O., Gendreau, M.: Vehicle routing problem with time windows, part I route construction and local search algorithms. Transportation Science 39(1), 104–118 (2005)
4. Bräysy, O., Gendreau, M.: Vehicle routing problem with time windows, part II metaheuristics. Transportation Science 39(1), 119–139 (2005)
5. Campbell, A.M., Savelsbergh, M.: Efficient insertion heuristics for vehicle routing and scheduling problems. Transportation Science 38, 369–378 (2004)
6. Dan, Z., Cai, L., Zheng, L.: Improved multi-agent system for the vehicle routing problem with time windows. Tsinghua Science Technology 14(3), 407–412 (2009)
7. Davidsson, P., Henesey, L., Ramstedt, L., Törnquist, J., Wernstedt, F.: An analysis of agent-based approaches to transport logistics. Transportation Research Part C: Emerging Technologies 13(4), 255–271 (2005)
8. Davis, R., Smith, R.G.: Negotiation as a metaphor for distributed problem solving. Artificial Intelligence 20, 63–109 (1983)
9. Desaulniers, G., Lessard, F., Hadjar, A.: Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. Transportation Science 42(3), 387–404 (2008)
10. Fischer, K., Müller, J.P., Pischel, M.: Cooperative transportation scheduling: an application domain for dai. Journal of Applied Artificial Intelligence 10, 1–33 (1995)
11. Gehring, H., Homberger, J.: A two-phase hybrid metaheuristic for the vehicle routing problem with time windows. European Journal of Operational Research 162(1), 220–238 (2005)
12. Kalina, P., Vokřínek, J.: Parallel solver for vehicle routing and pickup and delivery problems with time windows based on agent negotiation. In: 2012 IEEE Conference on Systems, Man, and Cybernetics (SMC), pp. 1558–1563 (2012)
13. Kalina, P., Vokřínek, J., Mařík, V.: The art of negotiation: Developing efficient agent-based algorithms for solving vehicle routing problem with time windows. In: Mařík, V., Lastra, J.L.M., Skobelev, P. (eds.) HoloMAS 2013. LNCS, vol. 8062, pp. 187–198. Springer, Heidelberg (2013)
14. Kohout, R., Erol, K.: In-time agent-based vehicle routing with a stochastic improvement heuristic. In: 11th Conference on Innovative Applications of Artificial Intelligence. AAAI/MIT Press (1999)
15. Komenda, A., Novák, P., Pěchouček, M.: Domain-independent multi-agent plan repair. Journal of Network and Computer Applications (in print, 2013)
16. Komenda, A., Vokrinek, J., Cap, M., Pechoucek, M.: Developing multiagent algorithms for tactical missions using simulation. IEEE Intelligent Systems 28(1), 42–49 (2013)
17. Komenda, A., Vokřínek, J., Pěchouček, M.: Plan representation and execution in multi-actor scenarios by means of social commitments. Web Intelligence and Agent Systems 9(2), 123–133 (2011)
18. Leong, H.W., Liu, M.: A multi-agent algorithm for vehicle routing problem with time window. In: Proceedings of the 2006 ACM Symposium on Applied Computing, SAC 2006, pp. 106–111. ACM, New York (2006)

19. Lim, A., Zhang, X.: A two-stage heuristic with ejection pools and generalized ejection chains for the vehicle routing problem with time windows. INFORMS Journal on Computing 19(3), 443–457 (2007)
20. Liu, F.-H., Shen, S.-Y.: The fleet size and mix vehicle routing problem with time windows. Operational Research Society 50, 721–732 (1999)
21. Lu, Q., Dessouky, M.M.: A new insertion-based construction heuristic for solving the pickup and delivery problem with hard time windows. European Journal of Operational Research 175, 672–687 (2005)
22. Moscato, P.: On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Technical Report C3P Report 826, California Institute of Technology (1989)
23. Nagata, Y.: Edge assembly crossover for the capacitated vehicle routing problem. In: Cotta, C., van Hemert, J. (eds.) EvoCOP 2007. LNCS, vol. 4446, pp. 142–153. Springer, Heidelberg (2007)
24. Nagata, Y., Bräysy, O., Dullaert, W.: A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. Comput. Oper. Res. 37(4), 724–737 (2010)
25. Prescott-Gagnon, E., Desaulniers, G., Rousseau, L.-M.: A branch-and-price-based large neighborhood search algorithm for the vehicle routing problem with time windows. Netw. 54(4), 190–204 (2009)
26. Ren, Y., Dessouky, M., Ordóñez, F.: The multi-shift vehicle routing problem with overtime. Comput. Oper. Res. 37(11), 1987–1998 (2010)
27. Solomon, M.M.: Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations Research 35, 254–265 (1987)
28. Vokřínek, J., Komenda, A., Pěchouček, M.: Abstract architecture for task-oriented multi-agent problem solving. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 41(1), 31–40 (2011)
29. Wang, F., Tao, Y., Shi, N.: A survey on vehicle routing problem with loading constraints. In: Proceedings of the 2009 International Joint Conference on Computational Sciences and Optimization, CSO 2009, vol. 2, pp. 602–606. IEEE Computer Society, Washington, DC (2009)