Improved Agent Based Algorithm for Vehicle Routing Problem with Time Windows using Efficient Search Diversification and Pruning Strategy

Petr Kalina[1] and **Jiří Vokřínek**[2]

Abstract.

We suggest an improved algorithm for the vehicle routing problem with time windows (VRPTW). The algorithm is based on negotiation between a fleet of agents corresponding to the routed vehicles using a set of generic negotiation methods and a state-of-the-art insertion heuristic. A search diversification and pruning strategy is introduced which allows for a wide range of competing algorithm instances to be instantiated and efficiently managed throughout the solving process. Experimental results on the widely used Solomon's and the extended Homberger's benchmarks prove that the algorithm is broadly competitive with respect to the established centralized state-of-the-art algorithms equalling the best known solutions in 64% of the cases with an overall relative error of 2.4%, thus achieving a new best known result for an agent based algorithm to date. The vastly improved negotiation process and the inherent parallelization features provide for excellent anytime features, outperforming even the state-of-the-art algorithms in this respect.

1 Introduction

The vehicle routing problem with time windows (VRPTW) is one of the most widely studied problems in the the domain of logistics. The VRPTW is a problem of finding a set of routes from a single depot to serve customers at geographically scattered locations. Each customer is visited by exactly one route with each route starting and ending at the depot. There are two constraining factors that need to be considered: (i) for each route the sum of demands of the customers served by the route must not exceed the capacity of the vehicle serving the route (capacity constraint) and (ii) the service at each customer must begin within a given time interval (time window constraints). The primary objective of the VRPTW is to find the minimum number of routes servicing all customers.

Recent years have seen a growing interest of the scientific community in multi-agent systems as an emerging choice for modeling complex systems with highly dynamic, heterogenous, potentially non-cooperative or privacy conscious environments. The real world applications of routing algorithms often display many of the above mentioned characteristics. Traditionally, the majority of VRPTW related research is concerned with centralized algorithms, with the agent based studies being scarce and typically concerned rather with the real-world applicability of the presented algorithms than their thorough performance and theoretical analysis.

Several very recent agent based works [6, 5] reported a competitive solution quality and provided a sound theoretical analysis of the respective algorithms. Within this paper we extend these works by introducing an improved parallel algorithm based on agent negotiation (Section 3) using an efficient search diversification and pruning strategy (Section 4). The performance, convergence and other aspects of the algorithm are assessed using the widely used benchmark sets of Solomon [14] and Homberger [4] (Section 5).

2 Related Work

We refer the reader to a comprehensive survey of VRPTW algorithms up to year 2005 provided by [1]. Also, the list the contemporary leading algorithms with respect to the benchmarks used within this study is maintained at [13].

In [9] the authors present an algorithm based on the *ejection pool* principle. The algorithm is based on performing very good unfeasible insertions of customers to individual routes, followed by an ejection procedure in which the feasibility is recovered by ejecting some other customers from the unfeasible routes. The algorithm equals the best known cumulative number of vehicles (CVN) of 405 on the Solomon's instances.

An improved algorithm presented in [11] further employs a specific local search strategy guiding the ejections. Also, a feasible insertion mechanism denoted as *squeeze* as well as a search diversification *perturb* procedure are employed throughout the solving process boosting the algorithm's convergence. The algorithm provides for the contemporary best known CVN of 10290 over the extended Homberger's benchmark set.

An agent based algorithm for VRPTW and PDPTW is presented in [6] based on the Task Agent — Allocation Agent — Vehicle Agent hierarchy. The tasks are allocated to a fleet of the Vehicle Agents in a series of auction steps based on the well known contract net protocol (CNP). A set of improvement methods is introduced based on agent negotiation, that can be executed either *dynamically* after each auction step or *finally* after all tasks have been allocated. Several initial task orderings are introduced, providing for a number of alternative *particular solvers*. These solvers are then run in parallel with the best result being returned. The algorithm provides for a CVN of 10889 over the Homberger's extended benchmark set. However, as already shown in [14] and [5], a cost structure targeting directly the temporal aspects of the problem has been proved superior to the used travel time savings heuristic. Also the set of competing particular solvers

[1] Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague, peta.kalina@gmail.com

[2] Agent Technology Center, Faculty of Electrical Engineering, Czech Technical University in Prague, jiri.vokrinek@fel.cvut.cz

is static accross all instances, resulting in a significant performance overhead.

The algorithm presented in [5] is based on similar concepts. However, it introduces a cost structure corresponding to the *slackness savings heuristic* similar to [10] as well as a set of more refined improvement methods. The algorithm is run using three initial task orderings with the best result being considered. The algorithm provides for a contemporary best known results achieved by an agent based algorithm for both the Solomon's and the extended Homerger's benchmarks with a respective CVN of 429 and 10609. However, the traversed search space is quite narrow due to the static choice of orderings and algorithm configurations. The above mentioned performance overhead is not addressed as well.

To our knowledge, the only other agent based algorithm with results comparable to those presented within this work is presented by [8] achieving a CVN of 436 over the Solomon's benchmark set. The remaining relevant studies [3, 7, 2] focus on real-world derived scenarios using ad-hoc problem sets and therefore do not provide comparable performance information.

3 Negotiation Based Allocation Algorithm

The main contribution of this work is (i) the extension of concepts presented in [5] and [6] by introducing an improved parallel algorithm based on agent negotiation with an efficient search diversification and pruning strategy and (ii) the assessment of its relevance with respect to the state-of-the-art centralized as well as previously presented agent based algorithms.

3.1 Core Agent Framework

The algorithm is based on a three layer basic architecture appearing also in [5] 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 (private) Vehicle Agent's plan processing.

Characteristic to the agent decomposition is the clear distinction between Vehicle Agents' local planning and the global planning managed by the Allocation Agent. This allows for a transparent inclusion of the typical real-world concepts such as loading constraints, heterogenous cost structures for individual vehicles or a more complex commitment semantics.

3.2 Improved Agent Negotiation Process

At the core of the algorithm is the negotiation carried out between the Allocation Agent and the fleet of Vehicle Agents. The improved negotiation process is illustrated by Figure 1.

`Input` : Set of problem instance tasks T
`Output` : Improving sequence of solutions

Procedure $negotiate(T)$
begin
1: $\sigma_{best} := null$;
2: **foreach** (*setting* $S \in$ diversified algorithm settings set Δ)
3: **foreach parallel** (*ordering* $O \in$ diversified orderings set Ω)
4: $vn := (\sigma_{best} = null \; ? \; CLB(T) : vn(\sigma_{best}) - 1)$
5: **repeat**
6: $T_O :=$ apply O on T;
7: $\sigma_O :=$ *initial fleet* of vn empty vehicles
8: **foreach** task $t \in T_O$
9: $v := auction(t, \sigma_O)$;
10: $\sigma_O := commit(t, v)$;
11: $\sigma_O := dynamicImprove(S, \sigma_O, T_O)$;
12: **end foreach**
13: $\sigma_O := finalImprove(S, \sigma_O, T_O)$;
14: **if** $(feasible(\sigma_O))$
15: **if** $(\sigma_{best} = null$ **or** $vn(\sigma) < vn(\sigma_{best}))$
16: $\sigma_{best} := \sigma_O$;
17: $output(\sigma_O)$;
18: **end if**
19: $vn := vn(\sigma_{best}) - 1$;
20: **else**
21: **if** $(\sigma_{best} = null$ **or** $vn(\sigma) < vn(\sigma_{best}) - 1)$
22: $vn := vn + 1$;
23: **else**
24: store σ_O for orderings pruning
25: **break repeat**;
26: **end if**
27: **end if**
28: **end repeat**
29: **end foreach parallel**
30: $prune(\Omega)$;
31: **end foreach**
end

Figure 1. Improved agent negotiation process

The process begins with resetting the temporarily best found solution σ_{best} (line 1). Follows the selection of a particular algorithm configuration from a set of increasingly complex algorithm configurations Δ (line 2). A particular algorithm configuration $C \in \Delta$ is given by specifying the particular improvement methods to be used for the *dynamic* and *final* improvement steps at lines 11 and 13. There are three improvement methods defined (e.g. the $ReallocateAll$) that are refinements of the similar methods presented in [5] providing for an efficient search diversification on the algorithm configuration level, described in detail in Section 4.1.

Follows the selection of the particular instance task processing ordering (line 3). Based on previous findings [5] the agent based negotiation mechanism provides a very good convergence given a fitting task ordering is provided. The set of orderings Ω is obtained by two specific ordering diversification operators described in detail in Section 4.2 applied to a set of canonical analytically sound orderings, providing for the search diversification on the task ordering level.

Lines 3–29 outline the internal task allocation loop, denoted also

as a particular *algorithm instance* parameterized by $C \in \Delta$ and $O \in \Omega$. All algorithm instances corresponding to the currently processed algorithm configuration $C \in \Delta$ are processed in parallel, providing for a set of solutions $\sigma_O, O \in \Omega$. After these algorithm instances have finished, the results are used to prune the set Ω in an effort to effectively direct the search in the most feasible direction.

Thus for each algorithm instance an initial empty partial solution σ_O is instantiated, corresponding to a fleet of empty vehicles (line 7). Individual tasks are allocated to σ_O in a series auction steps (lines 9,10), followed by a particular dynamic improvement step (line 11), with the final improvement step being applied after all tasks have been processed (line 13). As the task allocation process is vastly similar to the one presented in [5], for the sake of clarity we present only a simplified version, neglecting some of the finer details not relevant for the contribution presented within this work.

In a departure from the previous works [6, 5], the initial size of the fleet (number of vehicles - VN) is always determined to target a new best found solution with respect to the current best solution σ_{best}. In both above mentioned works the individual algorithm instances were initialized with a VN corresponding to the theoretical lower bound count on the number of vehicles (LBVN). In case the resulting solution was not feasible (e.g. contained uncovered customers), the instance was restarted with a VN incremented by 1 until a feasible solution was found. There are several drawbacks to this strategy.

The discrepancy between the LBVN and the VN eventually found by a particular algorithm instance has a multiplicative effect on the complexity of the resulting negotiation process due to the above mentioned restarts strategy. Considering an instance using an unfitting ordering a higher number of restarts is necessary before a feasible solution is found. Even more intriguingly, such a solution is likely to be superseded by a solution provided by an algorithm instance with a more fitting ordering. Thus the majority of the processing time is actually spent by constructing solutions that will be most likely discarded and thus wasted.

Also, by starting from the LBVN, *all* competing algorithm instances are bound to be restarted at least a fixed number of times corresponding to the difference between the VN of the best eventually found solution and the LBVN. Thus the complexity of the resulting algorithm depends heavily on the nature of the instance being solved, as the interdigitation of the underlying multiple bin packing problem and the time windows constrains may require a much greater VN than the LBVN. Lastly, the determination of the LBVN is actually a NP-hard problem in itself [10]. The algorithms presented in [6] and [5] use therefore a greedy $O(N^3)$ approximation, requiring further processing time.

Thus, as already mentioned, an alternative restart strategy is introduced, with the initial VN being set to improve on the best found solution to date σ_{best}. In case such a solution is not available (e.g. during the initial stages of the solving process), an alternative capacity based LBVN (CLB) is used, computed based on the vehicle capacity and the cumulative demand of all customers [6] that can be computed in $O(1)$ time (line 4).

After the task allocation has finished (line 14), depending on the feasibility of the resulting solution σ_O and the current σ_{best} (which may differ from the initial value from the beginning of the instance run), the instance can be (i) pruned (line 25), (ii) restarted with an increased initial VN (line 22) or (iii) restarted with a new best found targeting VN (line 19).

The first situation occurs when σ is unfeasible with a VN higher than the contemporary $vn(\sigma_{best}) - 1$. In such a case a restart with an increased fleet size $vn + 1$ cannot potentially yield a result superior to σ_{best} and is therefore redundant. In case the current σ_{best} has a higher VN or has not yet been set, the restart is feasible, corresponding to the second above mentioned situation. On the other hand when a feasible solution σ_O is found the process can be restarted with a VN targeting a new best found solution. In case σ also represents the new contemporary best found result, it is stored and output (lines 16, 17).

After all algorithm instances have finished and before a new algorithm configuration $C \in \Delta$ is processed, the set of orderings Ω is pruned for the next phase, based upon the solutions $\sigma_O, O \in \Omega$ stored during the previous phase (line 30). Two alternative ordering pruning strategies are introduced in Section 4.2.2.

Thus the search diversification is achieved by using diversified sets Δ and Ω with the resulting performance penalty being partially offset by introducing an *overall pruning strategy* consisting of (i) an ordering pruning strategy limiting the number of algorithm instances run for the more complex algorithm configurations and (ii) a strategy allowing for an efficient management of the overall solving process based on reusing the already achieved results to effectively limit (prune) the number of individual algorithm instance restarts.

4 Search Diversification and Pruning Strategy

As mentioned above, the broader search space coverage of the presented algorithm is achieved by the (i) diversification on the algorithm configuration level by means of fine grained control over the improvement methods and (ii) diversification on the instance task ordering level by means of two specific ordering diversification operators.

4.1 Algorithm Configuration Diversification

The three improvement methods used for the dynamic and final improvement steps are the *ReallocateAll* method, the *ε-ReallocateWorst* method and the *ε-ReallocateRandom* method. Their basic semantics is carried over from [5]. The methods are based on iterating through portions of each agent's plan and *reallocating* corresponding customers in a series of auction steps to agents with a better commitment cost estimates, thus improving the partial solution σ_O under construction. The methods differ by the specific order in which the customers in an individual agent's plan are processed. For purposes of this work these methods were further improved by introducing the *loop count* parameter, affecting the number of repetitions of the above mentioned process on a single agent's plan. This allows for a configurable tradeoff between the method's complexity and its relative strength.

The algorithm configurations basic notation is carried over from [5], with *B* (*Basic*) denoting a configuration with neither improvement step applied, *FI* (*Final Improvement*) denoting a configuration with only a $ReallocateAll$ based final improvement step being applied and *DI* denoting configurations extending the *FI* configuration with a particular dynamic improvement step being applied as well.

The set Δ thus comprises of a *B* configuration, a *FI* configuration and three subsets of *DI* configurations each corresponding to a particular method being used for the dynamic improvement step. These subsets are further diversified by using an increasing *loop count* parameter setting.

4.2 Ordering Diversification and Pruning

As mentioned above, the initial set of orderings Ω is generated from the set of analytically sound orderings using two specific operators.

Table 1. Performance of the algorithm compared to the best known results — CVN and relative error

Type	SotA [9, 11]	Agents [8]	Agents [6]	Agents [5]	*BP*	*CSP*	*BP+CSP*	$\Delta \times \Omega$
All	10695	–	–	+343 (3.2%)	+290 (2.7%)	+258 (2.4%)	+254 (2.4%)	–
100	405	+31 (7.7%)	–	+24 (5.9%)	+17 (4.2%)	+16 (4.0%)	+16 (4.0%)	–
200	694	–	–	+21 (3.0%)	+18 (2.6%)	+12 (1.7%)	+12 (1.7%)	–
400	1380	–	–	+38 (2.8%)	+35 (2.6%)	+31 (2.2%)	+29 (2.1%)	+29 (2.1%)
600	2065	–	–	+56 (2.7%)	+51 (2.5%)	+43 (2.0%)	+43 (2.1%)	+41 (2.0%)
800	2734	–	–	+89 (3.3%)	+76 (2.8%)	+72 (2.6%)	+71 (2.6%)	+70 (2.6%)
1000	3417	–	–	+115 (3.4%)	+92 (2.7%)	+84 (2.5%)	+83 (2.4%)	+82 (2.4%)
200–1000	10290	–	+609 (3.1%)	+319 (3.1%)	+273 (2.7%)	+242 (2.4%)	+238 (2.3%)	–

The canonical set contains the three orderings introduced in [5] with two additional orderings — the *larliest expiry first* (LEF) based on the *latest service start* value of individual tasks' time windows and the *Most Distant First* (MDiF) based on the distance of individual customers from the depot.

4.2.1 Ordering diversification operators

The two presented operators were introduced to provide means of *diversification* by providing for a diversified set Ω of initial instance task orderings. The *k-perturb(O)* operator is based on randomizing the order of sub-sequences of length k on the underlying set of tasks ordered by O. The *k-mixin(O_1, O_2)* operator combines two orderings by applying the secondary ordering O_2 to sequences of k tasks on the underlying set of tasks ordered by O_1.

As opposed to the well known *ordering crossover* and *mutation* operators used by the genetic ordering based algorithms [12], these two operators were specifically tailored to allow for traversing a neighborhood that is very close to the original analytically sound orderings and thus preserve the *nature* of these orderings. This effort was motivated by the previous findings [5], as it was proved that the analytically sound orderings significantly outperform randomly generated orderings. However, the rigorous assessment of suitability of known crossover operators or eventual applicability of a genetic based approach to identify the most fitting ordering is not part of this study and therefore remains an interesting area of future research.

4.2.2 Ordering pruning strategies

As outlined by the main negotiation process, after a particular algorithm configuration has been processed, the set of orderings is pruned based on the achieved results. Due to the fact that some results correspond to unfeasible partial solutions, instead of using the VN as the indicator of the resulting quality, we introduced the *average route size* metrics, corresponding to the average number of customers in a single route, providing for an ordering of the set of (partial) solutions σ_O. Two fundamentally different pruning strategies were introduced.

The *Basic Pruning Strategy* (*BP*) is based on the assumption that the optimal ordering may differ significantly for each problem instance. Thus a given number of the best (with longest average routes) orderings to be kept for each processed algorithm configuration is specified and the set Ω can be effectively pruned based on the results from the faster configurations in an effort to provide the most complex configurations with the only best fitting orderings.

An alternative strategy denoted as the *Minimal Covering Set Pruning Strategy* (*CSP*) is based on the opposite assumption — that the set of orderings and their interesting neighborhoods is limited and partially static across all problem instances. Thus using only a specified subset of problem instances (in our case the 100 and 200 customer instances from the Solomon's and Homberger's benchmark set) the whole space of orderings and algorithm configurations $\Delta \times \Omega$ is traversed. A minimal covering set of orderings Ω_C is identified. These orderings are then used across the whole solving process. Obviously the success of such a strategy is based on the fact that the processed instances are similar in their nature as the training set, however we argue that real world scenarios often display such a characteristic and therefore such an approach is an interesting alternative way of pruning the set of orderings.

A hybrid strategy denoted as *CSP+BP* combines the two by applying the *BP* pruning while treating the orderings $O \in \Omega_C$ as *unprunable*.

5 Experimental Evaluation

The experiments were carried out using the established Solomon's and the extended Homberger's benchmark sets [14, 4], sharing the same basic attributes. Thus the complete set of 356 instances contains instances of 6 different sizes with 100, 200, 400, 600, 800 and 1000 customers respectively, with 60 instances in each set (except the Solomon's with 56 instances). For each set 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 [14]. The inclusion of the extended Homberger's benchmarks provides for a relevant comparison with the established state-of-the-art centralized solvers that has been missing from most previous agent-based studies.

The algorithm prototype is implemented in JAVA programming language. The experiments were carried out using a 8G RAM AMD Athlon 2G Gentoo system running the 64-bit Sun JRE 1.6.0_22.

Four overall configurations were considered denoted as *BP*, *CSP*, *CSP+BP* and $\Delta \times \Omega$ corresponding respectively to the three previously defined pruning strategies being applied and a configuration with no ordering pruning employed thus traversing the whole diversified search space.

5.1 Overall Quality Analysis

The results summarizing the overall achieved solution quality are presented by Table 1. The *SotA [9, 11]*, *Agents [8]*, *Agents [6]* and the *Agents [5]* columns correspond to the state-of-the-art results for established centralized [9, 11] and the three previously presented comparable agent based algorithms [8, 6, 5] respectively. The remaining columns correspond to the four respective algorithm overall configurations. The presented results correspond the the *cumulative number of vehicles* (CVN) for the respective problem instance subsets, or the respective absolute and relative errors.

In overall the algorithm in it's full fledged *CSP+BP* configuration achieved a CVN of 10949 over all the benchmark instances, corresponding to a 2.4% relative error in terms of the primary optimization criteria with respect to the state-of-the-art centralized algorithms, equalling the best known results for 64% of the problem instances. With respect to previous agent based approaches this represents a new best known overall result, with the new best known result being achieved for 81 instances (23% of the cases).

The results thus strongly suggest that the presented diversification strategy is successful in terms of enabling the algorithm to traverse interesting areas of the search space resulting in a significantly improved solution quality.

5.2 Orderings Analysis

The experimental results presented within the Table 1 outline the respective success of the three alternative pruning strategies presented and the baseline $\Delta \times \Omega$ strategy without ordering pruning. The used set Ω corresponds to the 5 canonical orderings, extended by their *3-perturb* and *6-perturb* mutations and by their *10-mixin* and *20-mixin* combinations, providing for a set Ω of 65 instance task orderings.

The Ω_C corresponds to the minimal covering set of winning orderings from the $\Delta \times \Omega$ strategy being run over the Solomon's 100 and Homberger's 200 customer instances. Thus Ω_C contains 10 orderings, including the LEF canonical ordering attributing for the majority of wins, 4 orderings based on the *k-perturb* operator and 5 *k-mixin* based orderings.

The pruning strategy was set to retain 20 orderings based on the results from the first two *B* and *FI* algorithm configurations followed by a *minimal DI* configuration with the $ReallocateAll$ dynamic improvement method and a *loop count* $= 1$, upon which the set of orderings was pruned to mere two members processed by the rest of the algorithm configurations, corresponding to three subsets of configurations based on the three individual improvement methods being used for the dynamic improvement step with the *loop count* parameter being set to 3 and 6 respectively.

5.2.1 Diversification Operators

In overall the results indicate that the search diversification provided by the two introduced operators applied on the extended set of analytically sound orderings significantly improves the solution quality achieved by the solver. An additional experiment was carried out to determine a fitting parametrization of the mentioned ordering diversification operators, consisting of analyzing the corresponding Ω_C for the unrestricted $\Delta \times \Omega$ setting using a wider set of orderings generated by setting the k parameter to $k = 2, 4, 8, 16, 32, 64$ for both operators.

In case of the *k-mixin* operator the setting of the k parameter does not affect the success of the resulting ordering greatly, with the overall success roughly corresponding to the success of the two underlying orderings. However, with the *k-perturb* operator the lower k values dominate the higher values. These results suggest that (i) the *k-mixin* operator preserves the analytically derived soundness of the orderings irrespective of the k parameter settings providing for a flexible search diversification operation while (ii) the *k-perturb* operator diverges from the feasible analytically sound orderings with increasing k.

In terms of overall success the *k-mixin* operator slightly outperforms the *k-perturb* operator, however their simultaneous appearance in the identified set Ω_C mentioned in the previous section suggests that both provide for an effective alternative means of search diversification, given a fitting k is used for the *k-perturb* operator.

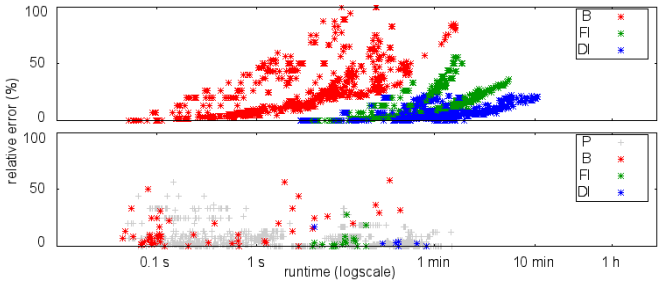

Figure 2. Individual algorithm instance results and runtimes with (bottom) and without (top) overall pruning strategy

5.3 Pruning Strategies

The results presented in Table 1 outline the success of the individual pruning strategies.

The *BP* strategy posted results that are clearly an improvement over the previous results based on a similar allocation process. With only the two best orderings being processed by the full blown solvers in the latter stages of the process the results suggest that the ordering pruning based on results of increasingly more complex algorithm configurations is quite successful. However, the convergence to the optimal ordering is not straightforward and — as outlined by the significantly superior results posted by the *CSP* strategy — often fails to identify the optimal ordering from the set Ω. We argue that this is due to two factors: (i) the metrics of a particular ordering success based on the average route size of the underlying (partial) solution is a very loose one, with many orderings achieving identical results and (ii) the relationship between relative success of two different algorithm configurations using the same instance task ordering is not straightforward, as the dynamic improvement methods process the tasks in the order as they appear in the corresponding agent's plan and thus the two solving processes based on an identical initial ordering can diverge significantly.

The very good results achieved by the *CSP* strategy suggests, that this strategy is actually very successful in identifying particularly good orderings or — in case of orderings produced by a particular ordering diversification operation — suggesting valuable neighborhoods of particular orderings. Considering the relatively small overall difference in achieved quality between the *CSP*, *CSP+BP* and the full $\Delta \times \Omega$ strategies the results suggest that the set of dominant orderings is relatively small and consistent over the whole benchmark set, further supporting the soundness of the *CSP* pruning strategy.

5.4 Runtime and Convergence analysis

The results presented within this study are based on an extended search based on diversified sets Δ and Ω inherently increasing the complexity of the overall process. To offset the increased complexity an *overall pruning strategy* is introduced based on ordering pruning and the improved negotiation process.

Figure 2 illustrates the improvements in runtime achieved by the introduced overall pruning strategy, corresponding to a subset of 16 problem instances from the 1000 customers instance set. The two

Table 2. Comparison with the state-of-the-art solvers in terms of runtime and convergence

	Nagata [11]	Lim [9]	*CSP+BP*	
Size	Avg. RT	Avg. RT	Avg. RT	Anytime RT
200	1 min	10 min	10 s	57 ms
400	1 min	20 min	2 min	300 ms
600	1 min	30 min	8 min	2 s
800	1 min	40 min	24 min	7 s
1000	1 min	50 min	54 min	14 s

compared settings correspond to (i) the *CSP+BP* strategy using the improved agent negotiation process outlined by Figure 1 (bottom) and (ii) the $\Delta \times \Omega$ strategy using a process based on [5] thus not employing any of the presented improvements (top). Individual points in the respective graphs correspond to the relative errors and runtimes recorded for individual algorithm instances. The results are grouped based on the underlying overall algorithm configuration with the pruned results being denoted as *P*.

The results suggest that the pruning strategies have a dramatic positive impact on the resulting runtime features of the algorithm. While in case of the original algorithm the biggest runtime penalty was implied by the least successful algorithm instances requiring numerous restarts with an incremented initial VN before a feasible solution is found, these instances are effectively pruned after only one restart in case of the presented algorithm. Furthermore by pruning the set of orderings the number of instantiated algorithm instances drops significantly with their increasing complexity with minimal impact on the resulting solution quality, providing for yet another massive boost in the algorithm's efficiency. Thus the recorded average composite single threaded time per processed instance (the sum of runtimes for all algorithm instances) is 41 minutes with the overall pruning in place, representing a massive 6.3 times improvement over the configuration not using the pruning strategy with 258 min.

The comparison in terms of runtime of the full fledged *CSP+BP* setting with the state-of-the-art algorithms is presented by Table 2. The listed values correspond to the average composite single threaded runtime. To provide illustration of the algorithm's anytime features the parallel runtime before the best solution is found is listed as well. The results confirm exceptional anytime features of the algorithm when its inherent parallelization features are fully exploited with the time before the best solution is found significantly outperforming all previous solvers. The composite runtimes are also competitive, especially considering that the underlying prototypal implementation is far from being performance optimized. We must note, however, that: (i) compared algorithms outperform presented algorithm in terms of CVN and (ii) are not computationally bound. Therefore to be able to draw definitive conclusions, settings with similar solution quality would have to be compared.

6 CONCLUSION

Within this paper we introduce an improved parallel agent based algorithm for the widely studied VRPTW problem, built around similar concepts as the algorithm presented in [5]. The algorithm is based on the execution of increasingly complex algorithm configurations over a set of instance task orderings.

The main presented contribution is (i) the introduction of an efficient search diversification strategy based on generating a diversified set of orderings using two specific introduced ordering diversification operators and (ii) the presented overall pruning strategy based on three efficient ordering pruning strategies and a vastly improved negotiation process, offsetting the increase in the overall complexity of the algorithm inherent to the diversified search.

The relevance of the improved algorithm is assessed using two sets of widely used benchmark instances. When compared to the state-of-the-art centralized solvers, the algorithm achieves a relative error of 2.4% while equalling the best known results for 64% of the instances. This result also represents a significant improvement over all previously presented agent based algorithm, with 81 new best found solutions. Moreover, benefitting from its inherently parallel nature the algorithm boasts an excellent anytime characteristics, outperforming even the centralized algorithms in this respect.

Future interesting research opportunity was identified in the assessment of suitability of known ordering crossover operators and the eventual applicability of a genetic based approach to identify a fitting ordering for a particular problem instance.

ACKNOWLEDGEMENTS

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] Olli Bräysy and Michel Gendreau, 'Vehicle routing problem with time windows, part I and II', *Transportation Science*, **39**(1), 104–139, (2005).
[2] Zhenggang Dan, Linning Cai, and Li Zheng, 'Improved multi-agent system for the vehicle routing problem with time windows', *Tsinghua Science Technology*, **14**(3), 407–412, (2009).
[3] Klaus Fischer, Jrg P. Mller, and Markus Pischel, 'Cooperative transportation scheduling: an application domain for dai', *Journal of Applied Artificial Intelligence*, **10**, 1–33, (1995).
[4] J. Homberger and H. Gehring, 'A two-phase hybrid metaheuristic for the vehicle routing problem with time windows', *European Journal of Operational Research*, **162**(1), 220–238, (2005).
[5] Petr Kalina and Jiří Vokřínek, 'Algorithm for vehicle routing problem with time windows based on agent negotiation', in *Proceedings of the 7th Workshop on Agents In Traffic and Transportation, AAMAS (to appear)*, (2012).
[6] Petr Kalina and Jiří Vokřínek, 'Parallel solver for vehicle routing and pickup and delivery problems with time windows based on agent negotiation', in *IEEE International Conference on Systems, Man and Cybernetics (to appear)*, (2012).
[7] Robert Kohout and Kutluhan 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).
[8] Hon Wai Leong and Ming Liu, 'A multi-agent algorithm for vehicle routing problem with time window', in *Proceedings of the 2006 ACM symposium on Applied computing SAC 06*. ACM, (2006).
[9] A Lim and X Zhang, '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).
[10] Quan Lu and Maged M. Dessouky, '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).
[11] Yuichi Nagata and Olli Brysy, 'A powerful route minimization heuristic for the vehicle routing problem with time windows', *Operations Research Letters*, **37**(5), 333–338, (2009).
[12] P.W. Poon and J.N. Carter, 'Genetic algorithm crossover operators for ordering applications', *Computers and Operations Research*, **22**(1), 135 – 147, (1995).
[13] Sintef. Top — http://www.sintef.no/projectweb/top/problems/.
[14] Marius M. Solomon, 'Algorithms for the vehicle routing and scheduling problems with time window constraints', *Operations Research*, **35**, 254–265, (1987).