Journal of Heuristics, 10: 613–627, 2004

A Variable Neighborhood Search for the Multi Depot Vehicle Routing Problem with Time Windows

MICHAEL POLACEK, RICHARD F. HARTL AND KARL DOERNER
Department of Production and Operations Management, Institute of Management Science, University of Vienna, A-1210 Vienna, Austria
email: michael.polacek@univie.ac.at
email: richard.hartl@univie.ac.at
email: karl.doerner@univie.ac.at

MARC REIMANN
Institute for Operations Research, ETH Zurich, CH-8092 Zurich, Switzerland
email: marc.reimann@ifor.math.ethz.ch

Submitted in August 2003 and accepted by Stefan Voss in October 2004 after 2 revisions

Abstract

The aim of this paper is to propose an algorithm based on the philosophy of the Variable Neighborhood Search (VNS) to solve Multi Depot Vehicle Routing Problems with Time Windows. The paper has two main contributions. First, from a technical point of view, it presents the first application of a VNS for this problem and several design issues of VNS algorithms are discussed. Second, from a problem oriented point of view the computational results show that the approach is competitive with an existing Tabu Search algorithm with respect to both solution quality and computation times.

Key Words: Metaheuristic, Variable Neighborhood Search, VNS, Multi Depot Vehicle Routing Problem with Time Windows, MDVRPTW

1. Introduction

Vehicle Routing Problems (VRP) are classical combinatorial optimization problems with considerable economic significance as pointed out by the following statement.

"The large number of real-world applications, both in North America and in Europe, have widely shown that the use of computerized procedures for the distribution process planning produces substantial savings (generally from 5 to 20%) in the global transportation costs. It is easy to see that the impact of these savings on the global economic system is significant. Indeed, the transportation process involves all stages of the production and distribution systems and represents a relevant component (generally from 10 to 20%) of the final cost of the goods."[1]

A brief description of the basic VRP can be given as follows: customers with known demands are serviced by a homogeneous fleet of vehicles with limited capacity. Routes are assumed to start and end at exactly one depot, each customer is fully served by exactly

one vehicle and the primary objective is to minimize the total distance travelled by all vehicles.

However, to satisfy real-life demands additional constraints are complicating the development of appropriate methods. For various applications, like bank deliveries, postal deliveries or school bus routing, a time interval is also associated with each customer to define the time of service. Moreover, large companies, as they can be found in the petroleum industry, have more than just one depot from which service stations are supplied. On account of this, multiple depots are a reasonable extension to the classical VRP.

The basic VRP and basically all of its variants belong to the class of NP-hard problems (c.f., Garey and Johnson (1979)) such that exact algorithms, as described in Kohl et al. (1999) and Laporte and Nobert (1987), become highly time consuming as soon as problem instances are increasing in size. Thus, for large scale problem instances as typically found in industrial applications finding an 'optimal' solution is not practicable. Therefore different types of heuristics, ranging from route construction over route improvement heuristics to sophisticated metaheuristics emerged which quickly produce solutions of reasonable quality.

The aim of this paper is to propose an algorithm based on the philosophy of the Variable Neighborhood Search (VNS), a metaheuristic, described in Hansen and Mladenović (1999) to solve Multi Depot Vehicle Routing Problems with Time Windows. The paper has two main contributions. First, from a technical point of view, it presents the first application of a VNS for this problem and several design issues of VNS algorithms are discussed. Second, from a problem oriented point of view the computational results show that the approach is competitive with the Tabu Search (TS) algorithm published in Cordeau, Laporte, and Mercier (2001), with respect to both solution quality and computation times.

The remainder of the paper is organized as follows. In the next section a problem description is given and related work is discussed. Section 3 reviews the main ideas underlying VNS and provides the details of the implementation and the design choices for the described problem. Computational results are presented and discussed in Section 4. In Section 5 we present results for an improved version of the algorithm that takes into account route duration considerations before Section 6 concludes the paper with a résumé of the applied approach and comments on possible directions for further research.

2. Problem description and related work

The problem tackled in this paper is the Multi Depot Vehicle Routing Problem with Time Windows (MDVRPTW). It is a generalization of the well known Vehicle Routing Problem with Time Windows (VRPTW) where instead of one depot, several depots with different locations and associated fleets have to be considered. Thus, the problem is defined on a complete graph $G = (V, A)$, where $V = \{v_1, \ldots, v_m, v_{m+1}, \ldots, v_{m+n}\}$ is the vertex set and $A = \{(v_i, v_j) : v_i, v_j \in V, i \neq j\}$ is the arc set. Vertices v_1 to v_m correspond to the m depots, while the vertices v_{m+1} to v_{m+n} represent the n customers. Each vertex $v_i \in V$ has several nonnegative weights associated to it, namely, a demand d_i, a service time s_i, as well as an earliest e_i and latest l_i possible start time for the service, which define a time window $[e_i, l_i]$. For the depots these time windows correspond to the opening hours.

Further, the depot vertices v_1 to v_m feature no demands and service times, i.e. $d_i = s_i = 0$, $\forall i \in \{v_1, \ldots, v_m\}$. Associated to each arc (v_i, v_j) is a nonnegative travel time or cost c_{ij}. Finally, a fleet of K vehicles is located at the m depots. Each vehicle k has associated a nonnegative capacity D_k and a nonnegative maximum route duration T_k. Note, that the distribution of vehicles over the depots is fixed a-priori and is given as input data.

Based on this graph, the MDVRPTW consists of building K vehicle routes such that each vehicle starts and ends at its home depot, each customer is served by one and only one vehicle, the total load and duration of vehicle k does not exceed D_k and T_k respectively, the service at each customer i begins within the associated time window $[e_i, l_i]$ and each vehicle route starts and ends within the time window of its depot. The objective is to minimize the total distance travelled by all vehicles.

As stated above, the MDVRPTW is a generalization of the VRPTW and is thus NP-hard. Hence, exact approaches are bound to be efficient for very small problem instances only and the use of heuristics should be justified. For the VRPTW this research direction has been followed intensively as can be seen from an abundant number of papers describing different approaches for the VRPTW, ranging from simple heuristic methods to very sophisticated metaheuristics of all kinds. Reviews of these works have been put together by Braysy and Gendreau (to appear). Also, the MDVRP, i.e. the version of the problem without time windows has been studied by several authors (c.f. e.g., Chao, Golden, and Wasil (1993), Cordeau, Gendreau, and Laporte (1997), and Renaud, Boctor, and Laporte (1996)).

To our best knowledge there is only one paper that addresses (among other problems) the MDVRPTW, namely the work of Cordeau et al. from 2001 (see Cordeau, Laporte, and Mercier (2001)) describing the application of an adapted TS algorithm, which had previously been applied to the MDVRP and the Periodic VRP. The main characteristics of this approach are the use of a very simple neighborhood, a move of one customer from one route to another, and the fact that infeasible solutions are allowed during the search. To ensure that the algorithm is able to move from infeasible to feasible parts of the search space an adaptive penalty function is used that reacts to the search history. Numerical results are presented for the Periodic VRPTW, the MDVRPTW and the VRPTW. While for the former two problems no benchmarks existed, the results for the VRPTW proved the quality of the approach as illustrated by a comparison with state of the art approaches on the classic Solomon instances.

3. VNS for the MDVRPTW

VNS is a recent metaheuristic for solving combinatorial and global optimization problems proposed by Mladenović and Hansen in 1997 (c.f. e.g., Hansen and Mladenović (1999, 2001) and Mladenović and Hansen (1997)). The basic idea is a systematic change of neighborhood within a local search. Here, several neighborhood structures are used instead of a single one, as it is generally the case in many local search implementations. Furthermore, the systematic change of neighborhood is applied during both a descent phase and an exploration phase, allowing to get out of local optima.

Initialization. Select the set of neighborhood structures $N_\kappa(\kappa = 1, \ldots, \kappa_{max})$, that will be used in the search; find an initial solution x; choose a stopping condition;
Repeat the following until the stopping condition is met:

1. Set $\kappa \leftarrow 1$;
2. Repeat the following steps until $\kappa = \kappa_{max}$:
   (a) *Shaking*. Generate a point x' at random from κ^{th} neighborhood of x $(x' \in N_\kappa(x))$;
   (b) *Local search*. Apply some local search method with x' as initial solution; denote with x'' the so obtained local optimum;
   (c) *Move or not*. If this local optimum x'' is better than the incumbent, move there $(x \leftarrow x'')$, and continue the search with N_1 $(\kappa \leftarrow 1)$; otherwise, set $\kappa \leftarrow \kappa + 1$;

Figure 1. Steps of the basic VNS (c.f. Hansen and Mladenović (2001)).

Differing from other local search based metaheuristics, VNS does not follow a trajectory but rather "explores increasingly distant neighborhoods of the current incumbent solution, and jumps from this solution to a new one if and only if an improvement has been made. In this way often favorable characteristics of the incumbent solution, e.g., that many variables are already at their optimal value, will be kept and used to obtain promising neighboring solutions. Moreover, a local search routine is applied repeatedly to get from these neighboring solutions to local optima."[2]

The steps of the basic VNS are shown in figure 1. Here, $N_\kappa(\kappa = 1, \ldots, \kappa_{\max})$ is a finite set of pre-selected neighborhood structures. The stopping condition may be, e.g., maximum CPU time allowed, maximum number of iterations or maximum number of iterations between two improvements.

The basic VNS consists of both a stochastic component, i.e., the randomized selection of a neighbor in the shaking phase, and a deterministic component, that is the application of a local search in each iteration. Finally, the solution obtained is compared to the incumbent one and will be accepted as new starting point if an improvement was made, otherwise it will be rejected. Thus, the procedure is a descent, first improvement method with randomization. However, as pointed out in Hansen and Mladenović (2001), it could be transformed into a descent-ascent method without much additional effort. Thereby x is also set to x'' with some probability, even if the solution is worse than the incumbent.

Below, the implementation of each part of the VNS to solve the MDVRPTW is described. The description comprehends the building of an initial solution, the shaking phase including the neighborhood structure definition with the necessary exchange operators, the local search method, and the acceptance decision in the *Move or not* phase.

3.1. Initial solution

To construct an initial solution for the MDVRPTW, each customer i is first assigned to the nearest depot. Afterwards, all customers within a depot are ordered by the center of their time window $\frac{1}{2}(e_i + l_i)$. If this is done, routes are constructed for each depot using the same simple procedure described in figure 2.

1. Set $k \leftarrow 1$.
2. Repeat until all customers are assigned to a route:
   (a) Sequentially select a pre-ordered customer $i \in \{1, \ldots, n\}$.
   (b) Insert customer i at the end of route k.
   (c) Set $k \leftarrow k + 1$.
   (d) if k exceeds the number of available vehicles m, set $k \leftarrow 1$.

Figure 2. Generation of an initial solution.

While the objective function value obtained by this method is of rather poor quality, the approach brings up two important benefits.

- short run times: results are achieved within a fraction of a second
- minor constraint violations, because of the time window ordering of customers and the fact that each vehicle is assigned approximately the same number of customers.

Since the initial solution is not necessarily feasible the following iterative process of the VNS needs to overcome this and to come up with a feasible solution. Note however, that finding a feasible initial solution is NP-complete given a fixed fleet size.

3.2. Shaking

The set of neighborhood structures used for shaking is the core of the VNS. The main difficulty is to find a balance between effectiveness and the chance to get out of local optima.

To define a neighborhood of the current solution an appropriate function or operator must be specified. The main issue is that the neighborhood operator should allow to sufficiently perturb the incumbent solution while still making sure that the new solution keeps important parts of the incumbent.

One of the best neighborhood structures for the VRPTW was proposed in Taillard et al. (1997) and is called CROSS-exchange. The main idea of this exchange is to take two segments of different routes and exchange them as illustrated in figure 3.

First, the two edges (X_1, X_1'), and (Y_1, Y_1') are removed from the first route while the edges (X_2, X_2'), and (Y_2, Y_2') are removed from the second route. Then the segments $X_1' - Y_1$ and $X_2' - Y_2$, which may contain an arbitrary number of customers, are swapped by introducing the new edges (X_1, X_2'), (Y_2, Y_1'), (X_2, X_1') and (Y_1, Y_2'). The orientation of both routes is preserved.

An extension to the CROSS-exchange operator is introduced in Braysy (2003). Here the sequences get inverted, i.e., the orientation of the selected route parts changes. Consequently, this operator is called inverted CROSS-exchange, iCROSS-exchange for short.

Both, CROSS-exchange and iCROSS-exchange operators are used to define a set of neighborhood structures for the VNS proposed in this paper. As mentioned above, the neighborhoods should explore increasingly distant solutions from the incumbent to overcome local and find global optimality. In this paper, two measures for the 'distance' between

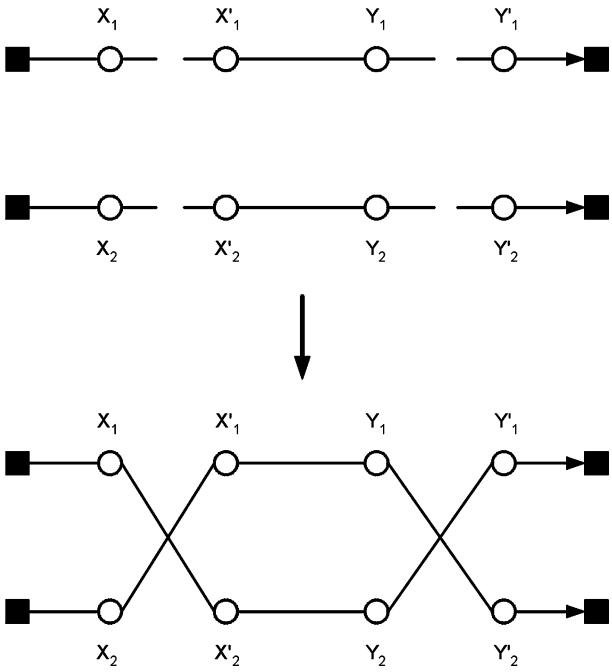

Figure 3. The CROSS-exchange operator.

two solutions are used. The first metric is simply the maximum length of the sequences that are exchanged. On top of that, the second metric deals with the number of depots involved in a neighborhood move.

More precisely, the set of neighborhood structures used in the VNS is divided into two parts. The first half considers only routes belonging to a given depot, whereas the second half selects routes from different depots. Let C_k denote the number of customers assigned to route k, then Table 1 shows for each neighborhood κ the number of depots involved, as well as the maximum sequence length considered. Clearly the maximum sequence length can not exceed C_k for any given route k.

Note, that the maximum sequence length just acts as an upper bound for the sequence length removed in a given neighborhood. Thus, a sequence of length 2 can be removed in the second, third, fourth, fifth and sixth neighborhood (and also in the 8th to 12th neighborhood), while a sequence of length four can only be removed in the neighborhoods 4–6 and 10–12. Thus, while in each neighborhood all the possible sequence lengths are equally likely to be chosen, overall there is a strong bias towards smaller sequence lengths to focus the search rather close to the incumbent solution. However, large changes may occur.

In addition, in each neighborhood the iCROSS-exchange operator is applied with a probability $1/\kappa_{\max}$ to further increase the extent of the perturbation. Finally, note that to allow for simply moving a set of customers from one route to another, which corresponds to the Or-opt operator, one of the sequences may have length 0. Without loss of generality we will assume that this is always the second sequence.

Table 1. Set of neighborhood structures.

κ	Depots	Max. sequence length
1	1	min (1, C_k)
2	1	min (2, C_k)
3	1	min (3, C_k)
4	1	min (4, C_k)
5	1	min (5, C_k)
6	1	C_k
7	2	min (1, C_k)
8	2	min (2, C_k)
9	2	min (3, C_k)
10	2	min (4, C_k)
11	2	min (5, C_k)
12	2	C_k

Note that the contribution of the different neighborhoods to the solution quality obtained by the VNS have also been tested. As was to be expected, the first neighborhoods have a much stronger success rate (i.e., lead to improved solutions much more often) than the more distant neighborhoods. In fact, while the first neighborhood has led to an improvement in solution quality 1 out of 3 times, the last neighborhood 'succeeds' with a probability of slightly more than 1 percent. While this seems to be a small rate, runs with a smaller number of neighborhoods have shown that the algorithm may easily get stuck in local optima if the number of neighborhoods is too small. Thus, these distant neighborhoods significantly improve the stability of the algorithm, in terms of the variance in solution quality observed over multiple runs.

3.3. Local search

A solution obtained through shaking is afterwards submitted to a local search to come up with a local optimal solution. In the current implementation this local search is a restricted version of 3-opt, with two characteristics. First, sequence inversion is not allowed and second, the length of the sequences to be exchanged is bounded by an upper limit of three. While the first mechanism is aimed at reducing the risk of infeasibility (which might be caused by time window violations due to an inversion), the second mechanism improves runtimes by reducing the size of the neighborhood significantly.

Another important issue related to algorithm effectiveness is that the local search is applied only on a route basis. Thus, after each shaking only the two routes that have changed need to be re-optimized.

Finally, as is commonly done, the local search restarts instantly after an improving move was found. This leads to shorter run times and better results by a constant number of iterations.

3.4. Acceptance decision

After the shaking and the local search procedures have been performed, the solution thus obtained has to be compared to the incumbent solution to be able to decide whether or not to accept it. In this context two important interrelated issues arise.

First, the evaluation function for solutions has to be specified, particularly if infeasible solutions are allowed. The evaluation function used in the context of the VNS proposed in this paper follows the approach proposed in Cordeau, Laporte, and Mercier (2001) which can be described as follows:

For a solution $s \in S$, let $c(s)$ denote the total travel time of the routes, and let $d(s)$, $t(s)$ and $w(s)$ denote the total violation of load, duration and time window constraints, respectively. Further, let a_i be the arrival time at customer i. It is assumed that an arrival time $a_i > l_i$ leads to an infeasible solution, while early arrival $a_i < e_i$ is allowed but leads to a waiting time. The total violation of load and duration constraints is computed on a route basis with respect to the values D_k and T_k, whereas the total violation of time window constraints is equal to $\sum_{i=1}^{n}(a_i - l_i)^+$ where $x^+ = \max\{0, x\}$. Solutions are then evaluated using a cost function $f(s) = c(s) + \alpha q(s) + \beta d(s) + \gamma w(s)$, where α, β and γ are positive weights. By dynamically adjusting the values of the three parameters, this relaxation mechanism facilitates the exploration of the search space and is particularly useful for tightly constrained instances.

Because VNS does not follow a trajectory it is much more difficult to implement an evaluation function in a reasonable way. Some initial experiments have shown that due to the shaking phase very little control over the constraint violations can be exerted such that the weights α, β and γ vary quite rapidly. While the addition of upper and lower bounds for these parameters alleviated this problem to some extent, the best results were achieved with fixed weights equal to 100. Thus, in the experiments reported below this setting was chosen.

Moreover, resulting from the strong bias towards feasible solutions induced by the evaluation function, many (infeasible) neighborhood moves are rejected and it is difficult to leave a local optimum. Thus, we use a modified acceptance decision which is based on Threshold Accepting ideas (c.f. Dueck and Scheuer (1990)). A move yielding an improvement is always accepted, while non-improving moves are accepted up to a fixed threshold. This threshold depends on the quality of the solution as well as on the number of iterations needed to reach the first feasible solution in a run.

Another method to overcome local optima is an extension of the basic VNS called Skewed VNS (c.f. Hansen and Mladenović (2001)). In this approach a solution is not only evaluated by its objective value but also by its distance to the incumbent solution, favoring more distant solutions. However, in the MDVRPTW considered in this paper the objective function is already skewed by the penalty parameters and the use of an additional distance parameter will impair the objective function even further and is not promising.

4. Computational results

In this section the VNS implementation proposed above will be analyzed and its performance will be compared to the results of the TS specified in Cordeau, Laporte, and Mercier (2001). The VNS was implemented in C++ and experiments were performed on a Pentium 3 with

Table 2. Comparison of instances and results.

No.	n	m	t	T	D	TS	VNS	RPD (%)
01	48	4	2	500	200	1083.98	1083.98	0.00
02	96	4	3	480	195	1763.07	1762.21	−0.05
03	144	4	4	460	150	2408.42	2374.36	−1.41
04	192	4	5	440	185	2958.23	2858.20	−3.38
05	240	4	6	420	180	3134.04	3040.84	−2.97
06	288	4	7	400	175	3904.07	3758.36	−3.73
07	72	6	2	500	200	1423.35	1422.65	−0.05
08	144	6	3	475	190	2150.22	2103.89	−2.15
09	216	6	4	450	180	2833.80	2783.95	−1.76
10	288	6	5	425	170	3717.22	3577.28	−3.76
11	48	4	1	500	200	1031.49	1005.73	−2.50
12	96	4	2	480	195	1500.48	1487.64	−0.86
13	144	4	3	460	150	2020.58	2014.02	−0.32
14	192	4	4	440	185	2247.72	2221.17	−1.18
15	240	4	5	420	180	2509.75	2494.82	−0.59
16	288	4	6	400	175	2943.90	2939.20	−0.16
17	72	6	1	500	200	1250.09	1239.13	−0.88
18	144	6	2	475	190	1809.35	1796.21	−0.73
19	216	6	3	450	180	2310.92	2318.59	0.33
20	288	6	4	425	170	3131.90	3079.73	−1.67
Avg.						2306.63	2268.10	−1.40

933 MHz using double precision floating point arithmetic. All VNS results presented below are based on runs over 10^8 iterations.

The problem instances used for the analysis originate from Cordeau, Laporte, and Mercier (2001) and are available on the internet at http://www.hec.ca/chairedistributique/data. The data set consists of 20 instances which differ with respect to their size as well as their time window tightness. Table 2 provides some information about the problem instances.

More precisely, the columns headed with n, m and t are representing the number of customers, the number of depots and the number of available vehicles at each depot, respectively. Instances from 11 to 20 have larger time windows and therefore a smaller number of vehicles is defined because it is easier for the same vehicle to service several customers. All instances consider a homogeneous fleet where each vehicle has a maximal route duration T and a capacity D.

Finally, Table 2 shows a first comparison between the TS and the VNS in the last three columns. More specifically, the column TS shows the best results reported in Cordeau, Laporte, and Mercier (2001) to be found during all the experiments. Analogously, the column VNS provides this information about the algorithm proposed in this paper. The

Table 3. Comparison of average results for VNS and TS after equivalent runtimes.

No.	Time (min)	TS 1 run	VNS avg. 10 runs
01	55	1083.98	1083.98
02	186	1763.84	1768.34
03	277	2408.42	2388.50
04	408	2976.11	2896.25
05	557	3165.69	3075.19*
06	769	3904.07	3816.14*
07	101	1423.35	1423.29
08	278	2163.32	2112.45
09	483	2833.80	2799.92*
10	703	3717.22	3603.18*
11	91	1031.49	1037.30
12	221	1506.08	1507.79
13	345	2020.58	2030.45
14	500	2247.72	2258.61
15	643	2509.75	2523.42*
16	829	2948.01	2951.31*
17	167	1250.09	1291.17
18	354	1823.16	1828.50
19	778	2310.92	2334.51*
20	824	3137.27	3115.97*
Avg.	428	2311.24	2292.31
		1.63%	1.17%

last column shows the relative percentage deviation (RPD) between the results for each instance. Finally, the last row shows averages over the 20 instances.

The results in Table 2 clearly show the strong performance of the VNS. For 18 out of the 20 instances a new best solution was found, for one instance the VNS and TS found identical solutions. Only for 1 instance does TS find a better solution than VNS.

To better understand these results it is interesting to evaluate the average performance of the VNS with a particular, good parameter setting. Thus, the VNS was run 10 times on each instance and all results presented below are averages over these 10 runs.

To get a more fair comparison between the VNS and the TS an analysis of the respective solution qualities after equivalent runtimes is presented in Table 3. More precisely, these results were obtained in the following way. First, the results for the runtimes and solution quality of the TS were taken from Cordeau, Laporte, and Mercier (2001). These results correspond to a single pass of the TS for 10^6 iterations for each instance on a Sun Ultra 2 with 300 MHz. According to Dongarra (2003) this machine is approximately 12% slower

than the Pentium used for the experiments with the VNS, such that the admissible runtimes for the VNS were computed by using a conversion factor of 0.89.

Thus, Table 3 shows for each instance the original TS runtimes in minutes, as well as the results of the TS (1 run) and the VNS (averages over 10 runs). The last two rows of Table 3 indicate the average results of the two algorithms in absolute numbers and as a percentage deviation from the best known results.

Finally, note that some of the results for the VNS are marked with an asterisk. For these instances, which are the largest ones, 10^8 iterations of the VNS took less time than would have been admissible related to the TS. For these instances the numbers presented correspond to the average results at the end of the VNS runs.

Given these remarks, Table 3 clearly shows that the VNS obtains better results than the TS after equivalent runtimes. Overall, the difference between the results of the two algorithms is approximately 0.5%. This effect seems to be magnified as the problem instance size increases.

To get a clearer picture about this effect the problem instances were divided into three clusters which were built based on the numbers of customers in the instances. More precisely, problems with at most 100 customers, those with 101 to 200 customers and those with more than 200 customers were grouped together.

For these clusters Table 4 presents the following information. First, the average relative percentage deviation from the best known results is given for both the TS and the VNS for equivalent runtimes (as in Table 3). Second the total time needed for 10^6 iterations of the TS and 10^8 iterations of the VNS is shown.

Table 4 reveals two results. First, for small instances the TS finds good results very fast, such that for the smallest group of problems it outperforms the VNS after a given runtime. However, as problem size increases this effect is reversed and particularly for the largest instances the VNS clearly dominates the TS. Second, looking at the total runtimes it is obvious that the computational burden for the TS increases very fast as problem size grows. On the other hand, there is only a very moderate increase in runtimes for the VNS over the three clusters. Thus, the VNS scales very well with problem size. While for the smallest instances runtimes are very similar for the TS and the VNS, the VNS is about 4 times faster for the largest instances. The reason for this difference lies in the structure of the VNS. More specifically, the complexity of the VNS depends mainly on the number of customers per route as the embedded Local Search works on a route basis. Looking at the problem instances, it becomes obvious that for the instances 1 to 10 the average number of customers per route is 8.5, while for the instances 11 to 20 it is 12, regardless of the actual problem size.

Table 4. VNS and TS computation times for different problem sizes.

	RPD		Time (min)	
Problem size	TS (%)	VNS (%)	TS (10^6 it.)	VNS (10^8 it.)
1–100 customers	0.80	1.51	137	118
101–200 customers	1.90	1.11	360	152
201–300 customers	2.06	0.97	698	161

Concluding this comparison, the VNS outperforms the TS with respect to both the best solutions found and the runtimes but more importantly it shows a much better scaling such that particularly for large real-world instances it can be expected to perform much better than the TS.

5. Performance improvements

In the last section we were able to show that our VNS consistently produces solutions competitive with a well known TS algorithm.

As pointed out in Cordeau, Laporte, and Mercier (2004) the results presented above could however be improved by taking the route duration constraints into account during the search. The problem instances for the MDVRPTW are characterized by the fact that the maximum route duration is significantly shorter than the planning horizon as given by the depot opening hours. Thus, making sure that the vehicles have little idle times can be crucial for solution feasibility and as a consequence improving solutions that were deemed infeasible before may be accepted.

Table 5. Comparison of improved results.

No.	TS old	VNS old	RPD (%)	TS new	VNS new	RPD (%)
01	1083.98	1083.98	0.00	1074.12	1074.12	0.00
02	1763.07	1762.21	−0.05	1762.21	1762.21	0.00
03	2408.42	2374.36	−1.41	2373.65	2373.65	0.00
04	2958.23	2858.20	−3.38	2852.29	2815.48	−1.29
05	3134.04	3040.84	−2.97	3029.65	2993.94	−1.18
06	3904.07	3758.36	−3.73	3627.18	3629.72	0.07
07	1423.35	1422.65	−0.05	1418.22	1418.22	0.00
08	2150.22	2103.89	−2.15	2102.61	2096.73	−0.28
09	2833.80	2783.95	−1.76	2737.82	2730.54	−0.27
10	3717.22	3577.28	−3.76	3505.27	3499.56	−0.16
11	1031.49	1005.73	−2.50	1005.73	1005.73	0.00
12	1500.48	1487.64	−0.86	1478.51	1472.76	−0.39
13	2020.58	2014.02	−0.32	2011.24	2001.83	−0.47
14	2247.72	2221.17	−1.18	2202.08	2215.51	0.61
15	2509.75	2494.82	−0.59	2494.57	2465.25	−1.18
16	2943.90	2939.20	−0.16	2901.02	2896.03	−0.17
17	1250.09	1239.13	−0.88	1236.24	1236.24	0.00
18	1809.35	1796.21	−0.73	1792.61	1796.21	0.20
19	2310.92	2318.59	0.33	2285.10	2292.45	0.32
20	3131.90	3079.73	−1.67	3079.16	3076.37	−0.09
Avg.	2306.63	2268.10	−1.40	2248.46	2242.63	−0.21

Following Savelsbergh (1992) the concept of forward slack times is used in the context of our VNS to introduce the consideration of route durations in our search. Note, that this concept is also used in Cordeau, Laporte, and Mercier (2004). The main idea is to keep track of the waiting times in the schedule and to compute the maximum amount of time by which service at a customer can be postponed without any time window violations along the route. Using this forward slack information to evaluate the route durations more accurately reduces the objective function values for both our VNS and the TS from Cordeau, Laporte, and Mercier (2004) as shown in Table 5.

Table 5 shows for each problem instance the best solutions obtained by the TS and the VNS both before and after the inclusion of the forward slack. Additionally the table shows the percentage difference between the objective values of the two metaheuristics again for both cases. Finally, the last row provides averages over all 20 instances.

The results presented in Table 5 reveal several things. First, the use of the forward slack leads to a decrease in the objective value by an average of almost 3% for the TS and slightly more than 1% for the VNS. Second, the gap between the two metaheuristics has narrowed

Table 6. Average time (in minutes) needed to reach solutions of pre-specified quality.

	RPD (avg. from 10 runs of the VNS)		
No.	10%	5%	2.5%
01	0.00	0.00	0.00
02	0.02	0.05	0.46
03	0.08	1.07	10.73
04	0.59	4.46	19.41
05	3.53	7.27	24.51
06	1.43	8.71	52.78
07	0.00	0.00	0.09
08	0.05	0.50	3.38
09	0.12	0.69	6.49
10	1.04	7.48	38.43
11	0.04	0.38	5.67
12	0.33	2.97	45.09
13	0.54	2.59	11.73
14	1.62	9.21	91.14
15	3.41	15.75	219.16
16	3.10	11.69	68.59
17	0.81	7.91	19.01
18	3.23	12.80	160.63
19	0.54	5.32	40.47
20	4.87	18.40	108.33

to 0.2% on average. Particularly, for the 6 smallest instances with less than 100 customers both algorithms find the same solutions.

Overall, the results show that the structural improvements of the two algorithms had a much greater impact on solution quality than the actual choice of the metaheuristic. Having shown, that the VNS is still competitive with the TS after including the forward slack mechanism we want to turn to the evolution of the solution quality obtained by the VNS over time.

To that end Table 6 shows for each instance the average time (in minutes) needed to obtain solutions that are 10%, 5% and 2.5% worse than the best known results. Note, that the times in Table 6 correspond to computation times on a Pentium 4 with 2 GHz. It can be seen from Table 6 that the VNS finds reasonably good solutions with an average deviation of 5 percent in less than 20 minutes for the largest instances. High quality solutions may take a VNS run of several hours with a small chance of getting stuck in a mediocre local optimum.

6. Conclusion

In this paper the MDVRPTW was tackled by an implementation of a VNS based metaheuristic. The main features of this algorithm are a simple and flexible Local Search as well as an acceptance criterion for neighboring solutions inspired by Threshold Accepting.

The results obtained through an extensive numerical analysis showed that the algorithm is competitive to an existing TS approach. Considering the best solutions found our algorithm outperforms the TS by finding 10 new best solutions and 6 ties, while the TS holds only 4 best known results. More importantly, the VNS proposed in this paper has shown a much better scaling than the TS, indicating its superiority for large real-world sized instances.

Apart from these statements about efficiency and effectiveness, the algorithm also fulfills two other criteria for a well balanced algorithm applicable in industrial settings, namely flexibility and simplicity. More precisely, the Local Search procedure is based on routes and uses only one kind of operator, namely a restricted version of 3-opt. Thus, the algorithm can be applied without much modification to the VRPTW and the VRP, and it can handle different complicating constraints like heterogeneous fleets and backhauls.

Acknowledgments

The authors wish to thank three anonymous referees, an associate editor and Pierre Hansen for valuable comments on an earlier version of this paper. Financial support from the Oesterreichische Nationalbank (OENB) for financial support under grant #8630 is gratefully acknowledged.

Notes

1. See Toth and Vigo (2002), p. 1.
2. See Hansen and Mladenović (2001), p. 450.

References

Braysy, O. (2003) "A Reactive Variable Neighborhood Search for the Vehicle-Routing Problem with Time Windows." *INFORMS Journal on Computing* 15(4), 347–368.

Braysy, O. and M. Gendreau. "Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms." To appear in *Transportation Science*.

Braysy, O. and M. Gendreau. "Vehicle Routing Problem with Time Windows, Part II: Metaheuristics." To appear in *Transportation Science*.

Chao, I.M., B.L. Golden, and E.A. Wasil. (1993). "A New Heuristic for the Multi-Depot Vehicle Routing Problem that Improves Upon Best Known Solutions." *American Journal of Mathematical and Management Sciences* 13, 371–406.

Cordeau, J.F., M. Gendreau, and G. Laporte. (1997). "A Tabu Search Heuristic for Periodic and Multi-Depot Vehicle Routing Problems." *Networks* 30, 105–119.

Cordeau, J.F., G. Laporte, and A. Mercier. (2001). "A Unified Tabu Search Heuristic for Vehicle Routing Problems with Time Windows." *Journal of the Operational Research Society* 52, 928–936.

Cordeau, J.F., G. Laporte, and A. Mercier. (2004). "An Improved Tabu Search Algorithm for the Handling of Route Duration Constraints in Vehicle Routing Problems with Time Windows." *Journal of the Operational Research Society* 55, 542–546.

Dongarra, J. (2003). "Performance of Various Computers Using Standard Linear Equations Software (Linpack Benchmark Report)." University of Tennessee Computer Science Technical Report, CS-89–85.

Dueck, G. and T. Scheuer. (1990). "Threshold Accepting: A General Purpose Optimization Algorithm Appearing Superior to Simulated Annealing." *Journal of Computational Physics* 90, 161–175.

Garey, M.R. and D.S. Johnson. (1979). *Computers and Intractability: A Guide to the Theory of NP Completeness*. New York: W. H. Freeman & Co.

Hansen, P. and N. Mladenović. (1999). "An Introduction to Variable Neighborhood Search." In S. Voss et al. (eds.), *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*. Boston: Kluwer Academic Publishers.

Hansen, P. and N. Mladenović. (2001). "Variable Neighborhood Search: Principles and Applications." *European Journal of Operational Research* 130, 449–467.

Kohl, N., J. Desrosiers, O.B. Madsen, M.M. Solomon, and F. Soumis. (1999). "2-Path Cuts for the Vehicle Routing Problem with Time Windows." *Transportation Science* 33, 101–116.

Laporte, G. and Y. Nobert. (1987). "Exact Algorithms for the Vehicle Routing Problem." *Annals of Discrete Mathematics* 31, 147–184.

Mladenović, N. and P. Hansen. (1997). "Variable Neighborhood Search." *Computers and Operations Research* 24, 1097–1100.

Renaud, J., F.F. Boctor, and G. Laporte. (1996). "An Improved Petal Heuristic for the Vehicle Routing Problem." *Journal of the Operational Research Society* 47, 329–336.

Savelsbergh, M.W.P. (1992). "The Vehicle Routing Problem with Time Windows: Minimizing Route Duration." *ORSA Journal on Computing* 4, 146–154.

Taillard, E.D., P. Badeau, M. Gendreau, F. Guertin, and J.Y. Potvin. (1997). "A Tabu Search Heuristic for the Vehicle Routing Problem with Soft Time Windows." *Transportation Science* 31, 170–186.

Toth, P. and D. Vigo. (eds.) (2002). "The Vehicle Routing Problem." *SIAM Monographs on Disrcete Mathematics and Applications*, Philadelphia.