A Variable Neighborhood Search Heuristic for Periodic Routing Problems

Vera C. Hemmelmayr, Karl F. Doerner[*], Richard F. Hartl

Department of Business Administration, University of Vienna, Bruenner Strasse 72, 1210 Vienna, Austria
{Vera.Hemmelmayr, Karl.Doerner, Richard.Hartl}@univie.ac.at

Abstract

The aim of this paper is to propose a new heuristic for the Periodic Vehicle Routing Problem (PVRP) without time windows. The PVRP extends the classical Vehicle Routing Problem to a planning horizon of several days. Each customer requires a certain number of visits within this time horizon while there is some flexibility on the exact days of the visits. Hence, one has to choose the visit days for each customer and to solve a VRP for each day. Our method is based on Variable Neighborhood Search (VNS). Computational results are presented, that show that our approach is competitive and even outperforms existing solution procedures proposed in the literature. Also considered is the special case of a single vehicle, i.e. the Periodic Traveling Salesman Problem (PTSP). It is shown that slight changes of the proposed VNS procedure is also competitive for the PTSP.

Keywords: Periodic Vehicle Routing Problem, Periodic Traveling Salesman Problem, Metaheuristics, Variable Neighborhood Search

1 Introduction

Vehicle Routing Problems (VRPs) have received considerable attention both in theoretical research and in real world applications. Forming the basis of many routing

[*]Corresponding author. Tel. ++43–1–4277– 38113, Fax.: +43–1–4277–38094

models, they have been extended in various directions. In this paper we focus on the Periodic Vehicle Routing Problem (PVRP) in which a planning period of several days is considered and customers must be visited more than once. Different customers usually require different numbers of visits in a certain time horizon. Customers with larger demands or smaller storage capacities require more visits than customers with smaller demands or larger storage capacities. This type of problem occurs e.g. in grocery distribution (Carter et al., 1996), soft drink industry (Golden and Wasil, 1987), waste collection (Beltrami and Bodin, 1974, Russel and Igo, 1979) and others.

Early heuristics for the PVRP are proposed by Beltrami and Bodin (1974) and by Russel and Igo (1979). Other heuristics are developed by Christofides and Beasley (1984), Tan and Beasley (1984) and Russel and Gribbin (1991). Gaudioso and Paletta (1992) present a heuristic for the PVRP with the objective to minimize the number of vehicles. Chao et al. (1995a) provide a two phase heuristic. To obtain an initial solution they solve an integer linear program to assign visit day combinations to the customers. In a second phase, they use several improvement operators while they relax the capacity of the vehicles. When getting stuck, re-initializations are performed.

Cordeau et al. (1997) propose a tabu search heuristic for the PVRP that can also be used to solve the Multi-Depot Vehicle Routing Problem and the Periodic Traveling Salesman Problem. The neighborhood consists of moving a customer from one route to another route of the same day or assigning a new visit combination to a customer. Insertions and removals of customers are performed using the GENI operator (Gendreau et al., 1992). The tabu search algorithm allows for infeasible solutions during the search process using an adaptive penalty function. Recently a scatter search procedure was developed by Alegre et al. (2007) for solving a problem of periodic pick-up of raw materials for a manufacturer of auto parts. They use a two-phase approach, that first assigns orders to days and then constructs the routes of each day. Finally Drummond et al. (2001) proposed parallel genetic algorithms.

Furthermore special implementations for real-world problems were provided by Hadjiconstantinou and Baldacci (1998) who propose a heuristic for a Multi-Depot Period Vehicle Routing Problem that arises in the utility sector. Angelelli and Speranza (2002) suggest a tabu search for a special application, a periodic vehicle routing problem with intermediate facilities, where vehicles can replenish their capacity at intermediate facilities. Blakeley et al. (2003) developed an optimization system relying on several algorithms for planning the maintenance of escalators and elevators.

The PTSP is a special case of the PVRP where only one vehicle is available every day and tour length or duration constraints are not considered. A mathematical formulation of the PVRP and the PTSP can be found in Cordeau et al. (1997). Heuristics for the PTSP are provided by Christofides and Beasley (1984), Paletta (1992).

These earlier solution techniques are outperformed by more recent metaheuristic approaches. Chao et al. (1995b) start from an initial solution and exchange visit day combinations by using the record-to-record approach of Dueck (1993). New solutions are accepted if their cost is below a specified threshold, being the cost of the best solution found plus a certain deviation. Local Search and re-initializations are performed afterwards. Cordeau et al. (1997) develop a tabu search method based on the GENI operator (Gendreau et al., 1992). They apply their method also to the Periodic Vehicle Routing Problem (PVRP) and to the multi depot vehicle routing problem. Paletta (2002) proposes an improvement procedure within a tour construction procedure and a new version of the improvement procedure is proposed by Bertazzi et al. (2004).

We develop another metaheuristic solution approach for the PVRP and the PTSP that is based on Variable Neighborhood Search (VNS). VNS is a local search based metaheuristic first proposed by Mladenović (1995), Mladenović and Hansen (1997), Hansen and Mladenović (1997). The VNS approach has already been successfully applied to other variants of the VRPs (see e.g. Braysy, 2003, Polacek et al., 2004, 2005). However, to the best of our knowledge VNS has not been applied to periodic routing problems so far.

The paper has two main contributions. First, from a technical point of view, it presents the first application of a VNS to periodic routing problems. Second, from a problem oriented point of view the computational results show that the approach is competitive with the existing techniques. The developed algorithm is simple, flexible and accurate and yields several new best solutions.

The remainder of the paper is organized as follows. In Section 2 we describe the proposed algorithm for the PVRP and in section 3 we discuss the appropriate adaptations for applying it to the PTSP. In Section 4, we present a computational study that analyzes the proposed solution approach and compares it to state of the art metaheuristics. Finally, Section 5, concludes the paper.

2 Solution Procedure for the PVRP

We first describe our VNS algorithm for the PVRP. In section 3 we will propose minor adaptations of this method in order to be used for solving the PTSP. The basic idea of VNS is a systematic change of neighborhoods within a local search procedure. Starting from any initial solution, a so called shaking step is performed by randomly selecting a solution from the first neighborhood. This is followed by applying an iterative improvement algorithm. This procedure is repeated as long as a new incumbent solution is found. If not, one switches to the next neighborhood (which is typically larger) and performs a shaking step followed by the iterative improvement. If a new incumbent solution is found one start with the first neighborhood; otherwise one proceeds with the next neighborhood, etc. The steps of the basic VNS are shown in Figure 1, where N_κ ($\kappa = 1, \ldots, \kappa_{max}$) is the set of neighborhoods. The stopping condition can be a limit on CPU time, a limit on the number of iterations, or a limit on the number of iterations between two improvements. See Mladenović and Hansen (1997) and Hansen and Mladenović (2000, 2001) for a more thorough description of VNS.

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, or if some acceptance criterion is met, 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 VNS (c. f. Hansen and Mladenović, 2001

In the next subsection, we describe the different components of the VNS implemented for the PTSP and the PVRP.

```
Apply Clarke And Wright
for each day do
  while number of routes > number of vehicles do
    shortest_route := find route with fewest number of customers
    for each customer ∈ shortest_route do
      delete
      insert in cheapest position of the remaining routes
    end for
  end while
end for
```

Figure 2: Build Initial Solution

2.1 Initial Solution

For obtaining an initial solution each customer is assigned a visit day combination randomly. Routes are constructed by solving a vehicle routing problem for each day using the Clarke and Wright savings algorithm (Clarke and Wright, 1964). The Clarke and Wright savings algorithm terminates when no two routes can feasibly be merged, i.e., no two routes can be merged without violating the route duration or capacity constraints. As a result, the number of routes may exceed the number of available vehicles. In that case, a route with the fewest customers is identified and the customers in this route are moved to other routes (minimizing the increase in costs). Note that this may result in routes that no longer satisfy the duration or capacity constraints. This step is repeated until the number of routes is equal to the number of vehicles. Since the initial solution may not be feasible the VNS needs to incorporate techniques that drive the search to a feasible solution.

2.2 Shaking

The set of neighborhoods used for shaking is at the heart of the VNS. Each neighborhood should strike a proper balance between perturbing the incumbent solution and retaining the good parts of the incumbent solution.

Two popular and effective neighborhoods for vehicle routing are based on the move and the cross-exchange operators. A classification of move and cross exchange can be found in Van Breedam (1994) and Kindervater and Savelsbergh (1997). The operator move inserts a segment of one route into a different route. For example in Figure 3 customers x'_2 and y_2 are moved from route two to route one. In our algorithm we relocate up to three customers. The CROSS exchange operator exchanges two segments of different routes. Figure 4 shows that the segment from customer x'_1 to y_1 of route one is exchanged with the segment x'_2 to y_2 of route two. We consider a segment length of up to six customers. The orientation of the segment(s) and of the route(s) is preserved by the move and cross-exchange operators. The move and cross-exchange operators are used to define a set of neighborhoods that allow the exploration of increasingly distant solutions from the incumbent to overcome local optimality and strive for global optimality. The metric to measure the increasing size of a neighborhood is given by the maximum number of customers in the route segments used within the operators. The cross- exchange operator is shown in Figure 4.

For the periodic vehicle routing problem it is essential to also have a neighborhood that changes the visit combinations for customers. We use neighborhoods in which the visit combinations of a limited number of customers are changed. For each of these a visit combination is chosen randomly. Here, the metric to measure the increasing size of a neighborhood is given by the maximum number of customers for which the visit day combination is changed. Table 1 shows the maximum segment length considered for each neighborhood κ.

In each neighborhood all the possible segment lengths and numbers of customers

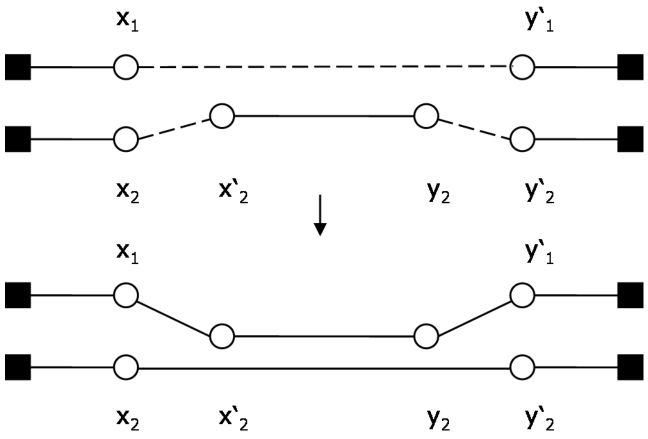

Figure 3: The Move Operator

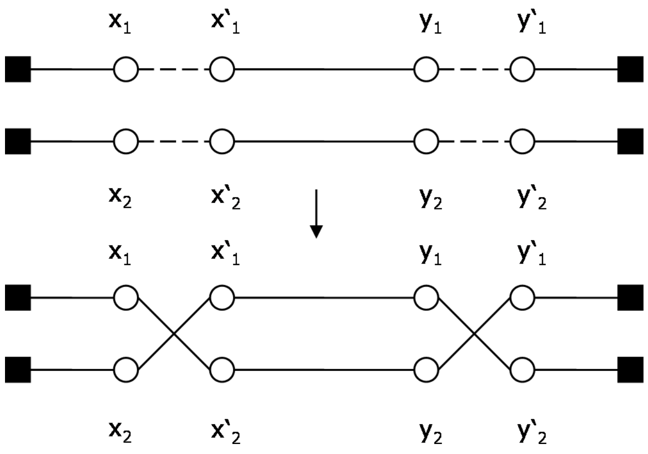

Figure 4: The CROSS Exchange Operator

Table 1: Set of Neighborhood Structures with $\kappa_{max} = 15$

κ	operator	min. customers	max. customers
1	change combination	1	1
2	change combination	1	2
3	change combination	1	3
4	change combination	1	4
5	change combination	1	5
6	change combination	1	6
		min. segment length	max. segment length
7	move	1	min(1, n)
8	move	1	min(2, n)
9	move	1	min(3, n)
10	cross	1	min(1, n)
11	cross	1	min(2, n)
12	cross	1	min(3, n)
13	cross	1	min(4, n)
14	cross	1	min(5, n)
15	cross	1	min(6, n)

are equally likely to be chosen. Hence our choice of neighborhoods is biased toward smaller changes to focus the search rather close to the incumbent solution.

2.3 Local Search

A solution obtained through shaking is submitted to a local search procedure to come up with a locally optimal solution. We apply one of the most popular iterative improvement procedures, namely 3-opt, which was introduced by Lin (1965). This heuristic tries all shifts of some subsequence to different positions in the same route. More precisely, three edges are deleted and replaced by three other edges. The tour is 3-optimal, when it cannot be improved by such a change. In our algorithm 3-opt without sequence inversion, also often denoted as 3-opt*, is used. Only individual routes are improved, so that only the routes that have changed during shaking have to be re-optimized. The local search restarts immediately after an improving move was found.

2.4 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. The acceptance criterion in the basic VNS is to accept only improvements. However that way the search can easily get stuck in a local optimum. Thus in most cases it is essential to also have a strategy of accepting non-improving solutions under certain conditions. We implement a scheme that is inspired by Simulated Annealing (SA) (Kirkpatrick et al., 1983). Hence, our method could be considered a hybrid of VNS and SA. However since the SA part is rather small we prefer to regard it as a VNS. More specifically, improving solutions are always accepted and inferior solutions are accepted with a probability $exp^{\frac{-(f(x'')-f(x))}{T}}$ where $f(x)$ is the cost of solution x possibly including penalty costs if it is infeasible. The acceptance of inferior solutions depends on a given temperature T and the difference between the costs of the new solution and the incumbent solution. The temperature T is linearly decreased in η/k stages during the search process, where η represents the total number of iterations executed. Thus, every k iterations T is decreased by an amount $\frac{T*k}{\eta}$. Different cooling schedules like exponential cooling and a constant temperature have also been considered but the linear annealing scheme provided the best results.

An alternative strategy would be the so called Skewed VNS, an extension of the basic VNS proposed by Hansen and Mladenović (2000). 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. Let the function $\rho(x, x'')$ measure the distance between the incumbent solution x and the new solution x''. A new solution is accepted if $f(x'') - \alpha\rho(x, x'') < f(x)$.

Another approach to accept non-improving solutions is based on threshold accepting (TA). A solution yielding an improvement is always accepted. Moreover ascending moves are accepted after a minimum number of iterations counted from the last accepted move, but only if the cost increase is below a certain threshold. We implemented these three possibilities (SA, skewed, TA). In the Skewed VNS approach we measure the distance $\rho(x, x'')$ by using the number of customers that are exchanged in the move and CROSS operator and the number of routes that are changed in the change combination operator. Our implementation of the TA approach is based on the one described in Polacek et al. (2004). Computational experiments show that SA delivers 2.71% better solutions than SVNS and 3.61% than threshold accepting. In what follows, we will only report results based on the SA acceptance criterion.

As mentioned at the start of this section, the VNS has to be able to handle infeasible solutions. Infeasibility occurs if the total capacity or tour duration exceed the specified limits. We use a weighted, linear penalty function for violations of this constraint. This penalty function is added to the objective function before the solution is evaluated for acceptance. The weights are adjusted dynamically. If the total capacity or tour duration of any tour is exceeded the respective weight is increased, if it is feasible the weight is decreased. However the weights can only be adjusted within predefined upper and lower bounds. Hence if a tour is infeasible, but the weight would exceed the upper bound it will not be increased any more and vice versa. The weight is initialized with its upper bound in order to lead the search toward feasible solutions in the beginning.

3 Solution Procedure for the PTSP

The solution procedure for the PTSP is based on the one for the PVRP. But in order to solve the PTSP more efficiently some adaptations are appropriate. The algorithm for the PVRP uses Clarke and Wright savings algorithm (Clarke and Wright, 1964) for building an initial solution. For the PTSP the savings measure is irrelevant and best insertion is used to build a starting solution.

In the PVRP algorithm the shaking phase is composed of the operators move, CROSS and change combination, where move and CROSS are used inter route only, i.e. customers are exchanged between routes. The shaking phase in the PTSP is similar, but the operators move and CROSS are now used intra route because there is only one route for every day.

For the PVRP the local search phase consists of 3-opt. But typically, in the standard benchmark instances each PTSP tour consists of considerably more customers than a PVRP tour. This is why an efficient implementation for the PTSP should employ a faster local search. In order to save computation time the 2-opt operator is applied for the PTSP instead of 3-opt as used for the PVRP. 2-opt was introduced by Croes, 1958. This operator deletes two edges of a tour and reconnects those paths in the other possible way. The local search restarts immediately after an improving move was found.

Chao et al. (1995b) imposed the constraint that the traveling salesman has to visit at least one city each day. Our algorithm handles this by penalizing an empty day with a constant penalty. The penalty is added to the objective function before the solution is evaluated.

4 Computational Experiments

4.1 Results on PVRP instances

4.1.1 Test Instances

We tested our algorithm on instances taken from the literature. There are two data sets available. The so called "old data" set contains 32 instances. Instances p1–p10 were proposed by Eilon et al. (1971) for the VRP and adapted to the PVRP by Christofides and Beasley (1984). Russel and Igo (1979) proposed instance p11. Instance p12 and p13 are taken from Russel and Gribbin (1991). Instances p14–p32 were introduced by Chao et al., 1995. In the latter data set $D = \infty$, which means that tour duration is not restricted. The 10 instances of the new data set were provided by Cordeau et al. (1997). The old data were solved by Christofides and Beasley (1984) (CB), Tan and Beasley (1984) (TB), Russel and Gribbin (1991) (RG), Chao et al. (1995a) (CGW), Cordeau et al. (1997) (CGL) and Alegre et al. (2007) (ALP) and we compare our results with their results. Results from Drummond et al. (2001) were not taken into account because according to Alegre et al. (2007) they are not comparable. The algorithms CGL and ALP delivered the best solution values so far and CGW delivers some ties. Results for the "new data" set were only given by Cordeau et al. (1997) (CGL). A description of the different instances of the old data set can be found in Table 2 and for the new data set can be found in Table 3.

4.1.2 Parameter Settings

Compared to other metaheuristics only few parameters have to be determined and tuned. In our implementation these are the initial temperature for accepting de-

Table 2: Instance description of the "old data" set for the PVRP, where n is the number of customers, m is the number of vehicles that can be used, t is the number of days in the planning horizon, D is the maximum duration of a route, Q is the maximum capacity of the vehicles, and f_i is the number of customers that must be visited i times.

Instance	n	m	t	D	Q	service frequencies					
						f1	f2	f3	f4	f5	f6
v-p01	50	3	2		160	50					
v-p02	50	3	5		160	17	26			7	
v-p03	50	1	5		160	50					
v-p04	75	2	5		140	75					
v-p05	75	6	5		140	30	34			11	
v-p06	75	1	10		140	75					
v-p07	100	4	2		200	100					
v-p08	100	5	5		200	40	46			14	
v-p09	100	1	8		200	100					
v-p10	100	4	5		200	40	46	14			
v-p11	139	4	5		235	103	22	12	1	1	
v-p12	163	3	5		140	148	8	7			
v-p13	417	9	7		2000	377	40				
v-p14	20	2	4		20	8	8		4		
v-p15	38	2	4		30	16	16		6		
v-p16	56	2	4		40	24	24		8		
v-p17	40	4	4		20	16	16		8		
v-p18	76	4	4		30	32	32		12		
v-p19	112	4	4		40	48	48		16		
v-p20	184	4	4		60	80	80		24		
v-p21	60	6	4		20	24	24		12		
v-p22	114	6	4		30	48	48		18		
v-p23	168	6	4		40	72	72		24		
v-p24	51	3	6		20	36	9				6
v-p25	51	3	6		20	36	9				6
v-p26	51	3	6		20	36	9				6
v-p27	102	6	6		20	72	18				12
v-p28	102	6	6		20	72	18				12
v-p29	102	6	6		20	72	18				12
v-p30	153	9	6		20	108	27				18
v-p31	153	9	6		20	108	27				18
v-p32	153	9	6		20	108	27				18

Table 3: Instance description of the "new data" set for the PVRP, where n is the number of customers, m is the number of vehicles that can be used, t is the number of days in the planning horizon, D is the maximum duration of a route, Q is the maximum capacity of the vehicles, and f_i is the number of customers that must be visited i times.

Instance	n	m	t	D	Q	service frequencies				
						f1	f2	f3	f4	f6
v-pr01	48	2	4	500	200	24	12		12	
v-pr02	96	4	4	480	195	48	24		24	
v-pr03	144	6	4	460	190	72	36		36	
v-pr04	192	8	4	440	185	96	48		48	
v-pr05	240	10	4	420	180	120	60		60	
v-pr06	288	12	4	400	175	144	72		72	
v-pr07	72	3	6	500	200	18	18	18		18
v-pr08	144	6	6	475	190	36	36	36		36
v-pr09	216	9	6	450	180	54	54	54		54
v-pr10	288	12	6	425	170	72	72	72		72

teriorating solutions and the initial values for the penalty terms in the objective function. Several experiments were made to find a good initial temperature T for SA. It turned out that the initial temperature should be higher the larger the average distance between customers is. In order not to have different initial temperatures for all instances we group the instances into those with large average distances between customers (p27–p32) and small average distances (all other instances). The initial temperature is set to 125 in the former case and to 7 in the latter case. The temperature is decreased every 1000 iterations, in a way that it becomes 0 in the last 1000 iterations. The weights for penalizing tour length and duration violations are adjusted dynamically. They are set to a value of 1000 in the beginning and multiplied by a factor 1.001 if the solution is infeasible or divided by this factor if the solution is feasible. But these adaptations are only applied if the weights remain in the interval between 10 and 1000.

4.1.3 Numerical Results

The algorithm was coded in ANSI C and compiled with the GNU C compiler version 3.4.4. Our experiments were performed on a PC with 3.2 GHz. Preliminary tests were performed in order to verify that all 15 neighborhoods contribute to the performance of the algorithm. Clearly neighborhoods with lower number found new incumbent solutions more often since they are used more frequently. However all neighborhoods found a significant number of new incumbent solutions (see Table 15 below).

Table 4 reports results for the old data set. It shows the objective function values for CB, TB, RG, CGW, CGL, ALP and our algorithm with 10^7 iterations (in Table 7 below we will show that the run times for 10^7 iterations are comparable with those of the best benchmark algorithms). The best results are always marked in bold.

Our algorithm was developed and tuned for solving truly periodic vehicle routing problems. In the old data set, however there are some degenerate instance that are not really periodic; e.g. in instances v-p1, v-p3, v-p4, v-p6, v-p7 and v-p9 all orders have frequency one. Since all customers need to be visited only once and this visit can take place on any day of the planning period, these instances reduce to simple VRPs (or more precisely multiple TSPs with given vehicle number). Moreover v-p3, v-p6 and v-p9 have only one vehicle available. It can be seen that our algorithm delivers competitive results also on these degenerate instances but outperforms the other algorithms on instances that have higher visit frequencies, i.e. instances v-p14–v-p32. Especially for larger instances our algorithm provides very good results.

As the algorithms CGL and ALP provided the best results on the old data set so far, we report the development of our solution approach in comparison to these two algorithms. Table 5 shows average results over 10 runs with 10^6, 10^7, 10^8 and 10^9 iterations and compares them to the results delivered by CGL and ALP. The last 2 rows show the average per cent differences between our results and CGL and ALP, respectively. More precisely, we computed the per cent difference of the average of 10 runs to the benchmark approaches for each instance and reported the average aver all instances. It is seen that after a very short running time, i.e. 10^6 iterations, the VNS is still about 1.5 % worse. When comparing VNS after 10^7 iterations to CGL and ALP the solution quality is already -0.17 % and -0.21 % better, respectively. The results still improve with increasing number of iterations. It should be noted that these average values also include the results for the degenerate (non-periodic) instances. If we would omit these or adapt the VNS to these special situations the average results would be better.

We applied our algorithm also to the new instances. Table 6 shows the results for the new data after 10^6, 10^7, 10^8 and 10^9 iterations compared to CGL. We present average results over 10 runs and we compare it to the results of the CGL (1 run). Note that these instances were only solved by CGL so far. Applying our algorithm 10^7 iterations we improved the results by -0.92 %. When applying the algorithm 10^9 iterations the improvement is almost -2 %. On some small instances there are only moderate improvements e.g. instance v-pr07 with 72 customers an improvement of -0.09 % is possible whereas for instance v-pr10 with 288 customers improved solutions of -4.83 % were possible. After 10^8 iterations already for all of the 10 instances improved results were found on average. The results confirm the findings for the old data. For truly periodic problems and in particular for larger instances our algorithm performs very well.

Table 7 reports computation times on the instances of the old data set for the so far best approaches CGL (Cordeau et al., 1997) and ALP (Alegre et al., 2007) and run times for 10^7 iterations of VNS. All run times are given in seconds. If algorithms are run on different machines, a direct comparison of computation times is always difficult. To make the results comparable at least to the CGL algorithm we ran the code of CGL on the same machine as VNS which yielded considerably shorter times compared to those originally reported by CGL. We are grateful to the authors of CGL for providing the code to us. Although the results by ALP were published in 2007, they use a rather old machine which is – according to Dongarra (2005) – 7.17 times slower than our machine. The total computation time 44914.6 must therefore be corrected to 6264.24 seconds, which is still larger than for CGL and our VNS.

Table 8 reports run times for the instances of the new data set of CGL and 10^7 iterations of VNS. Again both algorithms were run on the same machine. T refers to the total time used to execute the algorithm and T^* is the time needed to obtain the best solution during search process. It shows that the VNS algorithm is competitive to CGL and is faster in larger instances.

As also done by CGL, we collected all results obtained for different run times and different parameter settings during the fine tuning phase in order to keep track of the best solutions found. Our algorithm was able to improve almost all of the best known results reported in literature as can be seen in Tables 9 and 10. For the old data set 27 out of 32 instances are improved or equal results were found. For the new data set 9 out of 10 instances are improved and there is one tie. The best known results of the new data have been improved by -1.50 %.

4.2 Results on PTSP instances

4.2.1 Test Instances

Our algorithm was tested with the standard benchmark instances proposed in the literature. Instances t-p1 to t-p10 were given by Eilon et al. (1971) for the VRP and adapted to the PTSP by Christofides and Beasley (1984). Instances t-p11 to t-p23 were introduced by Chao et al. (1995b) and instances t-p24 to t-p34 are taken from Cordeau et al. (1997). Results were given by Christofides and Beasley (1984), Paletta (1992), Chao et al. (1995b), Cordeau et al. (1997), Paletta (2002) and Bertazzi et al. (2004). A detailed description of the instances indicating the number of cities and the planning horizon is given in Table 11.

As in case of the PVRP also some of the PTSP instances are degenerated. More precisely, in instances t-p1, t-p3, t-p4, t-p6, t-p7 and t-p9, all customers have a visit frequency of one. Due to the constraint that at least one customer has to be visited every day, the best solution in these cases is to form a TSP on one day with all the customers except for T-1 customers, that are close to the depot. Then on each of the remaining T-1 days one of these close customers is visited. We report our results also for these degenerated in instances, but no fine tuning was made to solve these more efficiently. Our code is designed to solve truly periodic problems.

4.2.2 Parameter Settings

In order to provide an (almost) generic solution approach, we kept all parameters for the PTSP same as for the PVRP. So we again set the initial temperature T for Simulated Annealing to 7 for all instances. The temperature is decreased linearly every 1000 iterations, in a way that it becomes 0 in the last iterations. The penalty for an empty day is set to 1000 and the stopping condition is a fixed number of iterations.

4.2.3 Numerical Results

Table 12 reports the results of our VNS with independent runs of 10^6, 10^7 and 10^8 iterations compared to the best results obtained by the different algorithms of Chao et al. (1995b) (CGW), Cordeau et al. (1997) (CGL), Paletta (2002) (P) and Bertazzi et al. (2004) (BPS). The average solution quality is about the same after 10^6 iterations compared to the best results found by the other authors. Applying the VNS for 10^7 iterations the solution quality can be improved by 0.34 %. Note that this improvement is over the best solution obtained by any of the approaches mentioned. An individual comparison of our VNS with any of the other approaches would yield even higher improvements. For longer run times further improvements are obtained. After 10^8 iterations the solution quality is 0.52 % better compared to the best results reported by the other authors. As mentioned in Section 4.2.1 there are some instances, that are no real PTSP instances but rather standard TSP instances. If these degenerated instances, namely t-p1, t-p3, t-p4, t-p6, t-p7, and t-p9 are disregarded, and the comparison is only made for the remaining ones, then we already have an average improvement by our VNS of -0.24 % after 10^6 iterations and improvements of -0.40 % and -0.49 % after 10^7 and 10^8 iterations, respectively.

Table 13 shows the best known results produced by our algorithm. Best known solutions means all solutions that were found with different parameter settings. The results are compared to the best known results given in the literature. New best results were found for 11 instances and for all the other instances tie were achieved.

We refrain from reporting detailed run times here since most algorithms did not solve all instances. But the run times for 10^6 iterations of the VNS are comparable or shorter than those of the other metaheuristic approaches also after adjustment of run times w.r.t. the machine used.

4.3 Analysis of Neighborhood Structure

We also investigated the contribution of all shaking operators to the performance of the algorithm and the best ordering of the operators. Table 14 reports the number of times that a neighborhood structure lead to a new incumbent solution after local search within 10^6 iterations. Three different orderings of the neighborhood structures are analyzed: move-CROSS-change combination (cc), CROSS-cc-move and cc-move-CROSS. From the average deviation to the best solution reported in literature it can be seen that the cc-move-CROSS ordering delivers the best results. This can be explained as follows. As VNS starts from the first neighborhood again when it finds a new incumbent solution (see Fig. 1), earlier neighborhood structures are used more often, especially in the beginning of the search. The ordering cc-move-CROSS leads to the best results, because it is important that in the beginning the most suitable visit day combination is selected. However, according to extensive tests the later neighborhoods, move and CROSS, also play an essential part for obtaining good results. These neighborhood structures become more important in the later phases of the optimization runs - when already good visit day combinations are assigned to the customers. Table 14 shows that the number of new incumbent solutions found is generally decreasing with the index of the neighborhood if the ordering of the neighborhoods is correct. If however the cc neighborhoods are scheduled after move and/or CROSS they can show more improvements than earlier neighborhoods. This is an indication that the ordering is not ideal.

While we report detailed tests results for different orderings of the neighborhoods only for the PTSP (similar results are obtained for the PVRP, where some limited test were performed), we investigated for PVRP and PTSP, whether all 15 neighborhoods are in fact reasonable, i.e., whether all contribute to the performance of the algorithm. Table 15 shows (for the best ordering of the neighborhood structures, i.e., cc-move-CROSS), the usage of the neighborhood structures for a different number of iterations. The reported results are the average over 10 runs, summed up over all instances. It can be seen that most incumbent solutions are found in an early phase of the optimization runs. Move and CROSS neighborhoods tend to find more improvements also in the later stages of the optimization.

Table 4: Results for PVRP old data, compared to Tan and Beasley (1984) (TB), Christofides and Beasley (1984) (CB), Russel and Gribbin (1991) (RG), Chao et al. (1995) (CGW), Cordeau et al. (1997) (CGL) and Alegre et al. (2007) (ALP)

Instance	TB	CB	RG	CGW	CGL	ALP	VNS
v-p1	-	547.4	537.3	**524.6**	**524.61**	531.02	**524.61**
v-p2	1481.3	1443.1	1355.4	1337.2	1330.09	**1324.74**	1332.01
v-p3	-	546.7	–	**524.6**	**524.61**	537.37	528.97
v-p4	-	843.9	867.8	860.9	**837.93**	845.97	847.48
v-p5	2192.5	2187.3	2141.3	2089	2061.36	**2043.75**	2059.74
v-p6	-	938.2	–	881.1	840.3	**840.1**	884.69
v-p7	-	839.2	833.6	832	**829.37**	829.65	829.92
v-p8	2281.8	2151.3	2108.3	2075.1	2054.9	**2052.21**	2058.36
v-p9	-	875	–	829.9	**829.45**	829.65	834.92
v-p10	1833.7	1674	1638.5	1633.2	1629.96	**1621.21**	1629.76
v-p11	878.5	847.3	820.3	791.3	817.56	**782.17**	791.18
v-p12	-	-	1312	1237.4	1239.58	**1230.95**	1258.46
v-p13	-	-	3638.1	3629.8	**3602.76**	-	3835.9
v-p14	-	-	-	**954.8**	**954.81**	**954.81**	**954.81**
v-p15	-	-	-	**1862.6**	**1862.63**	**1862.63**	**1862.63**
v-p16	-	-	-	**2875.2**	**2875.24**	**2875.24**	**2875.24**
v-p17	-	-	-	1614.4	**1597.75**	**1597.75**	1601.75
v-p18	-	-	-	3217.7	3159.22	3157	**3147.91**
v-p19	-	-	-	**4846.5**	4902.64	**4846.49**	4851.41
v-p20	-	-	-	**8367.4**	**8367.4**	8412.02	**8367.4**
v-p21	-	-	-	2216.1	2184.04	**2173.58**	2180.33
v-p22	-	-	-	4436.4	4307.19	4330.59	**4218.46**
v-p23	-	-	-	6769	**6620.5**	6813.45	6644.93
v-p24	-	-	-	3773	3704.11	**3702.02**	3704.6
v-p25	-	-	-	3826	**3781.38**	**3781.38**	**3781.38**
v-p26	-	-	-	3834	**3795.32**	3795.33	**3795.32**
v-p27	-	-	-	23401.6	23017.45	22561.33	**22153.31**
v-p28	-	-	-	23105.1	22569.4	22562.44	**22418.52**
v-p29	-	-	-	24248.2	24012.92	23752.15	**22864.23**
v-p30	-	-	-	80982.1	77179.33	76793.99	**75579.23**
v-p31	-	-	-	80279.1	79382.35	77944.79	**77459.14**
v-p32	-	-	-	83838.7	80908.95	81055.52	**79487.97**

Table 5: Results for PVRP old data, compared to CGL (Cordeau et al., 1997) and ALP (Alegre et al., 2007)

Instance	CGL	ALP	10^6	10^7	10^8	10^9
v-p1	**524.61**	531.02	532.69	**524.61**	**524.61**	**524.61**
v-p2	1330.09	**1324.74**	1339.23	1332.01	1328.98	1327.09
v-p3	**524.61**	537.37	545.87	528.97	**524.61**	**524.61**
v-p4	837.93	845.97	860.95	847.48	840.76	**836.03**
v-p5	2061.36	2043.75	2091.48	2059.74	2048.91	**2037.86**
v-p6	840.3	840.1	982.92	884.69	850.76	**839.51**
v-p7	829.37	829.65	833.82	829.92	**828.74**	**827.64**
v-p8	2054.9	2052.21	2068.85	2058.36	**2045.54**	**2041.55**
v-p9	829.45	829.65	842.74	834.92	830.15	**828.34**
v-p10	1629.96	1621.21	1651.45	1629.76	**1615.73**	**1606.03**
v-p11	817.56	782.17	807.32	791.18	784.01	**781.51**
v-p12	1239.58	1230.95	1278.37	1258.46	1238.01	**1220.48**
v-p13	3602.76	-	4066.8	3835.9	3692.72	**3574.63**
v-p14	**954.81**	**954.81**	**954.81**	**954.81**	**954.81**	**954.81**
v-p15	**1862.63**	**1862.63**	**1862.63**	**1862.63**	**1862.63**	**1862.63**
v-p16	**2875.24**	**2875.24**	**2875.24**	**2875.24**	**2875.24**	**2875.24**
v-p17	**1597.75**	**1597.75**	1610.21	1601.75	**1597.75**	**1597.75**
v-p18	3159.22	3157	3179.58	**3147.91**	**3146.61**	**3144.96**
v-p19	4902.64	4846.49	4846.49	4851.41	4846.49	**4845.28**
v-p20	**8367.4**	8412.02	**8367.4**	**8367.4**	**8367.4**	**8367.4**
v-p21	2184.04	2173.58	2198.58	2180.33	2181.2	**2173.08**
v-p22	4307.19	4330.59	**4279.61**	**4218.46**	4213.75	**4210.38**
v-p23	6620.5	6813.45	6716.54	6644.93	**6530.21**	**6478.72**
v-p24	3704.11	3702.02	3719.16	3704.6	**3695.69**	**3692.84**
v-p25	3781.38	3781.38	3781.38	3781.38	3781.16	**3780.8**
v-p26	**3795.32**	3795.33	3834.35	**3795.32**	**3795.32**	**3795.32**
v-p27	23017.45	22561.33	**22391.34**	**22153.31**	**22067.46**	**22001.28**
v-p28	22569.4	22562.44	**22554.72**	**22418.52**	**22381.97**	**22341.66**
v-p29	24012.92	23752.15	**23243.92**	**22864.23**	**22712.61**	**22665.19**
v-p30	77179.33	76793.99	77703.13	**75579.23**	**74915.72**	**74764.59**
v-p31	79382.35	77944.79	78852.79	**77459.14**	**76814.6**	**76630.67**
v-p32	80908.95	81055.52	81887.26	**79487.97**	**78488.16**	**78337.91**
VNS-CGL			1.40	-0.17	-0.85	-1.20
VNS-ALP			1.21	-0.21	-0.78	-1.04

Table 6: Results PVRP new data, development of solution quality by increasing the number of iterations, averaged over 10 runs

Instance	CGL	10^6		10^7		10^8		10^9	
v-pr01	2234.23	**2211.71**	-1.01	**2209.11**	-1.12	**2209.02**	-1.13	**2209.02**	-1.13
v-pr02	3836.49	**3810.48**	-0.68	**3787.51**	-1.28	**3781.28**	-1.44	**3778.49**	-1.51
v-pr03	5277.62	5305.04	0.52	**5243.09**	-0.65	**5228.92**	-0.92	**5210.37**	-1.27
v-pr04	6072.67	6103.66	0.51	**6011.39**	-1.01	**5961.18**	-1.84	**5930.43**	-2.34
v-pr05	6769.8	6968.5	2.94	6778	0.12	**6697.76**	-1.06	**6741.94**	-0.41
v-pr06	8462.37	8678.13	2.55	**8461.45**	-0.01	**8351.49**	-1.31	**8269.92**	-2.27
v-pr07	5000.9	5013.42	0.25	5007.01	0.12	**4998.18**	-0.05	**4996.34**	-0.09
v-pr08	7183.39	7234.89	0.72	**7119.61**	-0.89	**7063.46**	-1.67	**7026.84**	-2.18
v-pr09	10507.34	10540.4	0.31	**10259.09**	-2.36	**10183.56**	-3.08	**10119.09**	-3.70
v-pr10	13629.25	13894.99	1.95	**13342.41**	-2.10	**13157.95**	-3.46	**12969.70**	-4.83
			0.81		-0.92		-1.60		-1.99

Table 7: PVRP Old data – Reported run times in seconds executed on a Pentium III 600 MHz (ALG), and on a 3.2 GHz PC (CGL and VNS)

	T			T*	
Instance	ALP	CGL	VNS	CGL	VNS
v-p01	268	29.4	98.3	6.6	49.6
v-p02	494	35.4	81.6	11.4	47.6
v-p03	45	32.4	100.5	13.2	62.3
v-p04	1426	46.8	67.2	24	49.7
v-p05	1280	52.8	68	32.4	57
v-p06	1797	57	76	51.6	67.8
v-p07	199	76.8	183.2	15	148
v-p08	3584	123.6	142.9	119.4	126.5
v-p09	970	95.4	193.1	34.2	162.1
v-p10	9467	123.6	170	111.6	154.3
v-p11	6492	206.4	253.7	182.4	251.3
v-p12	515	236.4	354.7	184.2	353.6
v-p13		1491.6	127.6	97.2	126.9
v-p14	5	9.6	37.4	0	0
v-p15	1	23.4	93.9	1.2	0
v-p16	2	41.4	217.7	9.6	0.1
v-p17	96	22.2	56.7	12.6	21.6
v-p18	401	64.2	142.5	46.8	101.4
v-p19	20	135.6	258.3	79.2	119.6
v-p20	60	232.2	889.1	91.8	362.3
v-p21	373	33	72.5	21.6	49.7
v-p22	528	132	169.6	91.8	158.4
v-p23	42.09	342	341.4	306	310.3
v-p24	114.31	31.8	52.2	18.6	18.4
v-p25	69	31.2	46.9	5.4	13
v-p26	7.53	31.2	45.2	7.8	8.9
v-p27	219	82.8	66	82.8	64.3
v-p28	435	80.4	64.6	39	64.1
v-p29	19	76.2	59.3	27	57.2
v-p30	19.712	171	78	161.4	76.9
v-p31	7650	160.8	77.1	144.6	75.1
v-p32	8316	145.8	70.4	69	67.8
Total	44914.6	4454.4	4755.6	2099.4	3225.8

Table 8: PVRP New data – Run times in seconds, both algorithms executed on a PC with 3.2 GHz

	T		T*	
Instance	CGL	VNS	CGL	VNS
v-pr01	34.2	180.3	22.8	59.4
v-pr02	97.2	283.3	84.6	238.2
v-pr03	210.6	278.2	163.2	255.9
v-pr04	331.2	314.9	284.4	295.3
v-pr05	576.6	264.9	519	248.9
v-pr06	1090.2	324.4	1031.4	318.5
v-pr07	94.8	246.7	82.2	164.7
v-pr08	260.4	338.6	234	297.4
v-pr09	802.2	423.9	733.2	417.3
v-pr10	2023.8	376.1	2013.6	375.1
Total	5521.2	3031.3	5168.4	2670.7

Table 9: Best known results for PVRP old data, found with different parameter settings

Instance	CB	TB	RG	CGW	CGL	ALP	VNS
v-p1	547.4		537.3	**524.6**	**524.61**	531.02	**524.61**
v-p2	1443.1	1481.3	1355.4	**1322.9**	**1322.87**	1324.74	**1322.87**
v-p3	546.7			**524.6**	**524.61**	537.37	**524.61**
v-p4	843.9		867.8	840.2	835.43	845.97	**835.26**
v-p5	2187.3	2192.5	2141.3	2046.20	**2027.99**	2043.75	2028.02
v-p6	938.2			847.2	836.37	840.1	**835.45**
v-p7	839.2		833.6	831.1	**826.14**	829.65	827.39
v-p8	2151.3	2281.8	2108.3	2042.0	**2034.15**	2052.21	**2034.15**
v-p9	875			828.3	**826.14**	829.65	827.39
v-p10	1674	1833.7	1638.5	1611.9	1595.84	1621.21	**1593.45**
v-p11	847.3	878.5	820.3	785.7	779.29	782.17	**779.06**
v-p12			1312	1219.6	**1195.88**	1230.95	1201.79
v-p13			3638.1	3538	**3511.62**		3513.69
v-p14				**954.8**	**954.81**	**954.81**	**954.81**
v-p15				**1862.6**	**1862.63**	**1862.63**	**1862.63**
v-p16				**2875.2**	**2875.24**	**2875.24**	**2875.24**
v-p17				1614.4	**1597.75**	**1597.75**	**1597.75**
v-p18				3217.7	3147.24	3157	**3136.69**
v-p19				4846.5	**4834.34**	4846.49	**4834.34**
v-p20				**8367.4**	**8367.4**	8412.02	**8367.4**
v-p21				2216.1	2184.04	2173.58	**2170.61**
v-p22				4436.4	4271.11	4330.59	**4193.95**
v-p23				6769	6602.59	6813.45	**6420.71**
v-p24				3773	**3687.46**	3702.02	**3687.46**
v-p25				3826	**3777.15**	3781.38	**3777.15**
v-p26				3834	3795.33	3795.33	**3795.32**
v-p27				23401.6	21956.46	22561.33	**21956**
v-p28				23105.1	22934.71	22562.44	**22305.34**
v-p29				24248.2	22909.36	23752.15	**22639.85**
v-p30				80982.1	75016.58	76793.99	**74464.26**
v-p31				80279.1	78179.89	77944.79	**76552.25**
v-p32				83838.7	80479.2	81055.52	**78072.88**

Table 10: Best known results for PVRP new data, found with different parameter settings

Instance	CGL	VNS	%
v-pr01	**2209.02**	**2209.02**	0.00
v-pr02	3799.28	**3774.09**	-0.66
v-pr03	5218.13	**5175.15**	-0.82
v-pr04	6012.79	**5914.93**	-1.63
v-pr05	6769.8	**6618.95**	-2.23
v-pr06	8422.64	**8258.08**	-1.95
v-pr07	4997.41	**4996.14**	-0.03
v-pr08	7094.52	**6989.81**	-1.48
v-pr09	10370.45	**10075.4**	-2.85
v-pr10	13370.04	**12924.66**	-3.33
			-1.50

Table 11: Description of the PTSP test instances. The notation is the same as for the PVRP instances.

Instance	n	t	service frequencies					
			f1	f2	f3	f4	f5	f6
t-p1	50	2	50					
t-p2	50	5	17	26			7	
t-p3	50	5	50					
t-p4	75	2	75					
t-p5	75	5	30	34			11	
t-p6	75	10	75					
t-p7	100	2	100					
t-p8	100	5	40	46			14	
t-p9	100	8	100					
t-p10	100	5	40	46	14			
t-p11	65	4	48	12		5		
t-p12	87	4	64	16		7		
t-p13	109	4	80	20		9		
t-p14	131	4	96	24		11		
t-p15	153	4	112	28		13		
t-p16	48	4	32	16				
t-p17	66	4	44	22				
t-p18	84	4	56	28				
t-p19	102	4	68	34				
t-p20	120	4	80	40				
t-p21	77	4	56	14		7		
t-p22	154	4	112	28		14		
t-p23	231	4	168	42		21		
t-p24	48	4	24	12		12		
t-p25	96	4	48	24		24		
t-p26	144	4	72	36		36		
t-p27	192	4	96	48		48		
t-p28	240	4	120	60		60		
t-p29	288	4	144	72		72		
t-p30	72	6	18	18	18			18
t-p31	144	6	36	36	36			36
t-p32	216	6	54	54	54			54
t-p33	288	6	72	72	72			72

Table 12: Results for the PTSP with 10^6, 10^7 and 10^8 iterations compared to the best solution values of the algorithms of Chao et al. (1995b), Cordeau et al. (1997), Paletta (2002) and Bertazzi et al. (2004) with a recommended value of parameters.

Instance	min	VNS 10^6		VNS 10^7		VNS 10^8	
t-p1	436.50	**433.18**	-0.76	**432.1**	-1.01	**432.1**	-1.01
t-p2	1106.70	**1105.81**	-0.08	1106.84	0.01	**1105.81**	-0.08
t-p3	469.16	470.77	0.34	**467.42**	-0.37	**466.71**	-0.52
t-p4	554.20	556.68	0.45	**552.39**	-0.33	**549.05**	-0.93
t-p5	1384.75	1388.35	0.26	**1384.58**	-0.01	**1384.05**	-0.05
t-p6	643.59	666.16	3.51	652.65	1.41	645.65	0.32
t-p7	646.65	661.82	2.35	649.17	0.39	**644.53**	-0.33
t-p8	1633.92	**1618.91**	-0.92	**1615.51**	-1.13	**1614.39**	-1.20
t-p9	733.13	747.74	1.99	**729.33**	-0.52	**723.08**	-1.37
t-p10	1240.01	1247.37	0.59	**1237.72**	-0.18	**1235.01**	-0.40
t-p11	**490.97**	**490.97**	0.00	**490.97**	0.00	**490.97**	0.00
t-p12	**664.10**	**664.1**	0.00	**664.1**	0.00	**664.1**	0.00
t-p13	**830.80**	**830.8**	0.00	**830.8**	0.00	**830.8**	0.00
t-p14	**994.60**	**994.6**	0.00	**994.6**	0.00	**994.6**	0.00
t-p15	**1157.07**	1157.09	0.00	**1157.07**	0.00	**1157.07**	0.00
t-p16	**660.12**	660.52	0.06	**660.12**	0.00	**660.12**	0.00
t-p17	**776.43**	778.82	0.31	776.71	0.04	**776.43**	0.00
t-p18	**873.70**	879.8	0.70	875.82	0.24	874.42	0.08
t-p19	**958.51**	964.61	0.64	965.54	0.73	960.69	0.23
t-p20	**1033.58**	1045.45	1.15	1035.51	0.19	1035.27	0.16
t-p21	**1375.07**	**1375.08**	0.00	**1375.07**	0.00	**1375.07**	0.00
t-p22	4319.72	**4312.33**	-0.17	**4312.31**	-0.17	**4312.31**	-0.17
t-p23	8390.53	**8384.88**	-0.07	**8349.26**	-0.49	**8315.45**	-0.89
t-p24	**2064.84**	2065.03	0.01	**2064.84**	0.00	**2064.84**	0.00
t-p25	3231.50	**3211.93**	-0.61	**3208.49**	-0.71	**3207.88**	-0.73
t-p26	4084.75	**4041.66**	-1.05	**4045.73**	-0.96	**4033.36**	-1.26
t-p27	4621.36	**4554.13**	-1.45	**4547.77**	-1.59	**4545.29**	-1.65
t-p28	4682.54	**4635.26**	-1.01	**4628.24**	-1.16	**4622.16**	-1.29
t-p29	5595.45	**5542.09**	-0.95	**5529.68**	-1.18	**5527.39**	-1.22
t-p30	4453.15	**4443.23**	-0.22	**4436.31**	-0.38	**4435.39**	-0.40
t-p31	5405.40	**5375.4**	-0.56	**5370.59**	-0.64	**5368.45**	-0.68
t-p32	7346.32	**7259.14**	-1.19	**7244.02**	-1.39	**7238.45**	-1.47
t-p33	8394.52	**8242.78**	-1.81	**8216.48**	-2.12	**8208.38**	-2.22
average			0.05		-0.34		-0.52

Table 13: Best known solutions for PTSP instances, found with different parameter settings

Instance	BKS PTSP	BKS VNS	%
t-p1	**432.1**	**432.10**	0.00
t-p2	**1105.81**	**1105.81**	0.00
t-p03	**466.71**	**466.71**	0.00
t-p4	**549.05**	**549.05**	0.00
t-p5	**1382.33**	**1382.33**	0.00
t-p6	**643.5**	**643.50**	0.00
t-p7	**643.8**	**643.80**	0.00
t-p8	1613.42	**1611.96**	-0.09
t-p9	721.24	**720.72**	-0.07
t-p10	1237.77	**1233.53**	-0.34
t-p11	**490.97**	**490.97**	0.00
t-p12	**664.1**	**664.10**	0.00
t-p13	**830.8**	**830.80**	0.00
t-p14	**994.6**	**994.60**	0.00
t-p15	**1157.07**	**1157.07**	0.00
t-p16	**660.12**	**660.12**	0.00
t-p17	**776.43**	**776.43**	0.00
t-p18	**873.73**	**873.73**	0.00
t-p19	**958.51**	**958.51**	0.00
t-p20	**1033.58**	**1033.58**	0.00
t-p21	**1375.07**	**1375.07**	0.00
t-p22	**4312.31**	**4312.31**	0.00
t-p23	**8308.51**	**8308.48**	0.00
t-pr01	**2064.84**	**2064.84**	0.00
t-pr02	3207.44	**3205.94**	-0.05
t-pr03	4030.54	**4027.71**	-0.07
t-pr04	4558.94	**4538.19**	-0.46
t-pr05	4628.89	**4613.58**	-0.33
t-pr06	5534.94	**5521.24**	-0.25
t-pr07	**4435.39**	**4435.39**	0.00
t-pr08	5376.11	**5366.53**	-0.18
t-pr09	7282.39	**7234.35**	-0.66
t-pr10	8280.07	**8199.55**	-0.97

Table 14: Use of different neighborhoods with different ordering of neighborhood structures

N_k	move-CROSS-cc	CROSS-cc-move	cc-move-CROSS
1	m 1551	cr 1688	cc 1845
2	m 942	cr 936	cc 1249
3	m 622	cr 532	cc 820
4	cr 421	cr 311	cc 539
5	cr 276	cr 195	cc 363
6	cr 174	cr 121	cc 260
7	cr 113	cc 703	m 155
8	cr 77	cc 506	m 107
9	cr 54	cc 345	m 71
10	cc 523	cc 254	cr 54
11	cc 396	cc182	cr 36
12	cc 276	cc 132	cr 25
13	cc 198	m 20	cr 17
14	cc 138	m 17	cr 15
15	cc 115	m 11	cr 9
avg. dev. to best solution	3.64	0.98	0.05

Table 15: Use of different neighborhoods with cc-move-cross for PVRP and PTSP

	PVRP			PTSP		
N_k	10^6	10^7	10^8	10^6	10^7	10^8
1	3006	3389	3673	1845	1734	1915
2	1962	2167	2442	1249	1221	1281
3	1429	1623	1746	820	801	833
4	1062	1195	1334	539	520	528
5	835	947	1039	363	361	368
6	665	770	847	260	257	259
7	395	456	512	155	170	177
8	296	337	389	107	122	123
9	259	296	336	71	79	81
10	314	365	451	54	56	60
11	235	260	320	36	42	41
12	189	219	261	25	28	28
13	167	178	227	17	16	18
14	147	168	204	15	13	15
15	145	160	188	9	10	6

5 Conclusion

We presented an (almost) generic Variable Neighborhood Search heuristic for the Periodic Vehicle Routing Problem and the Periodic Traveling Salesman Problem that is competitive or even outperforms the existing methods. The main features of this algorithm are a simple and flexible local search as well as an acceptance criterion for neighboring solutions inspired by Simulated Annealing. We show the robustness of our approach by applying the identical basic algorithm to both problem classes. The only difference between the implementations for PVRP and PTSP is the choice of the local search. We also made no special efforts to adapt the algorithm to degenerated problem instances (i.e. those without any periodic aspects). Nevertheless, the results obtained through an extensive numerical analysis showed that the algorithm is competitive to other state of the art approaches applied to these problem classes. Considering the best solutions found our algorithm for the PVRP outperforms the existing techniques by finding 24 new best solutions and 13 ties. In detail, we improved 9 test instances of the 10 instances of the new data and 15 test instances out of the 32 instances of the old data. The strength of our algorithm is that on average it provides better results than the existing techniques especially when the problem size increases. Another important aspect with respect to runtime is that the algorithm scales quite well. The increase in runtime is much lower when the problem size increases compared to the other algorithms in the literature. For the PTSP our VNS finds 11 new best results on the existing instances in the literature. Moreover also the average results resemble or even outperform the results published in the literature within a comparable runtime.

For future research we will extend our algorithm to be applicable to Inventory Routing Problems. This algorithm will then be applied to the periodic delivery of blood products to hospitals (see Hemmelmayr et al., 2006).

Acknowledgments

Financial support from the Oesterreichische Nationalbank (OeNB) by grant #11187 and from the Austrian Science Fund (FWF) by grant #L286-N04 is gratefully acknowledged.

References

[1] Alegre J, Laguna M, Pacheco J. Optimizing the periodic pick-up of raw materials for a manufacturer of auto parts. European Journal of Operational Research 2007; 179; 736–746.

[2] Angelelli E, Speranza MG. The periodic vehicle routing problem with intermediate facilities. European Journal of Operational Research 2002; 137(2); 233–247.

[3] Beltrami EJ, Bodin LD. Networks and vehicle routing for municipial waste collection. Networks 1974; 4; 290–306.

[4] Bertazzi L, Paletta G. Speranza MG. An improved heuristic for the period traveling salesman problem. Computers & Operations Research 2004; 31; 1215–1222.

[5] Blakeley F, Bozkaya B, Cao B, Hall W, Knolmajer J. Optimizing periodic maintenance operations for Schindler elevator corporation. Interfaces 2003; 33(1); 67–79.

[6] Braysy O. A Reactive Variable Neighborhood Search for the Vehicle-Routing Problem with Time Windows. INFORMS Journal on Computing 2003; 15; 347–368.

[7] Carter MW, Farvolden JM, Laporte G, Xu J. Solving an integrated logistics problem arising in grocery distribution. INFOR 1996; 34; 290–306.

[8] Chao IM, Golden BL, Wasil EA. An improved heuristic for the period vehicle routing problem. Networks 1995a; 26; 25–44.

[9] Chao IM, Golden BL, Wasil EA. A new heuristic for the period traveling salesman problem. Computers & Operations Research 1995b; 22; 553–565.

[10] Christofides N, Beasley JE. The period routing problem. Networks 1984; 14; 237–256.

[11] Clarke G, Wright JW. Scheduling of vehicles from a central depot to a number of delivery points. Operations Research 1964; 12; 568-81.

[12] Cordeau JF, Gendreau M, Laporte G. A Tabu Search Heuristic for Periodic and Multi-Depot Vehicle Routing Problems. Networks 1997; 30; 105-119.

[13] Croes GA. A method for solving traveling salesman problems. Operations Research 1958; 6; 791–812.

[14] Dongarra JJ. Performance of Various Computers Using Standard Linear Equations Software, Technical Report of the Computer Science Department, University of Tennessee, Knoxville, 2005.

[15] Dueck G. New optimization heuristics: The great deluge algorithm and the record-to-record travel. Journal of Computational Physics 1993; 104; 86-92.

[16] Drummond LMA, Ochi LS, Vianna DS. An asynchronous parallel metaheuristic for the period vehicle routing problem. Future Generation Computer Systems 2001; 17; 379–386.

[17] Eilon S, Gandy W, Christofides N. Distribution Management: Mathematical Modeling and Practical Analysis. Griffin: London; 1971.

[18] Gaudioso M, Paletta G. A heuristic for the periodic vehicle routing problem. Transportation Science 1992; 26(2); 86–92.

[19] Gendreau M, Hertz A, Laporte G. New insertion and postoptimization procedures for the traveling salesman problem. Operations Research 1992; 40; 1086–1094.

[20] Golden BL, Wasil EA. Computerized vehicle routing in the soft drink industry. Operations Research 1987; 35; 6–17.

[21] Hadjiconstantinou E, Baldacci R. A multi-depot period vehicle routing problem arising in the utilities sector. Journal of the Operational Research Society 1998; 49(12); 1239–1248.

[22] Hansen P, Mladenović N. Variable Neighborhood Search for the p-median. Location Science 1997; 5; 207–226.

[23] Hansen P, Mladenović N. Variable Neighborhood Search In: Pardalos PM, Resende MGC (Eds), Handbook of Applied Optimization, Oxford University Press:New York; 2000. p. 221–234.

[24] Hansen P, Mladenović N. Variable Neighborhood Search: Principles and Applications. European Journal of Operational Research 2001; 130; 449–467.

[25] Hemmelmayr V, Doerner KF, Hartl RF, Savelsbergh M. Delivery Strategies for Periodic Blood Supplies. submitted, 2006.

[26] Kindervater GAP, Savelsbergh MWP. Vehicle routing: Handling edges exchanges windows. In Aarts E, Lenstra JK (Eds), Local Search in Combinatorial Optimization, Wiley:Chichester; 1997. p. 337-360.

[27] Kirkpatrick S, Gelatt CD, Vecchi MP. Optimization by Simulated Annealing. Science 1983; 220(4598); 671–680.

[28] Lin S. Computer Solutions of the Traveling Salesman Problem. Bell Systems Technical Journal 1965; 44; 2245-2269.

[29] Mladenović, N. A variable neighbourhood algorithm - a new metaheuristic for combinatorial optimization. Abstract of Papers presented at Optimization Days 1995, 112.

[30] Mladenović N, Hansen P. Variable Neighborhood Search. Computers & Operations Research 1997; 24; 1097–1100.

[31] Paletta G. A multiperiod traveling salesman problem: heuristic algorithms. Computers & Operations Research 1992; 19; 789–795.

[32] Paletta G. The period traveling salesman problem: a new heuristic algorithms. Computers & Operations Research 2002; 29; 1343–1352.

[33] Polacek M, Hartl RF, Doerner KF, Reimann M. A Variable Neighborhood Search for the Multi Depot Vehicle Routing Problem with Time Windows. Journal of Heuristics 2004; 10; 613–627.

[34] Polacek M, Doerner KF, Hartl RF, Kiechle G, Reimann M. Scheduling periodic customer visits for a Travelling Salesperson. European Journal of Operational Research 2007; 179; 823-837.

[35] Russel RA, Gribbin D. A multiphase approach to the period routing problem. Networks 1991; 21; 747–765.

[36] Russel RA, Igo W. An assignment routing problem. Networks 1979; 9; 1–17.

[37] Tan CCR, Beasley JE. A heuristic algorithm for the periodic vehicle routing problem. Omega 1984; 12(5); 497–504.

[38] Van Breedam A. An Analysis of the behaviour of heuristics for the vehicle routing problem for a seletion of problems with vehicle-related, customer-related and time-related constraints. Ph.D. dissertation, University of Antwerp, 1994.