A Variable Neighborhood Search for the Periodic Vehicle Routing Problem with Time Windows[*]

Sandro Pirkwieser Günther R. Raidl

Institute of Computer Graphics and Algorithms,
Vienna University of Technology
Favoritenstrasse 9-11/1861, 1040 Vienna, Austria
{pirkwieser|raidl}@ads.tuwien.ac.at

1 Introduction

Transportation problems appear in many practically highly relevant areas of our daily life. In this work we deal with a generalized variant of the classical Vehicle Routing Problem (VRP), namely the *Periodic Vehicle Routing Problem with Time Windows* (PVRPTW), for which only few specific solution techniques have been described in the literature. The basis forms the VRP which is defined on a complete graph $G = (V, A)$, where $V = \{v_0, v_1, \dots v_n\}$ is the vertex set and $A = \{(v_i, v_j) : v_i, v_j \in V, i \neq j\}$ is the arc set. Vertex v_0 represents the depot at which are based m vehicles having capacities $Q_1, \dots, Q_m$. Each vertex of $V \setminus \{v_0\}$ corresponds to a customer and has an associated demand $q_i \geq 0$ as well as a service duration $d_i \geq 0$. For each arc $(v_i, v_j) \in A$ there are further given travel times or costs $c_{ij} \geq 0$. The first generalization is due to additionally specifying a time window $[e_i, l_i]$ per customer and the depot, where e_i and l_i are nonnegative integers and denote the earliest as well as latest beginning of the service, respectively. The second generalization concerns the extension to a planning horizon of t days: Each customer has defined a service frequency f_i and a set C_i of allowable combinations of visit days, thus a customer has to be visited periodically. The whole PVRPTW then consists of selecting a single visit combination per customer and designing (at most) m vehicle routes on each of the t days on G such that

(1) each route starts and ends at the depot,

(2) each customer i belongs to f_i routes over the horizon,

(3) the total demand of the route for vehicle k does not exceed Q_k,
and its total duration does not exceed a preset value D_k,

(4) the service at customer i begins in the interval $[e_i, l_i]$ and
every vehicle leaves the depot and returns to it in the interval $[e_0, l_0]$, and

(5) the total travel cost of all vehicles is minimized.

[*]This work is funded by the Austrian Science Fund (FWF) under contract number P20342-N13.

In the following we will report on our work of solving the PVRPTW with a Variable Neighborhood Search (VNS) metaheuristic. In Section 2 we refer to related work. All parts composing the VNS are reported in Section 3. Comparative results are given in Section 4 and concluding remarks in Section 5.

2 Related Work

Although numerous (meta-)heuristics have been described for the classical version of the VRP [3], and also the VRP with Time Windows (VRPTW) is covered to quite an extent [2], the PVRPTW is to our knowledge only dealt with in the works by Cordeau et al. [4, 5]. They developed a Tabu Search (TS) heuristic and applied it to the VRPTW and two generalizations, among those also the PVRPTW. Related VNS metaheuristics were described by Polacek et al. [10] for the Multi-Depot VRPTW and by Hemmelmayr et al. [8] for the Periodic VRP. Our VNS for the PVRPTW adequately combines, adapts, and extends concepts of these works for considering time windows and periodicity at the same time.

3 Variable Neighborhood Search for the PVRPTW

The VNS metaheuristic [6] is a well-established heuristic search technique and was successfully applied to a plethora of different problems [7]. Diversification is accomplished by applying shaking, i.e. random moves in larger neighborhoods, while in the basic VNS a local search component is included for intensification. Below we report on the parts composing our VNS.

3.1 Penalized Cost Function

To help the VNS finding a good feasible solution we explicitly allow infeasible solutions during the search process, thus smoothing the search space, by relaxing conditions (3) and (4) of the problem definition. For a solution s, we denote the total travel cost by $c(s)$, the total violations of the load, duration, and time window constraints by $q(s)$, $d(s)$, and $w(s)$, respectively. While $q(s)$ and $d(s)$ are calculated on a route basis considering the values of Q_k and D_k, $w(s)$ is determined by $\sum_{i=1}^{n} \max\{0, a_i - l_i\}$, where a_i is the arrival time at customer i. The cost function is defined as $f(s) = c(s) + \alpha q(s) + \beta d(s) + \gamma w(s)$, where α, β, and γ are positive weights which could be adapted in a dynamic way during the search. However, according to preliminary tests we settled on a common fixed value of 100, which was also the case in [10].

3.2 Initial Solution

Our initialization method is based on those introduced in [4] with a few modifications. First we randomly choose a visit combination per customer, thus assigning customers to the days of the planning horizon. In case of Euclidean instances all customers are ordered according to the angle they make with the depot; ties are broken choosing the customer with the earlier center of its time window $(e_i + l_i)/2$ first. An arbitrary ordering can be used otherwise. Next a customer $j \in \{1, \dots, n\}$ is chosen at random. At most m routes per day are then constructed by running the following procedure for each day of the planning horizon:

1. Set $k \leftarrow 1$.
2. For each customer $i = j, j+1, \ldots, n, 1, \ldots, j-1$ do:
   (a) If the insertion of customer i into route k would result in the violation of load or duration constraints, set $k \leftarrow \min\{k+1, m\}$.
   (b) Insert customer i into route k so as to minimize the increase of the cost function.

Using this procedure only the last route of each day might violate load or duration constraints, whereas all routes might violate time window constraints.

3.3 Shaking

In our VNS we make use of three different neighborhood structures utilized in the shaking phase. For each of these structures we define six moves with increasing maximal perturbation size δ, hence resulting in a total of 18 shaking neighborhoods. A move chooses the amount of change randomly in the interval $[1, \delta]$.

(a) Change up to δ visit combinations randomly: Remove a customer from previous visit combination days and insert it in corresponding days of the new visit combination so as to minimize the increase of the cost function. It turned out to be beneficial to also allow "changing" the visit combination of customers offering only one combination, thus removing and inserting the customer on the same days. Due to the greedy insertion the latter operation can be regarded as a local search.

(b) Move a random segment of maximal length δ, if $\delta \in [1, 5]$, or bounded by the route size in case $\delta = 6$ from a route to another one, i.e. perform a customer relocation. Thereby reverse the segment with a small probability p_{rev}.

(c) Exchange two random segments of varying maximal length (as in the previous move operator) between two routes, performing a CROSS exchange move. Reverse the segments with a small probability p_{rev}, occasionally performing an iCROSS exchange move [1].

We set p_{rev} for both (b) and (c) to 0.1 according to preliminary tests.

3.3.1 Shaking Neighborhood Order

According to findings in [8] we considered one VNS variant with a fixed neighborhood order of change combination, move segment, and finally exchange segments. The six moves of a specific neighborhood structure are arranged in increasing order according to δ.

In addition to this we investigated variants of adaptive orderings, since we believe an a priori fixed ordering might not fit well all instances and each phase of the heuristic search. In our experiments a random VNS (RVNS) outperformed all other variants. This RVNS starts with a random neighborhood ordering and generates a new one each time a full VNS iteration finishes.

3.4 Local Search Procedures

We apply local search on each tour changed during the shaking phase. Thereby we make use of the well-known 2-opt neighborhood, where a single move corresponds to exchanging two edges within a tour, inverting a segment. The neighborhood is searched in lexicographic order and the search is repeatedly applied until no more improvement is possible. Extensive tests revealed that the best improvement variant clearly outperforms a first improvement strategy, both in terms of solution quality and runtime.

Additionally each new incumbent solution is subject to a 2-opt* inter-route exchange heuristic [11]. Hereby for each pair of routes of the same day all possible exchanges of the routes' end segments are tried. Again this local search iterates until no further improvement is possible and all days are considered. For 2-opt* the first improvement version yielded slightly better results, so only these are reported in Section 4.

3.5 Acceptance Decision

To avoid that the VNS becomes too easily trapped in local optima, due to the cost function guiding towards feasible solutions and most likely complicating the escape of basins surrounded by infeasible solutions, we also allow to accept worse solutions under certain conditions. This is accomplished by utilizing a Metropolis criterion like in simulated annealing [9] for inferior solutions s' and accept them with a probability of $e^{-(f(s')-f(s))/T}$, depending on the cost difference to the actual solution s of the VNS process and the temperature T. We apply a linear cooling scheme and decrease T every τ_T iterations by an amount of $(T * \tau_T)/\tau_{\max}$, where $\tau_{\max}$ denotes the maximal VNS iterations. Preliminary tests showed a satisfying performance when setting $\tau_T = 100$ and using an initial temperature value of $T_0 = 10$.

4 Computational Results

The algorithm has been implemented in C++, compiled with GCC 4.1 and executed on a 2.2 GHz Dual-Core AMD Opteron 2214 PC with 4 GB RAM. All VNS variants are run 30 times per instance for 10^6 iterations and average results are reported. These Euclidean instances were introduced in [4], they range from 48 to 288 customers, 3 to 20 homogeneous vehicles, and have a planning horizon of 4 or 6 days. Instances 1a–10a and 1b–10b have narrow and larger time windows, respectively. Table 1 shows the results of VNS and RVNS without and with additional 2-opt* improvement as described. The second column states the previously best known solution values from the TS in [4]. For the VNS variants, best and average solution values (travel costs) as well as corresponding standard deviations are printed. Statistical Wilcoxon rank sum tests with an error-level of 5% confirm that RVNS often outperforms VNS, both without and with 2-opt*, and the variants utilizing 2-opt* outperform the corresponding ones without on many instances, especially in case of the normal VNS. All considered VNS variants consistently outperform the TS in terms of the best solution found. The last line denotes average CPU-times. They reveal that RVNS takes less than 10% longer than VNS (due to repeatedly generating random orderings as well as applying more costly shaking moves more frequently); using the 2-opt* improvement procedure incurs nearly no increase in runtime at all. In Table 2 we additionally give the relative percentage deviations (RPD) to the previously best known solutions.

4.1 Improved Route Evaluation

So far we implicitly assumed that each vehicle arrives as early as possible at the first customer j served by simply setting the waiting time at the depot to $\max\{e_0, e_j - c_{0j}\}$. An improved

Table 1: Comparison of final solution values of the TS from [4] and different VNS variants with 10^6 iterations.

No.	TS [4]	VNS			RVNS			VNS with 2-opt*			RVNS with 2-opt*		
	best	best	avg.	std.	best	avg.	std.	best	avg.	std.	best	avg.	std.
1a	3007.84	2989.58	3021.22	23.90	3000.97	3017.18	13.59	2989.58	3014.66	14.48	2989.58	3011.41	10.53
2a	5328.33	5161.19	5269.02	81.33	5175.57	5264.30	57.77	5127.98	5262.48	72.61	5146.99	5240.45	59.99
3a	7397.10	7318.88	7440.17	62.66	7260.37	7395.76	70.35	7301.46	7414.25	73.30	7286.76	7391.74	58.16
4a	8376.95	8091.19	8242.71	65.99	8109.24	8218.47	63.12	8089.15	8205.00	70.97	8111.20	8211.57	47.10
5a	8967.90	8765.19	8895.04	73.42	8754.70	8897.47	106.92	8760.44	8870.79	67.32	8723.63	8840.93	68.52
6a	11686.91	11209.84	11323.50	70.66	11121.67	11313.85	111.59	11063.00	11281.67	87.08	11106.63	11255.80	76.96
7a	6991.54	6940.25	7026.89	42.88	6918.53	7013.80	53.16	6926.58	7033.89	56.08	6917.71	7009.86	51.16
8a	10045.05	9935.41	10067.34	70.00	9879.48	10038.22	68.20	9905.87	10055.57	68.83	9854.36	9998.75	53.58
9a	14294.97	13917.87	14238.18	123.64	13941.09	14159.34	101.71	13891.03	14105.26	108.49	13978.65	14129.01	71.15
10a	18609.72	18198.57	18484.32	110.50	18120.17	18407.11	134.36	18130.46	18369.76	116.09	18023.62	18362.83	110.74
1b	2318.37	2289.17	2289.92	1.73	2289.17	2289.54	1.13	2289.17	2291.29	5.73	2289.21	2290.02	1.66
2b	4276.13	4167.07	4239.47	32.79	4151.49	4213.42	43.10	4180.12	4238.65	35.48	4149.96	4221.31	39.67
3b	5702.07	5649.84	5721.65	37.38	5649.09	5711.27	55.32	5611.62	5705.25	41.98	5608.67	5704.53	61.81
4b	6789.73	6554.15	6658.90	49.35	6541.23	6649.30	42.44	6589.06	6656.73	42.52	6534.12	6633.67	54.37
5b	7102.36	7017.71	7143.51	63.16	7016.89	7119.15	55.09	7021.54	7146.39	65.42	6995.87	7111.50	61.57
6b	9180.15	8947.97	9019.76	44.14	8955.06	9041.25	52.28	8914.71	8999.79	44.26	8895.31	9024.72	54.87
7b	5606.08	5522.76	5595.78	34.44	5517.71	5587.57	34.42	5525.67	5576.99	24.15	5522.85	5579.25	27.29
8b	7987.64	7712.40	7921.17	65.48	7795.23	7880.96	54.95	7732.15	7887.61	83.12	7808.49	7882.10	50.62
9b	11089.91	11034.16	11194.49	78.77	10973.52	11125.45	62.99	11050.61	11136.13	72.75	10944.59	11085.97	70.57
10b	14207.64	14170.19	14362.18	102.60	14065.16	14310.99	106.21	14143.15	14330.41	96.90	14076.87	14293.03	121.86
Avg.	8448.32	8279.67	8407.76	61.74	8261.82	8382.72	64.43	8262.17	8379.13	62.38	8248.25	8363.92	57.61
t[s]	-	83.78			91.69			85.07			92.65		

Table 2: Improvements of (R)VNS w.r.t. the best known solutions from [4] and [5].

No.	TS [4]	(R)VNS	RPD(%)	using forward time slack [12] TS [5]	(R)VNS	RPD(%)
1a	3007.84	2989.58	-0.61	2911.03	2909.02	-0.07
2a	5328.33	5127.98	-3.76	5055.05	5036.27	-0.37
3a	7397.10	7260.37	-1.85	7229.73	7138.70	-1.26
4a	8376.95	8089.15	-3.44	7953.08	7882.06	-0.89
5a	8967.90	8723.63	-2.72	8593.00	8492.45	-1.17
6a	11686.91	11063.00	-5.34	10927.45	10713.75	-1.96
7a	6991.54	6917.71	-1.06	6825.07	6787.72	-0.55
8a	10045.05	9854.36	-1.90	9748.36	9721.25	-0.28
9a	14294.97	13891.03	-2.83	13614.47	13463.96	-1.11
10a	18609.72	18023.62	-3.15	17735.59	17650.89	-0.48
Avg.	9470.63	9194.04	-2.66	9059.28	8979.61	-0.81
1b	2318.37	2289.17	-1.26	2294.03	2277.44	-0.72
2b	4276.13	4149.96	-2.95	4257.40	4137.45	-2.82
3b	5702.07	5608.67	-1.64	5648.76	5575.27	-1.30
4b	6789.73	6534.12	-3.76	6594.54	6476.67	-1.79
5b	7102.36	6995.87	-1.50	7054.95	6970.33	-1.20
6b	9180.15	8895.31	-3.10	8928.37	8819.32	-1.22
7b	5606.08	5517.71	-1.58	5505.23	5504.67	-0.01
8b	7987.64	7712.40	-3.45	7875.31	7729.32	-1.85
9b	11089.91	10944.59	-1.31	10889.77	10885.93	-0.04
10b	14207.64	14065.16	-1.00	13980.55	13943.61	-0.26
Avg.	7426.01	7271.30	-2.16	7302.89	7232.00	-1.12
Avg.	8448.32	8232.67	-2.41	8181.09	8105.80	-0.97

method involves the concept of forward time slack introduced in [12]. This was already applied to the PVRPTW by the TS heuristic [5]. The idea is to postpone leaving the depot as late as possible without increasing the time window violation, resulting in minimization of the route duration and probably rendering previously infeasible routes and thus solutions feasible. Due to limited space we cannot present all results, but we state at least the RPD with regard to the previously best known solutions, shown in the right half of Table 2. Since the different VNS variants exhibit a very similar behavior than before, the relative performance differences stay the same.

5 Conclusions

We presented a Variable Neighborhood Search (VNS) metaheuristic for the so far barely treated Periodic Vehicle Routing Problem with Time Windows (PVRPTW), a generalized variant of the classical VRP. For this we considered other successful VNS solution approaches developed for similar problems and combined and adapted some of their concepts. In addition we found out that a random VNS often yielded significantly better results than a VNS using a reasonable fixed ordering of the shaking neighborhoods. Furthermore, a selectively applied simple inter-route improvement procedure, 2-opt*, was shown to considerably improve both

VNS variants at nearly no computational cost at all. Finally, the good performance of our method is demonstrated when comparing our best solution values to the previously best known solution values of a Tabu Search heuristic, yielding an average improvement of around 1–3%, and in some cases even more.

References

[1] O. Bräysy. A reactive variable neighborhood search for the vehicle-routing problem with time windows. *INFORMS Journal on Computing*, 15(4):347–368, 2003.

[2] O. Bräysy and M. Gendreau. Vehicle routing problem with time windows, part II: Metaheuristics. *Transportation Science*, 39(1):119–139, 2005.

[3] J.-F. Cordeau, M. Gendreau, A. Hertz, G. Laporte, and J.-S. Sormany. New heuristics for the vehicle routing problem. In A. Langevin and D. Riopel, editors, *Logistics Systems: Design and Optimization*, pages 279–297. Springer US, 2005.

[4] J.-F. Cordeau, G. Laporte, and A. Mercier. A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational Research Society*, 52:928–936, 2001.

[5] J.-F. Cordeau, G. Laporte, and A. Mercier. 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(5):542–546, 2004.

[6] P. Hansen and N. Mladenović. Variable neighborhood search. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, pages 145–184. Kluwer Academic Publishers, Boston MA, 2003.

[7] P. Hansen, N. Mladenović, and J. A. M. Pérez. Variable neighborhood search: Methods and applications. *Les Cahiers du GERAD* G-2008-39, HEC Montreal, Montreal, Canada, 2008.

[8] V. Hemmelmayr, K. Doerner, and R. Hartl. A variable neighborhood search for periodic routing problems. *European Journal of Operational Research*, 2007. (to appear).

[9] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[10] M. Polacek, R. F. Hartl, K. Doerner, and M. Reimann. A variable neighborhood search for the multi depot vehicle routing problem with time windows. *Journal of Heuristics*, 10:613–627, 2004.

[11] J.-Y. Potvin and J.-M. Rousseau. An exchange heuristic for routeing problems with time windows. *Journal of the Operational Research Society*, 46:1433–1446, 1995.

[12] M. W. P. Savelsbergh. The vehicle routing problem with time windows: Minimizing route duration. *ORSA Journal on Computing*, 4:146–154, 1992.