Boosting a Variable Neighborhood Search for the Periodic Vehicle Routing Problem with Time Windows by ILP Techniques[*]

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

The *periodic vehicle routing problem with time windows* (PVRPTW) is a generalized variant of the classical *vehicle routing problem with time windows* (VRPTW) where customers must be served several times in a given planning period instead of only once on a single day. Such a setting occurs in many real-world applications as in courier services, grocery distribution or waste collection.

The PVRPTW is defined on a complete directed graph $G = (V, A)$, where $V = \{0, 1, \ldots, n\}$ is the vertex set and $A = \{(i, j) : i, j \in V, i \neq j\}$ is the arc set. The considered planning horizon shall be t days, also referred to as $T = \{1, \ldots, t\}$. Vertex 0 represents the depot with time window $[e_0, l_0]$ at which are based m vehicles having capacities $Q_1, \ldots, Q_m$ and maximal daily working times $D_1, \ldots, D_m$. Each vertex $i \in V_C$, with $V_C = V \setminus \{0\}$, corresponds to a customer and has an associated demand $q_i \geq 0$, a service duration $d_i \geq 0$, a time window $[e_i, l_i]$, a service frequency f_i and a set C_i of allowable combinations of visit days. Finally, each arc $(i, j) \in A$ is assigned a travel time (cost) $c_{ij} \geq 0$. The problem 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 planning horizon,
3. the total daily demand of the route for vehicle k does not exceed capacity limit Q_k, and its daily duration does not exceed the maximal daily working time D_k,
4. the service at each 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.

Arriving before e_i at customer i incurs a waiting time at no additional cost, arriving later than l_i is not allowed, i.e. we assume "hard time windows". In the remainder we further assume a homogeneous vehicle fleet, i.e. $Q_1, \ldots, Q_m = Q$ and $D_1, \ldots, D_m = D$.

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

In this work we exploit the capabilities of a suitable *integer linear programming* (ILP) formulation of the problem solved with a generic ILP solver to boost the performance of a variable neighborhood search metaheuristic. We refer to related work in Section 2. The variable neighborhood search is described in Section 3, an ILP formulation in Section 4, and the proposed hybrid method in Section 5. Experimental results are given in Section 6, and finally, Section 7 finishes the work with concluding remarks.

2 Related Work

The PVRPTW was first mentioned in [1], where a tabu search algorithm is described for it. In our previous work [9] we applied a variable neighborhood search (VNS), outperforming the former tabu search. Related VNS metaheuristics exist for the Multi-Depot VRPTW [10] and the Periodic VRP (PVRP) [6]. We are not aware of other exact or hybrid methods for the PVRPTW, yet similar PVRPs are dealt with in [3] and [8]. A more general survey of different PVRP variants and solution methods is given in [4]. A similar approach as the one followed in this work was recently applied for ready-mixed concrete delivery [13]. Our work also extends this by highlighting further aspects of this kind of hybridization. In a somewhat related work Danna et al. [2] apply an ILP solver for deriving (better) integer solutions during a branch-and-price procedure. Finally, the emerging field of hybrid methods is covered in [12].

3 VNS for the PVRPTW

Variable neighborhood search (VNS) [5] is a metaheuristic applying random steps in neighborhoods with growing size for diversification, referred to as shaking, and using a local search component for intensification. In the following we will describe our VNS for the PVRPTW in short; see [9] for more details.

The VNS uses a penalized cost function to smooth the search space, taking into account the excess of vehicle load, route duration, and time window violation. The creation of the initial solution was kept quite simple by selecting a visit combination per customer at random and afterwards partitioning the customers—according to the angle they make with the depot—at each day into routes, only allowing load or duration constraints to be violated for the last route on each day, whereas all routes might violate time window constraints.

In the shaking phase we utilize three different neighborhood structures, each with six moves of increasing perturbation size: (i) randomly changing several visit combinations with greedy insertion for the new visit days, (ii) moving a random segment of a route to another one on the same day, and (iii) exchanging two random segments between two routes on the same day. In the latter two cases the segments are occasionally reversed. Here we only consider a fixed shaking neighborhood order.

For intensification we apply the well-known 2-opt intra-route exchange procedure in a best improvement fashion, only considering routes changed during shaking. 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.

To enhance the overall VNS performance not only better solutions are accepted, but sometimes also solutions having a worse objective value. This is done in a systematic way using a Metropolis criterion like in simulated annealing [7]. We use a linear cooling scheme such that the acceptance rate of worse solutions is nearly zero in the last iterations.

4 Set Covering ILP Model for the PVRPTW

The prerequisite for hybridizing the VNS described in Section 3 with ILP based techniques is a suitable formulation of the problem facilitating the desired combination.
We express the PVRPTW by the following set covering model:

$$\min \sum_{\tau \in T} \sum_{\omega \in \Omega} \gamma_\omega \, \chi_{\omega\tau} \tag{1}$$

$$\text{s.t.} \quad \sum_{r \in C_i} y_{ir} \geq 1 \qquad \forall i \in V_C \tag{2}$$

$$\sum_{\omega \in \Omega} \chi_{\omega\tau} \leq m \qquad \forall \tau \in T \tag{3}$$

$$\sum_{\omega \in \Omega} \alpha_{i\omega} \, \chi_{\omega\tau} - \sum_{r \in C_i} \beta_{ir\tau} \, y_{ir} \geq 0 \qquad \forall i \in V_C;\ \forall \tau \in T \tag{4}$$

$$y_{ir} \in \{0,1\} \qquad \forall i \in V_C;\ \forall r \in C_i \tag{5}$$

$$\chi_{\omega\tau} \in \{0,1\} \qquad \forall \omega \in \Omega;\ \forall \tau \in T \tag{6}$$

The set of all feasible routes (satisfying restrictions 1, 3, and 4 in Section 1), which is exponentially large, is denoted by Ω, and for each route $\omega \in \Omega$ its cost is γ_ω. Variable $\chi_{\omega\tau}$ is the number of times route ω is selected on day $\tau \in T$. For each customer $i \in V_C$, variables y_{ir} indicate whether or not visit combination $r \in C_i$ is chosen. Following constraints are used: Cover constraints (2) guarantee that at least one visit day combination is selected per customer, fleet constraints (3) restrict the number of daily routes to not exceed the available vehicles m, and finally visit constraints (4) link the routes and the visit combinations, whereas $\alpha_{i\omega}$ and $\beta_{ir\tau}$ are binary constants indicating whether or not route ω visits customer i and if day τ belongs to visit combination $r \in C_i$ of customer i, respectively.

In general, such an ILP formulation is the basis of a column generation approach, where one starts with a small set of initial columns (routes) and gradually increases it by adding potentially improving columns to determine a valid lower bound. This can be combined with branch-and-bound to derive integer solutions, too (and eventually prove their optimality). However, this is another line of research followed by us which turns out to be applicable to relatively small instances only. Here, we use the VNS as the sole provider of columns for the set covering model, which is then solved via a generic ILP solver. This hybrid method is explained in the next section.

5 VNS and ILP Hybrid

In the following the ILP model of the preceding section is used to boost the performance of the VNS of Section 3. This is achieved by introducing feasible VNS solutions into the set covering model, i.e. by adding the single routes of these solutions as new columns. This way the feasibility of the model is guaranteed. The resulting ILP is solved by a (basically) branch-and-bound based generic ILP solver, gradually fixing the visit combination (5) and route variables (6). A similar approach was

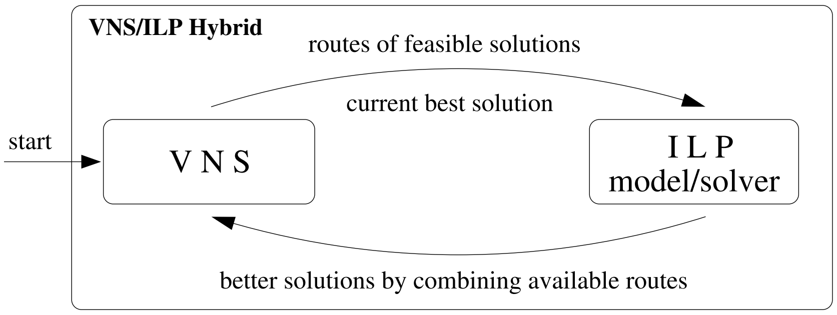
VNS/ILP Hybrid
routes of feasible solutions
current best solution
start
V N S
I L P
model/solver
better solutions by combining available routes

Figure 1: Information exchange between VNS and the ILP model/solver.

introduced in [13], where the authors highlight the “global view” property of such an exact model. For a set of solutions’ routes the ILP solver might be able to derive a more favorable combination and provide a better (less costly) solution in this way. Obviously the potential of the ILP solver depends on the routes contained in the model since it is neither able to alter routes nor create ones on its own. Hence it is crucial to provide (i) a suitable amount of routes, (ii) cost-effective routes, and (iii) diverse enough routes. Adding not enough or only weak routes might prevent finding a better solution at all, on the other hand a too large set naturally increases the runtime, which might also prevent finding better solutions quickly enough in case a time limit is given. Therefore solutions should not be added arbitrarily to the model. Only finding new combinations of routes constituting a feasible solution is not sufficient, the solution should also improve on the current incumbent in terms of travel costs. This is dealt with by primarily adding improved and feasible solutions found by the VNS (i.e. solutions that improved on the current best solution at the time they were derived), as it is done in [13]. However, we found it often not enough to solely add such improved solutions, since in case the VNS gets stuck for a while no improved solutions are available at all and no further solving of the ILP is meaningful. In order to be able to still exploit the power of the ILP solver we propose to further add some “intermediate” VNS solutions, i.e. feasible solutions derived in an iteration but not improving on the best solution. For ensuring a certain quality, we define a maximal deviation ε from the current best solution (at the time of checking), and avoid duplicates.

We apply the following hybrid scheme, which can be regarded an intertwined collaborative cooperation [12]; see Figure 1. The hybrid algorithm divides the execution of a VNS run in several equally long sections $\mathcal{S}$, and after each section the current ILP model is solved. The latter is gradually enriched by columns extracted from selected feasible VNS solutions. For this we choose a number n_{sol} of overall solutions to consider and first try to insert only the most current improved VNS solutions. In case less than n_{sol} such solutions are provided by the previous VNS section we select the remaining ones from the set of intermediate solutions (also accumulated during the previous VNS section) at random, in order to obtain a set of diverse solutions. Further, the ILP solver is always initialized with the current best solution to speed up the process. If the ILP solver is able to improve on the current best solution this new solution is transformed and transferred to the VNS. During transformation over-covered solutions are repaired by choosing exactly one visit combination (the first active) and omitting customers from following routes if they are already covered or do not need to be covered on this day. This over-covering might happen since we use a set covering model. In contrast, a set partitioning model (derived by turning inequalities (2) and (4) into equalities) would yield only feasible solutions but at the same time exclude many potentially improving combinations. Finally, when injecting this solution into the VNS it is also subject to the previously mentioned 2-opt* improvement procedure. In case routes were altered during these procedures, corresponding new columns are also added to the ILP model.

What we did not mention so far is the applied route injection scheme, which is also of concern in the classical column generation approach. Basically it is possible to either add the route for the corresponding day only or for all days. The latter scheme produces significantly larger ILP models (of factor t) which might yield better solutions at the expense of longer running times to solve the ILP model. We compare these alternatives in our experimental results. For both variants we allow the ILP solver the same overall amount of CPU-time as the VNS, though it is expected that the former variant consumes only a fraction of it.

6 Experimental Results

The algorithms have 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. We derived new PVRPTW instances from the Solomon VRPTW benchmark instances[1] by evenly assigning the possible visit combinations to the customers at random. We did so for the first five instances of type random (R), clustered (C), and mixed random and clustered (RC) for a planning horizon of four and six days. The number of vehicles m was altered (reduced) in such a way that few or none empty routes occur in feasible solutions, yet it is not too hard to find feasible solutions quite early in the solution process.
For the VNS we set an iteration limit of either 10^6 or $2 \cdot 10^6$, an initial temperature of 10 and apply linear cooling every 100 iterations. The VNS/ILP hybrid is also based on a VNS run with 10^6 iterations and $\mathcal{S}$ is set to 10, i.e. it applies ten sequences of 10^5 VNS iterations with a subsequent ILP solving phase each. For the latter we apply the sophisticated ILOG CPLEX 11.2 generic MIP solver, which potentially also adds cutting planes to speed up the process.

We performed tests with three settings of ε (0%, 5%, and 10%, whereas 0% implies that no intermediate VNS solutions are considered) and two settings of n_{sol} (5 and 10), denoted by VNS/ILP$_{n_{\mathrm{sol}},\varepsilon}$. The solutions' routes are mainly added for the corresponding day only (single day strategy VNS/ILP$^{\mathrm{s}}$), some experiments are done with insertion for all days (all days strategy VNS/ILP$^{\mathrm{a}}$). Each algorithm setting is run 30 times per instance and we report average results, stating the average travel costs (avg.), corresponding standard deviations (sdv.) and average CPU-times in seconds (t[s]). The results for the instances with a planning horizon of four ($t = 4$) and six days ($t = 6$) and a single day strategy are given in Table 1 and 2, respectively. Here we compare the performance of the mentioned hybrid variants to the VNS with 10^6 iterations. We mark all results of the hybrid variants with an underline that yield a significant improvement over the VNS, whereas we use a Wilcoxon rank sum test with an error level of 5%. Among the hybrid variants there is rarely a single one significantly outperforming all others, therefore we will compare the number of times the methods improved over the VNS. In case a hybrid variant needs considerably more CPU-time when compared to the VNS ($> 20\%$), we omit it from the row-wise comparison and put it in parentheses. As can be seen it is generally beneficial to also consider intermediate VNS solutions, yielding twice as many significant improvements. Regarding the quality of these solutions, there is a slight advantage when using better ones ($\varepsilon = 5\%$). While the performance of the hybrid method is nearly the same when adding 5 or 10 solutions at a time for instances having a planning horizon of four days, the performance notably degrades for a planning horizon of six days. This is mainly

[1]available at `http://web.cba.neu.edu/~msolomon/problems.htm`

Table 1: Results of VNS and VNS/ILP hybrids on periodic Solomon instances with a planning horizon of four days.

Instance	VNS (10^6)			VNS/ILP$^s_{5,0\%}$			VNS/ILP$^s_{5,5\%}$			VNS/ILP$^s_{5,10\%}$			VNS/ILP$^s_{10,5\%}$			VNS/ILP$^s_{10,10\%}$		
	avg.	sdv.	t[s]	avg.	sdv.	t[s]	avg.	sdv.	t[s]	avg.	sdv.	t[s]	avg.	sdv.	t[s]	avg.	sdv.	t[s]
p4r101	4146.36	20.00	29.66	4145.00	20.12	29.89	4124.41	26.96	32.61	4120.23	21.25	32.99	4112.11	18.70	33.60	4116.64	21.45	33.36
p4r102	3757.69	14.24	31.71	3751.65	13.86	32.43	3750.50	17.28	35.17	3746.69	12.74	35.24	3746.59	13.51	35.61	3742.11	9.51	36.00
p4r103	3197.80	13.25	32.30	3189.23	11.62	32.60	3188.55	13.43	35.80	3182.65	11.98	36.06	3186.97	10.79	37.44	3189.47	16.25	37.49
p4r104	2610.97	12.63	36.40	2613.31	14.06	36.89	2599.43	13.06	39.97	2602.90	15.10	39.97	2603.18	12.89	42.47	(2605.60	15.75	44.84)
p4r105	3697.77	17.44	31.50	3708.13	20.01	32.87	(3687.65	16.05	41.17)	(3681.62	15.19	40.42)	(3675.09	16.22	59.09)	(3680.50	20.57	57.21)
p4c101	2919.60	30.51	28.41	2910.65	0.52	29.04	2910.23	0.07	32.98	2910.23	0.11	33.12	2910.17	0.16	33.98	2910.90	3.94	33.80
p4c102	2965.59	39.10	32.97	2966.03	38.24	32.75	2951.60	35.15	35.29	2958.70	35.20	35.29	2958.09	37.02	35.47	2969.43	36.34	35.79
p4c103	2800.13	39.60	37.46	2803.67	36.22	38.11	2804.11	34.12	40.05	2804.70	35.37	39.93	2806.62	32.80	40.50	2813.96	41.50	40.63
p4c104	2483.52	23.97	36.06	2480.78	23.65	37.04	2473.77	19.26	39.16	2477.40	21.74	39.19	2478.59	24.24	41.80	2475.36	19.31	41.82
p4c105	2993.90	62.84	31.66	2998.87	48.60	31.80	3018.12	55.35	33.59	3006.50	48.83	33.60	2999.33	60.91	34.03	2990.91	59.02	34.17
p4rc101	4010.02	13.90	33.34	4005.68	17.78	34.05	3985.18	11.01	36.23	3985.42	12.51	36.05	3983.69	9.78	37.72	3982.46	14.43	37.42
p4rc102	3815.51	20.00	33.47	3809.55	22.06	34.05	3806.21	28.21	36.36	3797.11	16.26	36.26	3800.89	18.71	37.44	3804.03	32.14	37.36
p4rc103	3504.46	20.95	36.66	3499.57	21.17	36.17	3498.59	27.70	38.65	3499.13	29.05	38.83	3498.99	29.38	39.91	3496.25	27.83	39.69
p4rc104	3068.52	16.64	37.05	3054.05	19.02	37.79	3045.37	18.22	40.46	3052.38	16.27	40.31	3050.05	17.55	44.29	(3047.55	18.93	44.87)
p4rc105	4017.37	28.11	32.62	3998.70	30.43	32.99	3996.14	17.61	36.18	3995.71	21.62	36.20	4000.55	33.11	36.64	3994.90	26.07	36.57
sign. better than VNS				5× (33%)			10× (66%)			9× (60%)			10× (66%)			7× (46%)		

due to the too large run time when the ILP solver consumes more of the allotted time, leading to an exclusion of these variants for some instances (e.g. happening six times for $\text{VNS/ILP}^{s}_{10,[5\%,10\%]}$). However, for many variants and instances the overall execution time of the ILP solver itself is very short (a few seconds), often solving the ILP to optimality in fractions of a second. The remaining overhead compared to solely running the VNS is mainly due to the information exchange, especially for storing the intermediate VNS solutions and avoiding duplicates. It can be noted that the least improvement occurs for the clustered instances.

Now we will also have a look at the all days strategy, applied with $n_{\text{sol}} = 5$ and $\varepsilon = 5\%$, since this setting yielded good results in the former tests. To be fair, its performance must be compared to the VNS with $2 \cdot 10^6$ iterations, since its time consumption is basically bounded by taking twice the run time of the VNS with 10^6 iterations, plus the overhead for the information exchange. The results are shown in Table 3, alongside with the best performing single day strategy of each instance (see Table 1 and 2), whereas no variants need to be excluded due to too high time consumption this time. Again, significant improvements are underlined. It can be observed that the all days strategy is generally worse than the single day strategy, though still yielding a significant improvement for 11 of the 30 instances (36%). Most notably the best runs of the single day strategy improve the results in 14 cases (46%), although the CPU-time consumption is considerably less than that of both competitors. All in all one of the single day strategies either performs better or at least comparable to the VNS with $2 \cdot 10^6$ iterations.

7 Conclusions

We hybridized a VNS for the periodic vehicle routing problem with time windows (PVRPTW)—yielding satisfying results on its own—with a generic ILP solver applied to a proposed set covering ILP formulation of the problem. This formulation proved suitable for the desired combination, which can be classified as an intertwined collaborative cooperation where the VNS supplies the exact method with feasible solutions' routes and the current best solution, and the ILP solver takes a global view and seeks to determine better feasible route combinations. For testing we derived new PVRPTW instances from the Solomon VRPTW benchmark instances. Experimental results show the advantage of the hybrid approach, yielding for two third of all instances a statistically significant improvement over solely applying the VNS. It was further clearly beneficial to consider both improved and intermediate VNS solutions of a certain quality for enriching the ILP model. Generally, it is better to include not too much solutions to keep the runtime for solving the ILP small and increase the chance of finding an improved solution in limited time, as well as rather to include the solutions' routes for the corresponding day only, instead for all days. This way the hybrid method still performs significantly better for almost half of the instances even when compared to VNS runs with twice the iteration limit, additionally requiring considerably less CPU-time.

Further work could include better handling of intermediate VNS solutions (reducing overhead and probably increasing overall quality and diversity), some sort of column management for the ILP to be able to include more solutions (probably for all days) without performance degradation via discarding unpromising columns over time, or adding problem specific cuts to the ILP. It might also be possible to use such a hybrid method for finding initial feasible solutions by combining several partly-feasible ones. Based on the current algorithm, we could also investigate the effect of differing numbers of VNS/ILP sequences and ILP solver time limits, as was done in [13].

Table 2: Results of VNS and VNS/ILP hybrids on derived periodic Solomon instances with a planning horizon of six days.

Instance	VNS (10^6)			VNS/ILP$^s_{5,0\%}$			VNS/ILP$^s_{5,5\%}$			VNS/ILP$^s_{5,10\%}$			VNS/ILP$^s_{10,5\%}$			VNS/ILP$^s_{10,10\%}$		
	avg.	sdv.	t[s]	avg.	sdv.	t[s]	avg.	sdv.	t[s]	avg.	sdv.	t[s]	avg.	sdv.	t[s]	avg.	sdv.	t[s]
p6r101	5423.46	23.18	34.34	5414.87	19.56	35.13	5405.78	15.66	39.48	5403.15	16.26	39.70	5406.81	33.87	40.21	5395.27	11.92	40.66
p6r102	5275.77	19.14	36.15	5272.03	23.82	37.26	(5251.86	13.84	48.80)	(5245.82	17.96	46.26)	(5240.46	13.06	67.55)	(5237.75	9.99	66.22)
p6r103	4031.03	26.87	39.12	4023.50	29.41	39.69	4003.06	21.85	44.62	4002.98	16.75	45.14	(4001.86	20.65	55.62)	(4002.10	22.13	58.24)
p6r104	3393.07	15.28	40.12	3378.59	17.55	41.04	3372.56	12.23	47.85	3375.67	13.96	48.12	(3372.30	13.35	68.29)	(3374.17	11.84	70.73)
p6r105	4362.75	30.50	37.43	4349.95	23.36	38.02	4347.25	32.53	42.28	4341.64	24.98	43.95	(4334.60	26.79	50.00)	(4337.72	29.86	53.87)
p6c101	4072.01	38.11	39.08	4079.75	35.41	40.17	4070.44	40.33	41.56	4081.62	44.61	41.72	4097.04	40.32	42.63	4087.37	36.03	42.21
p6c102	3893.34	28.14	40.01	3889.46	21.10	40.46	3883.75	16.49	43.78	3888.12	20.54	43.99	3884.12	22.29	44.64	3891.87	24.27	45.29
p6c103	3609.70	38.95	45.69	3618.54	45.92	46.66	3600.07	42.33	49.60	3598.06	41.47	49.89	3611.73	47.27	52.04	3594.89	43.15	51.89
p6c104	3298.08	17.19	44.89	3294.79	20.49	45.29	3280.58	17.62	48.91	3290.76	20.18	49.88	(3283.59	19.40	67.16)	(3283.18	26.32	66.36)
p6c105	4145.25	58.37	39.06	4158.06	51.36	39.44	4172.46	65.96	42.70	4161.26	56.62	42.30	4167.56	60.96	43.63	4160.67	38.34	43.59
p6rc101	5845.46	24.02	36.17	5840.33	22.63	36.96	5832.27	18.42	41.18	5832.69	27.61	40.86	5824.73	19.31	43.31	5829.05	21.68	43.10
p6rc102	5487.18	24.34	37.70	5477.66	25.88	38.44	5468.09	29.44	42.23	5471.45	34.57	42.03	5463.50	26.65	44.17	5464.50	27.73	43.49
p6rc103	4382.03	26.68	42.94	4358.47	20.57	43.83	4356.26	21.69	46.72	4358.39	22.54	46.50	4344.02	19.31	48.71	4361.51	17.25	48.95
p6rc104	4142.07	23.93	41.71	4128.66	26.69	43.31	4128.70	20.94	47.65	4133.26	29.08	47.25	(4128.21	29.48	62.58)	(4122.25	21.93	60.63)
p6rc105	5343.05	24.38	38.54	5337.68	30.95	39.56	5328.62	24.46	42.05	5331.82	28.56	42.24	5321.55	20.96	42.96	5319.48	27.39	43.24
sign. better than VNS				5× (33%)			10× (66%)			9× (60%)			5× (33%)			6× (40%)		

Table 3: Results of VNS and VNS/ILP hybrids on derived periodic Solomon instances with a planning horizon of four and six days, inserting new routes for one or all days.

Instance	VNS $(2 \cdot 10^6)$			best VNS/ILP^s			VNS/ILP$^a_{5,5\%}$		
	avg.	sdv.	t[s]	avg.	sdv.	t[s]	avg.	sdv.	t[s]
p4r101	4127.42	19.24	59.45	$\underline{4112.11}$	18.70	33.60	$\underline{4095.28}$	12.71	59.48
p4r102	3756.41	16.24	63.33	$\underline{3742.11}$	9.51	36.00	$\underline{3738.38}$	8.71	54.25
p4r103	3193.08	14.46	64.81	$\underline{3182.65}$	11.98	36.06	$\underline{3181.41}$	9.45	65.93
p4r104	2603.00	11.99	72.81	2599.43	13.06	39.97	2599.15	14.72	72.66
p4r105	3696.73	13.12	62.92	$\underline{3675.09}$	16.22	59.09	$\underline{3691.70}$	12.25	67.25
p4c101	2916.43	24.05	56.72	$\underline{2910.17}$	0.16	33.98	$\underline{2910.39}$	0.99	34.50
p4c102	2949.99	29.82	65.52	2951.60	35.15	35.29	2940.16	27.49	51.93
p4c103	2805.30	37.45	75.01	2804.11	34.12	40.05	2804.47	39.61	67.21
p4c104	2473.83	13.67	72.56	2473.77	19.26	39.16	$\underline{2468.82}$	19.53	72.03
p4c105	2975.93	42.54	63.24	2990.91	59.02	34.17	$\underline{2957.54}$	44.84	44.84
p4rc101	3996.97	9.69	66.59	$\underline{3982.46}$	14.43	37.42	$\underline{3981.48}$	10.16	68.42
p4rc102	3796.21	16.17	66.80	3797.11	16.26	36.26	3796.19	25.14	54.99
p4rc103	3486.83	21.52	73.35	3496.25	27.83	39.69	3485.47	31.89	71.63
p4rc104	3053.40	16.92	74.50	$\underline{3045.37}$	18.22	40.46	3046.81	21.87	77.60
p4rc105	3995.62	16.23	65.36	3994.90	26.07	36.57	$\underline{3985.82}$	19.36	62.64
p6r101	5412.93	12.74	68.60	$\underline{5395.27}$	11.92	40.66	$\underline{5389.07}$	6.59	75.22
p6r102	5265.78	18.39	72.04	$\underline{5237.75}$	9.99	66.22	5267.80	21.54	75.26
p6r103	4024.17	26.05	78.19	$\underline{4001.86}$	20.65	55.62	4025.26	28.44	77.89
p6r104	3376.56	14.08	80.47	3372.30	13.35	68.29	3382.73	13.88	87.33
p6r105	4350.44	25.16	74.75	$\underline{4334.60}$	26.79	50.00	4360.48	31.13	80.01
p6c101	4064.57	38.93	77.96	4070.44	40.33	41.56	4087.26	48.51	82.01
p6c102	3880.12	20.69	79.83	3884.12	22.29	44.64	3877.56	19.98	84.85
p6c103	3601.82	48.59	91.46	3594.89	43.15	51.89	3604.09	45.23	96.55
p6c104	3292.57	16.47	89.86	$\underline{3280.58}$	17.62	48.91	3291.90	25.13	95.58
p6c105	4138.61	57.90	78.01	4158.06	51.36	39.44	4159.95	51.79	83.05
p6rc101	5829.66	17.54	73.37	5824.73	19.31	43.81	$\underline{5818.06}$	21.08	78.46
p6rc102	5474.10	28.45	75.65	5463.50	26.65	44.17	5467.22	34.58	81.23
p6rc103	4365.87	16.65	85.80	$\underline{4344.02}$	19.31	48.71	4367.08	21.05	91.13
p6rc104	4128.83	23.68	84.26	4122.25	21.93	60.63	4141.07	26.66	89.55
p6rc105	5329.82	22.16	77.16	$\underline{5319.48}$	27.39	43.24	5336.40	30.51	81.87
sign. better than VNS				14× (46%)			11× (36%)		

References

[1] 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.

[2] E. Danna and C. Le Pape. Branch-and-price heuristics: A case study on the vehicle routing problem with time windows. In G. Desaulniers et al., editors, *Column Generation*, chapter 4, pages 99–129. Springer, 2005.

[3] P. Francis, K. Smilowitz, and M. Tzur. The period vehicle routing problem with service choice. *Transportation Science*, 40(4):439–454, 2006.

[4] P. M. Francis, K. R. Smilowitz, and M. Tzur. The period vehicle routing problem and its extensions. In B. Golden et al., editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 73–102. Springer, 2008.

[5] 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.

[6] V. C. Hemmelmayr, K. F. Doerner, and R. F. Hartl. A variable neighborhood search heuristic for periodic routing problems. *European Journal of Operational Research*, 195(3):791–802, 2009.

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

[8] M. Mourgaya and F. Vanderbeck. Column generation based heuristic for tactical planning in multi-period vehicle routing. *European Journal of Operational Research*, 183(3):1028–1041, 2007.

[9] S. Pirkwieser and G. R. Raidl. A variable neighborhood search for the periodic vehicle routing problem with time windows. In C. Prodhon et al., editors, *Proceedings of the 9th EU/MEeting on Metaheuristics for Logistics and Vehicle Routing*, Troyes, France, 2008.

[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] G. R. Raidl and J. Puchinger. Combining (integer) linear programming techniques and metaheuristics for combinatorial optimization. In C. Blum, M. B. Aguilera, A. Roli, and M. Sampels, editors, *Hybrid Metaheuristics: An Emerging Approach to Optimization*, volume 114 of *Studies in Computational Intelligence*. Springer, 2008.

[13] V. Schmid, K. F. Doerner, R. F. Hartl, M. W. P. Savelsbergh, and W. Stoecher. A hybrid solution approach for ready-mixed concrete delivery. *Transportation Science*, 43(1):70–85, 2009.