Matheuristics for the Periodic Vehicle Routing Problem with Time Windows*

Sandro Pirkwieser and Günther R. Raidl

Institute of Computer Graphics and Algorithms
Vienna University of Technology, Vienna, Austria
{pirkwieser,raidl}@ads.tuwien.ac.at

Abstract. We investigate two matheuristic strategies using the periodic vehicle routing problem with time windows as a testbed. Two different metaheuristics are suitably combined with parts of a developed column generation approach: On the one hand a variable neighborhood search (VNS) acts as the sole provider of columns for a set covering model, hence realizing a pure metaheuristic column generation. Hereby, the VNS and the resolving of the model are performed in an intertwined way. On the other hand the solution to the linear programming (LP) relaxation of the set covering model, i.e. the columns (routes) and their respective (accumulated) LP values, found by a classical column generation approach are successfully exploited in a subsequent evolutionary algorithm. Both matheuristics often yield significantly better results than their pure metaheuristic counterparts. These approaches are applicable to other classes of combinatorial optimization problems as well.

1 Introduction

In the course of our current research project we aimed at investigating different promising hybrids of (meta-)heuristics and mathematical programming techniques, especially integer linear programming (ILP) methods. The testbed for these hybrids is the *periodic vehicle routing problem with time windows* (PVRPTW), where customers must be served several times in a given planning period instead of only once on a single day. In literature, this variant has not been covered to a great extent before, although its practical applicability is evident.

The PVRPTW, as considered here, is defined on a complete directed graph $G = (V, A)$ with $V = \{0, 1, \ldots, n\}$ being the set of vertices and $A = \{(i, j) \mid i, j \in V, i \neq j\}$ the set of arcs. A planning horizon of t days, referred to by $T = \{1, \ldots, t\}$, is considered. Vertex 0 represents the depot with time window $[e_0, l_0]$ at which are based m vehicles having capacity Q. Each vertex $i \in V_C$, with $V_C = V \setminus \{0\}$, corresponds to a customer and has associated a 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 non-empty set $C_i \subseteq \{T' \mid T' \subseteq T,\ |T'| = f_i\}$ of allowed combinations

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

of visit days. Each arc $(i, j) \in A$ has assigned a travel time (cost) $c_{ij} \geq 0$. The challenge consists of selecting one visit combination per customer and finding (at most) m vehicle routes on each of the t days on G such that

- each route starts and ends at the depot,
- each customer i belongs to f_i routes over the planning horizon,
- the total demand of each route does not exceed vehicle capacity Q,
- 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
- the total travel cost of all vehicles is minimized.

Arriving before e_i at a customer i implies a waiting time until this start of the time window (without further cost). Contrary, arriving later than l_i is not allowed, i.e. we assume hard time window constraints.

After discussing related work in Section 2, we first present a column generation approach for the PVRPTW based on a set covering model in Section 3, giving details on the master problem as well as on the corresponding pricing subproblem. Next, Sections 4 and 5 deal with a variable neighborhood search (VNS) and an evolutionary algorithm (EA), respectively. In Section 6 we present two matheuristics combining the before mentioned methods: On the one hand the VNS acts as sole provider of columns for the set covering model, and on the other hand the solution to the linear programming relaxation of the set covering model found by classical column generation is successfully exploited in the EA. Comparative experimental results of the matheuristics and their pure counterparts are given in Section 7. Finally, Section 8 finishes with conclusions.

2 Related Work

The PVRPTW first appeared in [1], where a tabu search algorithm is proposed for solving it. Recently we suggested a variable neighborhood search (VNS), outperforming the former tabu search [2]. It will also be used in the present work. Related VNS metaheuristics exist for the multi-depot VRPTW [3] and the *periodic vehicle routing problem* (PVRP) [4].

Evolutionary algorithms for the VRPTW are subject of [5], yet we know of no application to periodic routing problems, albeit some references for similar problem variants (e.g. the multi-depot case) can be found in [6].

One of our considered matheuristics is also presented in [7], whereas a more sophisticated variant of it can be found in [8]. Here we want to face it with another matheuristic especially from the methodical point of view. A similar idea as in [7] was recently applied to a ready-mixed concrete delivery problem [9].

Apart from our works we are not aware of other exact or hybrid methods for the PVRPTW, yet similar PVRPs (partly with other objectives) are dealt with in [10] and [11], where in [11] Mourgaya and Vanderbeck also apply column generation, yet exploit it in another way via a rounding heuristic. Finally a more general survey of different PVRP variants and solution methods is given in [12], whereas for a more general overview on ILP/metaheuristic hybrids we refer to [13, 14].

3 Column Generation Approach

Among the most successful solution approaches for VRPs in general, at least of moderate size, are algorithms based on *column generation* [15]. There the initial basis is a restricted master problem gradually enriched by new columns via iteratively solving pricing subproblems. In the following we present a suitable master problem with its corresponding pricing subproblem; more details, especially on the latter, can be found in [16].

First we formulate the *integer master problem* (IMP) for the PVRPTW as a set covering model:

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

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

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

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

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

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

Here the set of all feasible routes visiting a subset of customers is denoted by Ω. Obviously, this set is exponentially large w.r.t. the instance size. For each route $\omega \in \Omega$, let γ_ω be the corresponding costs. We introduce binary variables $\chi_{\omega\tau}$ indicating whether or not route ω is used on day τ, $\forall \omega \in \Omega$, $\tau \in T$. Furthermore, for each customer $i \in V_C$, binary variables y_{ir} indicate whether or not visit combination $r \in C_i$ is chosen. The objective function (1) corresponds to the total costs of all selected routes. 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 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.

Due to the huge amount of variables it is not possible to directly solve this ILP formulation for instances of practical size. Therefore we assume having an initial *restricted master problem* (RMP), i.e., we start with a small set $\Omega' \subset \Omega$, and subsequently determine the linear programming (LP) relaxation of it. We then proceed by solving the so-called pricing subproblem, delivering new, potentially improving columns to enter the master problem. For the PVRPTW, as for most vehicle routing problems, the subproblem can be formulated as an *elementary shortest path problem with resource constraints* (ESPPRC) [17], whereat we are searching for negative reduced cost paths (routes) from a start to an end depot (a duplicate of the former) using reduced costs based on the dual values of the LP solution. All constraints regarding single routes are thus shifted to the

subproblem. However, this comes at a price, especially when forcing elementary paths in this context (having negative cost arcs/cycles): the problem becomes NP hard, although in prospect of better lower bounds.

For our purposes, which will be detailed in Section 6, it suffices to find some negative reduced cost columns and not necessarily the most negative one(s). Following recent trends we tackle the subproblem heuristically and with an exact approach. The heuristic is similar to the tabu search method described in [18] and performs better than the one applied previously in [16]. We denote it by "reuse heuristic" since it exploits currently active columns in the master problem as initial paths. For diversification it randomly perturbs them to some extent and subsequently applies a first-improvement random local search consisting of inserting, deleting, moving, replacing, and exchanging customers. The exact method in use is a label correcting dynamic programming algorithm. However, for practical reasons we only use it in a truncated and thus also heuristic way via applying an aggressive dominance rule potentially filtering out (near) optimal least cost paths. The execution is further stopped if a specified amount of new columns were generated; see [16] for more details on the applied algorithm.

New columns generated for one of the daily subproblems are inserted for all days and the LP of the RMP is re-solved. This strategy leads to a substantial speed-up compared to only inserting the columns on the corresponding day. In the following iteration the same daily subproblem is solved again. This process continues until a full iteration over all days yields no new columns.

4 Variable Neighborhood Search

Variable neighborhood search (VNS) [19] is a metaheuristic that applies random steps in neighborhoods with growing size for diversification, referred to as shaking, and uses an embedded local search component for intensification. It has been successfully applied to a wide range of combinatorial optimization problems. In the following we give a rather short overview on our VNS for the PVRPTW as it has been already described in more detail in [2].

To smooth the search space, the VNS relaxes the vehicle load and time window restrictions and adds penalties corresponding to the excess of these constraints to the cost function, whereat both kinds of penalty terms are weighted by a constant factor of 100. The creation of the initial solution was kept quite simple by selecting a single visit day combination per customer at random and afterwards partitioning the customers at each day into routes. This partitioning is performed by sorting the customers according to the angles they make with the depot—ties are broken using the center of the time windows $(e_i + l_i)/2$—and inserting the customers in this order and a greedy fashion into at most m routes. This insertion is performed in such a way that all but the last routes of each day will comply to the load constraints, while time window constraints might be violated by all routes. The procedure is similar to the one introduced in [1].

In the shaking phase we utilize three different neighborhood structures, each with six moves of increasing perturbation size, yielding a total of 18 shaking

neighborhoods of fixed order: (i) randomly changing up to six visit combinations with greedy insertion for the new visit days, whereat we also allow reassigning the same visit combination, (ii) moving a random segment of up to six customers of a route to another one on the same day, and (iii) exchanging two random segments of up to six customers between two routes on the same day. In the latter two cases the segments are occasionally reversed.

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 [20]. 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 the Metropolis criterion like in simulated annealing [21]. A linear cooling scheme is used and the acceptance rate of worse solutions is almost zero in the last iterations.

5 Evolutionary Algorithm

Contrary to the single search trajectory followed by the VNS of the previous section an *evolutionary algorithm* (EA) [22] is a population-based metaheuristic inspired by the evolutionary process observed in nature. Although the following EA was designed already having the specific hybridization in mind, it achieves relatively good solutions on its own and is the first of its kind for periodic routing problems. Our intention was that the EA should mainly operate with whole (feasible) routes, however it turned out very quickly that an appropriate combined repair/local search component is vital.

The initial population of the stand-alone EA is created by repeatedly applying the initialization procedure described in Section 4. Also the same penalized objective function is used, however, this time with a penalty weight of 1000 to almost enforce the selection of feasible solutions only.

For recombination we have three different operators affecting different aspects, whereat in each case a standard uniform crossover is used. The first applies recombination solely on the visit combinations; the other two exclusively consider routes: either whole days including all routes are exchanged or, more fine-grained, single routes of a specific day are exchanged.

Mutation causes following changes: removal of a route, swapping of two routes of differing days, greedy addition of a random customer on a random day, removal of one occurrence of a random customer, or the change of one visit combination.

Each newly derived chromosome is subject to a procedure which eventually adjusts the visit combinations according to the routes, i.e. the visit combinations are independently chosen s.t. the least under-covering occurs, breaking ties w.r.t. the least over-covering. Though this is not applied if recombining the visit combinations, also visit combinations changed during mutation are left out. Afterwards, over-covering is tackled via removing redundant customers in

a random sequence, followed by a 2-opt intra-route improvement on the altered routes. Finally, missing customers are added in a greedy way and subsequently 2-opt improvement is applied again. Adding missing customers is the most critical part potentially rendering a solution infeasible, most likely because of a time window violation. Due to this we initially adjust the visit combinations to reduce the amount of necessary insertions. Similar to the VNS each new incumbent solution is subject to the mentioned 2-opt* procedure.

The EA applies a steady-state reproduction with a population of 100 individuals, using binary tournament selection with replacement, and accepting no duplicates (based on the objective function). All different recombination as well as mutation operators are applied with equal probability.

6 Matheuristic Variants

Having the components introduced before, we come to the actual core of this contribution: the hybridizations of the metaheuristics and parts of the ILP-based column generation approach, which can be of interest in a more general sense for other problems, too.

Our first matheuristic concerns a rather high-level combination of the VNS and the set covering ILP formulation (denoted as VNS-ILP). Hereby the VNS acts as the sole provider of columns for the set covering model, thus we have a novel pure metaheuristic column generation. The VNS and the ILP solver are applied in an intertwined way, always executing some VNS iterations to obtain new columns followed by solving the actual ILP model with the general purpose solver CPLEX. The crucial points here are which routes should enter the model, how many of them, and in which way. To increase the ILP solver's chance of finding an improved solution paired with a limited (accumulated) runtime we chose (and recommend) to consider only a few (between 5 and 10) either improved or at least above average intermediate VNS solutions and insert their (feasible) routes on the corresponding day only; for a comparison to the variant of inserting them on all days we refer to the results reported in [7]. If solving the ILP model yields an improved solution, its redundant customers are removed and the whole solution is transferred to the VNS, undergoing 2-opt* local improvement. The best incumbent solution also acts as a starting solution for the ILP solver to gain some speed.

As said, this is a rather high-level hybridization since the components basically exchange whole solutions, it could further be classified as an intertwined collaborative combination [13].

Our second matheuristic is a combination of the column generation approach and the EA (denoted as CG-EA). We were motivated by the fact that columns created when solving the LP relaxation of the problem often lend themselves to good primal solutions when the resulting model is subsequently solved with an ILP solver. This led us to think about ways of exploiting these columns and more generally the LP information derived when performing column generation. First we apply an artificial start of the RMP by inserting slack variables having

a high penalty (set to 500) and allowing to visit no customers yet to meet the visit constraints. Next we apply the re-use heuristic until it either does not find new columns or the amount of new columns decreased in the last five iterations. Afterwards we switch to the dynamic programming algorithm applied in a heuristic way. Each time the RMP is solved we keep track of the LP values of the columns (routes). Since in the end we want to have a predictable running time, we limit the runtime for solving the LP relaxation. Finally we exploit the obtained data for initializing the EA. The number of nonempty routes per day is set to the rounded down sum of the LP values of all active routes of that day. These are then set to routes corresponding to active columns of that day in the last solved LP relaxation of the RMP, whereat we apply a binary tournament selection for each route according to the accumulated LP values and prefer those having a higher value, i.e. those which have proven suitable. Currently, this procedure is applied to half of the initial chromosomes, the remaining ones are initialized as described before. Although a solution created in such a way is most likely not feasible due to over- and/or under-covering, it presumably includes high quality routes advantageous for the whole gene pool.

Since we can expect that the initially created solutions will change quite soon, it seems desirable to have an ongoing exploitation of the column generation data. Preliminary experiments turned out that a simple yet effective way of achieving this is via mutation: CG-EA can additionally replace a route by another one selected from a given pool of routes. The latter is created per day and contains all corresponding routes that were at least once active in a solution to an LP relaxation of the RMP. Again, we apply a binary tournament selection using the accumulated LP values as a decision criterion. Though more sophisticated operations would certainly be possible (e.g. a more advanced initialization, exploiting the column pool in a local search, or applying re-initializations) we rather aim here at a proof of concept that the data from column generation (i.e., generated variables and their (accumulated) LP values) can be successfully used to boost a metaheuristic.

We deem this hybridization as lower-level since specific information of the column generation is exploited in the EA, and classify it as a sequential collaborative combination [13].

To some extent the counterpart of both presented matheuristics is to apply column generation and subsequently solve the final RMP to integrality by a general purpose ILP solver both possibly with a certain time limit. In general, this is often referred to as a *column generation based heuristic*. For comparison purposes, we examine this approach also here and denote it as CG-ILP. As for CG-EA we only consider at least once active routes. Finally having a solution to the ILP we remove redundant customers and apply our 2-opt procedure on all routes. This repair process is repeated several times ($100\times$) for the same initial solution with a randomized customer removal, keeping the best solution found.

7 Experimental Results

The algorithms have been implemented in C++, compiled with GCC 4.3 and executed on a 2.83 GHz Intel Core2 Quad Q9550 with 8 GB RAM. We derived new PVRPTW instances from the Solomon VRPTW benchmark instances[1] by evenly assigning the available 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, six, and eight days, denoted by p4, p6, and p8, respectively. For p4 the customers need to be visited either 1, 2, or 4 times, for p6 either 1, 2, 3, or 6 times, and for p8 either 1, 2, 3, 4, or 8 times. 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. All instances contain 100 customers and the capacity constraint was left untouched.

For the standard VNS we set an iteration limit of 10^6, an initial temperature of 10, and apply linear cooling every 100 iterations. The intertwined VNS-ILP consists of 10 major iterations of the VNS (with 10^5 iterations each) and we insert the feasible routes of 8 improved and/or intermediate solutions (lying within the range of 5% of the actual incumbent) in the ILP model on the corresponding day only. The accumulated runtime of the ILP solver is bounded by that of the VNS. The EA always applies recombination and performs mutation following a Poisson distribution with $\lambda = 1$, running for $2 \cdot 10^5$ iterations. To have equal conditions CG-EA and CG-ILP are based on the same runs of column generation, limiting the latter to 20 seconds, which suffices for many of the instances considered.

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]). For the EA, CG-EA, and CG-ILP we also state the number of runs yielding a feasible solution.

Tables 1, 2, and 3 give results for instances having a planning horizon of four, six, and eight days, respectively. In the bottom line we state the number of times standard VNS is significantly better than VNS-ILP and vice versa, and the same for EA and CG-EA. Significantly better results are further underlined, whereat we used a Wilcoxon rank sum test with an error level of 5% for testing statistical significance. As can be observed the matheuristics very often yield significantly better results than their pure metaheuristic counterparts, in fact on up to 80% of the instances. The relative improvement of CG-EA is even more consistent than the one of VNS-ILP, which is interesting from a methodical point of view. However, comparing the absolute results would clearly be in favor of VNS-ILP, since VNS is also clearly superior to the EA. What should not be neglected is the longer runtime of the matheuristics due to the "overhead" of the corresponding exact method. For VNS-ILP the increase in runtime is often negligible, and its benefit even when compared to the VNS consuming at least the same runtime is documented in [7] as well as for a somewhat similar setting in [8]. For CG-EA we also compared the results to additional pure EA runs with $3 \cdot 10^5$ iterations,

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

i.e. an increase of 50%, which in contrast results often in (much) more allotted runtime especially for smaller instances. Nevertheless, CG-EA was still significantly better in 13, 8, and 12 cases (instead of 12, 10, and 12 cases with the shorter EA runs; see the bottom line of the tables) and the prolonged pure EA did never outperform CG-EA.

Finally we compare the matheuristics to CG-ILP: The latter only is significantly better as VNS-ILP for instance p4c102 and as CG-EA for instances p4r101, p4c102, and p4c104. Essentially, it is only meaningful when the resulting ILP model is not too large, otherwise its performance deteriorates quickly.

8 Conclusions

We investigated two new variants of matheuristics using the periodic vehicle routing problem with time windows as a testbed. Two different metaheuristics were combined with parts of a developed column generation approach: On the one hand a VNS acts as the sole provider of columns for a set covering model, realizing a pure metaheuristic column generation. On the other hand the solution to the LP relaxation of the set covering model, determined by classical column generation, is successfully exploited in a subsequent EA. Both matheuristics often yield significantly better results on newly derived instances when compared to their pure metaheuristic counterparts. They also clearly outperform a column generation based heuristic. It is to be noted that these matheuristics not only seem promising for other variants of routing problems, but their concept can fairly easily be applied to other classes of combinatorial optimization problems as well.

References

1. Cordeau, J.F., Laporte, G., Mercier, A.: A unified tabu search heuristic for vehicle routing problems with time windows. Journal of the Operational Research Society **52** (2001) 928–936
2. Pirkwieser, S., Raidl, G.R.: A variable neighborhood search for the periodic vehicle routing problem with time windows. In Prodhon, C., et al., eds.: Proceedings of the 9th EU/MEeting on Metaheuristics for Logistics and Vehicle Routing, Troyes, France (2008)
3. Polacek, M., Hartl, R.F., Doerner, K., Reimann, M.: A variable neighborhood search for the multi depot vehicle routing problem with time windows. Journal of Heuristics **10** (2004) 613–627
4. Hemmelmayr, V.C., Doerner, K.F., Hartl, R.F.: A variable neighborhood search heuristic for periodic routing problems. European Journal of Operational Research **195**(3) (2009) 791–802
5. Bräysy, O., Dullaert, W., Gendreau, M.: Evolutionary algorithms for the vehicle routing problem with time windows. Journal of Heuristics **10**(6) (2004) 587–611
6. Gendreau, M., Potvin, J.Y., Bräysy, O., Hasle, G., Løkketangen, A.: Metaheuristics for the vehicle routing problem and its extensions: A categorized bibliography. [23] 143–169

7. Pirkwieser, S., Raidl, G.R.: Boosting a variable neighborhood search for the periodic vehicle routing problem with time windows by ILP techniques. In Caserta, M., Voß, S., eds.: Proceedings of the 8th Metaheuristic International Conference (MIC 2009), Hamburg, Germany (2009)
8. Pirkwieser, S., Raidl, G.R.: Multiple variable neighborhood search enriched with ILP techniques for the periodic vehicle routing problem with time windows. In Blesa, M.J., et al., eds.: Proceedings of Hybrid Metaheuristics – Sixth International Workshop, HM 2009, Udine, Italy, October 16–17, 2009. Volume 5818 of LNCS., Springer (2009) 45–59
9. Schmid, V., Doerner, K.F., Hartl, R.F., Savelsbergh, M.W.P., Stoecher, W.: A hybrid solution approach for ready-mixed concrete delivery. Transportation Science **43**(1) (2009) 70–85
10. Francis, P., Smilowitz, K., Tzur, M.: The period vehicle routing problem with service choice. Transportation Science **40**(4) (2006) 439–454
11. Mourgaya, M., Vanderbeck, F.: Column generation based heuristic for tactical planning in multi-period vehicle routing. European Journal of Operational Research **183**(3) (2007) 1028–1041
12. Francis, P.M., Smilowitz, K.R., Tzur, M.: The period vehicle routing problem and its extensions. [23] 73–102
13. Raidl, G.R., Puchinger, J.: Combining (integer) linear programming techniques and metaheuristics for combinatorial optimization. In Blum, C., et al., eds.: Hybrid Metaheuristics: An Emerging Approach to Optimization. Volume 114 of Studies in Computational Intelligence. Springer (2008) 31–62
14. Puchinger, J., Raidl, G.R., Pirkwieser, S.: MetaBoosting: Enhancing integer programming techniques by metaheuristics. In Maniezzo, V., et al., eds.: Matheuristics: Hybridizing Metaheuristics and Mathematical Programming. Volume 10 of Annals of Information Systems. Springer (2010) 71–102
15. Desrosiers, J., Lübbecke, M.E.: A primer in column generation. [24] chapter 1 1–32
16. Pirkwieser, S., Raidl, G.R.: A column generation approach for the periodic vehicle routing problem with time windows. In Scutellà, M.G., et al., eds.: Proceedings of the International Network Optimization Conference 2009, Pisa, Italy (2009)
17. Irnich, S., Desaulniers, G.: Shortest path problems with resource constraints. [24] chapter 2 33–65
18. Desaulniers, G., Lessard, F., Hadjar, A.: Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. Transportation Science **42**(3) (2008) 387–404
19. Hansen, P., Mladenović, N.: Variable neighborhood search. In Glover, F., Kochenberger, G., eds.: Handbook of Metaheuristics. Kluwer Academic Publishers, Boston MA (2003) 145–184
20. Potvin, J.Y., Rousseau, J.M.: An exchange heuristic for routeing problems with time windows. Journal of the Operational Research Society **46** (1995) 1433–1446
21. Kirkpatrick, S., Gelatt Jr., C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220**(4598) (1983) 671–680
22. Alba, E., Cotta, C.: Evolutionary algorithms. In Olariu, S., Zomaya, A.Y., eds.: Handbook of Bioinspired Algorithms and Applications. Chapman & Hall/CRC (2006) 3–19
23. Golden, B., et al., eds.: The Vehicle Routing Problem: Latest Advances and New Challenges. Springer US (2008)
24. Desaulniers, G., et al., eds.: Column Generation. Springer (2005)

Table 1. Results of standard and hybridized methods on derived periodic Solomon instances with a planning horizon of four days.

Instance		VNS			VNS-ILP			EA				CG-EA				CG-ILP			
Id	m	avg.	sdv.	t[s]	avg.	sdv.	t[s]	avg.	sdv.	t[s]	feas	avg.	sdv.	t[s]	feas	avg.	sdv.	t[s]	feas
p4r101	14	4141.04	22.90	22.6	4119.99	25.53	22.8	4199.14	45.00	28.7	30	4162.54	35.92	31.4	30	4119.54	15.56	28.2	30
p4r102	13	3759.61	19.76	23.5	3747.57	13.28	24.5	3784.31	33.14	27.1	30	3780.50	24.52	31.7	30	3777.63	29.37	30.0	30
p4r103	10	3191.59	13.80	24.2	3188.41	11.62	26.1	3248.05	31.50	26.9	30	3217.31	24.84	33.9	30	3258.92	44.13	33.4	30
p4r104	7	2613.83	16.03	27.2	2599.56	10.07	30.1	2691.66	36.48	28.7	30	2673.09	29.85	40.7	30	2780.10	64.70	40.2	21
p4r105	11	3697.78	14.08	24.0	3679.88	15.46	37.2	3777.90	34.10	26.7	30	3745.00	28.82	29.7	30	3801.99	33.80	29.3	30
p4c101	10	2910.72	0.53	22.1	2910.13	0.38	23.3	2918.47	12.02	30.8	30	2921.08	22.11	35.0	30	2917.91	6.32	11.9	30
p4c102	8	2963.50	34.59	24.7	2960.80	32.16	26.3	3032.23	49.41	30.1	30	2963.28	42.32	44.7	30	2925.01	49.33	41.2	30
p4c103	7	2806.39	40.81	27.5	2801.59	37.26	30.5	2874.99	54.80	31.0	30	2825.01	42.33	43.9	30	2973.94	89.65	43.6	18
p4c104	7	2481.58	18.45	26.4	2474.19	20.85	31.2	2542.46	24.39	29.2	30	2518.90	32.90	46.7	30	2479.80	24.76	46.3	30
p4c105	8	3025.67	84.37	24.3	2997.44	57.16	24.8	3072.79	86.04	29.6	30	2977.45	54.82	38.1	30	2991.24	77.13	37.6	30
p4rc101	10	4003.77	12.22	25.6	3980.51	12.04	26.2	4081.77	44.36	26.9	30	4047.87	32.44	31.4	30	4087.80	43.80	31.0	30
p4rc102	10	3814.02	19.93	25.3	3795.87	15.96	26.0	3904.33	56.09	28.3	30	3869.21	53.28	32.1	30	3870.02	35.30	31.6	30
p4rc103	8	3500.84	29.90	27.5	3485.33	26.11	29.0	3596.08	45.32	28.2	29	3549.13	33.54	34.9	29	3670.73	60.17	34.5	18
p4rc104	7	3069.41	16.90	27.9	3047.90	23.54	31.9	3142.79	37.99	29.4	30	3114.51	36.46	39.4	30	3185.14	42.46	38.9	21
p4rc105	11	4008.80	25.59	24.5	3988.68	24.21	26.1	4052.78	42.09	28.7	30	4040.32	22.11	30.8	30	4047.39	40.29	30.3	30
# sign. better		0			11			0				12							

Table 2. Results of standard and hybridized methods on derived periodic Solomon instances with a planning horizon of six days.

Instance		VNS			VNS-ILP			EA				CG-EA				CG-ILP			
Id	m	avg.	sdv.	t[s]	avg.	sdv.	t[s]	avg.	sdv.	t[s]	feas	avg.	sdv.	t[s]	feas	avg.	sdv.	t[s]	feas
p6r101	14	5418.76	10.48	25.9	5404.81	19.55	27.2	5471.23	33.24	39.7	30	5453.07	32.60	41.4	30	5505.08	42.90	41.0	30
p6r102	12	5276.07	23.74	27.0	5239.89	15.29	40.3	5315.03	31.43	36.8	30	5318.87	25.76	43.0	30	5445.35	40.39	42.6	30
p6r103	9	4035.13	29.34	28.8	4004.74	22.00	36.8	4149.57	41.18	36.9	30	4120.37	34.46	48.5	30	4254.40	67.58	48.1	30
p6r104	8	3389.61	16.30	29.5	3371.90	16.10	40.9	3465.46	28.20	37.1	30	3441.55	22.04	53.1	30	3665.01	62.43	52.6	30
p6r105	9	4355.02	27.90	28.3	4348.88	38.62	32.3	4514.95	46.59	35.8	30	4457.93	48.46	44.0	30	4647.59	112.43	43.6	28
p6c101	7	4084.67	37.49	30.1	4084.71	40.45	29.9	4192.24	77.09	36.8	30	4162.92	68.33	50.0	30	4592.38	194.23	49.6	4
p6c102	7	3888.96	21.34	29.7	3884.62	22.65	32.8	3960.89	56.36	38.9	30	3950.54	65.92	55.2	30	4414.48	208.85	54.7	19
p6c103	6	3616.61	45.55	33.8	3609.83	38.10	37.8	3788.68	63.95	37.3	30	3719.95	82.20	55.3	30	4191.75	170.11	54.8	13
p6c104	6	3295.32	18.40	32.8	3285.39	18.78	40.2	3450.31	54.19	36.5	30	3422.22	56.05	54.5	30	3766.94	92.55	54.0	18
p6c105	7	4164.39	66.01	29.9	4173.86	83.14	30.8	4285.79	84.27	37.1	30	4181.50	56.15	53.3	30	4551.39	187.19	52.9	13
p6rc101	10	5846.32	25.11	27.4	5826.98	19.20	30.6	5932.49	46.38	34.6	30	5909.63	40.87	39.4	30	6128.02	67.00	39.0	30
p6rc102	9	5483.15	27.08	28.2	5473.92	33.20	31.4	5577.50	54.72	36.0	30	5553.47	52.54	42.3	30	5756.83	83.19	41.8	30
p6rc103	7	4370.85	26.07	32.2	4351.65	21.16	34.3	4521.57	50.34	35.4	30	4476.44	45.03	50.5	30	4699.27	67.30	50.0	23
p6rc104	7	4147.69	26.45	31.4	4133.38	25.98	40.8	4306.30	52.83	36.1	30	4267.67	41.26	50.5	30	4436.69	85.22	49.9	30
p6rc105	9	5340.30	22.46	28.9	5323.00	21.92	30.6	5467.39	58.06	35.6	30	5450.10	44.81	42.3	30	5582.21	70.66	41.8	30
# sign. better			0			9			0				10						

Table 3. Results of standard and hybridized methods on derived periodic Solomon instances with a planning horizon of eight days.

Instance		VNS			VNS-ILP			EA				CG-EA				CG-ILP			
Id	m	avg.	sdv.	t[s]	avg.	sdv.	t[s]	avg.	sdv.	t[s]	feas	avg.	sdv.	t[s]	feas	avg.	sdv.	t[s]	feas
p8r101	11	6574.92	36.55	26.2	6557.78	37.26	29.8	6711.50	46.38	43.8	21	6696.89	75.06	53.5	27	6820.33	68.96	53.0	30
p8r102	10	6193.62	99.24	28.3	6205.41	45.66	30.6	6300.33	49.19	44.7	20	6313.65	70.20	60.8	22	6508.59	125.34	60.4	29
p8r103	8	4809.73	26.53	31.9	4806.69	34.24	36.4	4999.87	67.08	44.2	30	4930.83	43.14	61.4	30	5250.39	114.26	61.0	29
p8r104	7	4495.36	28.29	32.8	4477.20	28.89	43.0	4667.39	52.74	44.3	30	4598.77	64.23	62.4	30	5181.33	117.11	61.9	26
p8r105	9	5600.71	39.34	28.5	5585.25	37.59	35.6	5817.43	69.13	43.1	30	5744.09	53.36	58.2	30	6013.38	105.65	57.7	27
p8c101	7	4781.05	41.91	30.2	4786.88	39.11	32.6	4991.15	119.77	44.3	30	4900.84	75.41	62.0	30	5185.67	131.66	61.5	19
p8c102	6	5169.88	71.00	37.8	5188.80	76.68	39.4	5410.25	121.16	44.7	30	5308.69	114.27	64.9	30	6442.40	0.00	64.5	1
p8c103	5	4794.70	50.18	35.9	4788.86	36.49	40.5	5029.64	105.63	44.3	30	4965.95	69.92	62.4	30	6428.76	272.02	61.9	13
p8c104	8	4845.02	71.06	26.4	4853.57	65.88	37.3	5234.18	79.30	41.5	30	5202.05	91.42	60.4	30	5592.48	254.28	59.9	5
p8c105	7	5261.79	58.32	30.9	5237.81	42.58	33.4	5434.17	91.13	45.6	30	5384.95	95.36	61.7	30	6293.36	254.71	61.2	14
p8rc101	9	7075.80	75.20	28.5	7035.37	64.98	31.3	7225.04	98.40	43.2	30	7134.84	80.09	55.1	29	7432.93	152.95	54.6	20
p8rc102	8	5951.36	47.72	29.9	5954.42	67.27	32.2	6249.95	102.21	41.8	30	6163.02	76.37	60.5	28	6392.33	149.02	60.1	23
p8rc103	7	5560.42	34.89	33.1	5552.43	33.69	34.5	5847.79	95.54	44.7	30	5778.00	73.80	61.7	30	6126.24	128.82	61.3	8
p8rc104	6	5080.84	35.82	32.1	5071.39	38.96	38.0	5301.08	52.53	43.2	30	5277.23	46.59	60.3	30	5873.20	144.27	59.9	25
p8rc105	9	6383.60	43.49	28.6	6358.24	30.28	30.8	6606.78	79.69	42.9	30	6530.36	62.94	58.5	30	6727.78	120.43	58.0	30
# sign. better		0			4			0				12							