A Column Generation Approach for the Periodic Vehicle Routing Problem with Time Windows[1]

Sandro Pirkwieser* Günther R. Raidl*

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

Abstract

We present a column generation approach for obtaining strong lower bounds to the periodic vehicle routing problem with time windows (PVRPTW) where customers must be served several times during a planning period. For this a set-covering model is introduced whose linear programming relaxation is solved. The pricing subproblem, responsible for generating new columns, is an elementary shortest path problem with resource constraints. The latter is solved by a label correcting dynamic programming algorithm for which we introduce appropriate label resources, extension functions, and dominance rules. Different settings for this algorithm are suggested and applied in combination to tune its behavior. We further propose a greedy randomized adaptive search procedure (GRASP) to solve the pricing subproblem. Experimental results on test instances differing in size, time windows and period length indicate strong lower bounds for many instances and the advantage of applying the metaheuristic in combination with the dynamic programming algorithm to save computation time on larger instances.

Keywords: *Periodic Vehicle Routing Problem with Time Windows, Column Generation, Dynamic Programming, Heuristic Pricing, Hybridization*

1 Introduction

Periodic vehicle routing problems (PVRPs) are generalized variants of the classical vehicle routing problem (VRP) where customers must be served several times for a given planning period. They occur in real world applications as in courier services, grocery distribution or waste collection. We contribute to the *Periodic Vehicle Routing Problem with Time Windows* (PVRPTW), a PVRP variant deserving more attention.

The PVRPTW is defined on a complete directed 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. The considered planning horizon shall be t days, also referred to as $T = \{1, \dots, t\}$. Vertex v_0 represents the depot with time window $[e_0, l_0]$ at which are based m vehicles having capacities $Q_1, \dots, Q_m$ and maximal working times $D_1, \dots, D_m$. Each vertex $i \in V_C$, with $V_C = V \setminus \{v_0\}$, 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. For each arc $(v_i, v_j) \in A$ there are further given travel times (costs) $c_{ij} \geq 0$. 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]This work is supported by the Austrian Science Fund (FWF) under contract number P20342-N13.

(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 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.

Related work is mentioned in Section 2, an integer linear programming (ILP) formulation and the solution of its linear programming (LP) relaxation in order to get lower bounds is presented in Section 3, followed by experimental results in Section 4. Concluding remarks are given in Section 5.

2 Related Work

The PVRPTW was first mentioned in [3], where a tabu search algorithm is described for it. In our previous work [10] we applied a variable neighborhood search, outperforming the former tabu search. To our knowledge this specific problem has not been tackled by exact methods so far. This stays in contrast to the related VRPTW, for which a lot of exact approaches have been published; an overview is given in [8]. A more general survey of different PVRP variants and solution methods is reported in [6].

3 A Set-Covering Formulation for the PVRPTW

Among the most successful solution approaches for VRPs are algorithms based on column generation [4], where the initial basis is a restricted master problem gradually enriched by new columns by iteratively solving pricing subproblems. Therefore we focus on an ILP formulation suitable for such an approach.

We formulate the *integer master problem* (IMP) for the PVRPTW as a set-covering model, since such an approach also led to strong bounds in case of the VRPTW [1]:

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

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

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

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

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

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

The set of all feasible routes, which is exponentially large, is denoted by Ω, and for each route $\omega \in \Omega$ its cost is γ_ω and $\chi_{\omega\tau}$ is the number of times route ω is selected on day τ. 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 and indicate whether route ω visits customer i and if day τ belongs to visit combination $r \in C_i$ of customer i, respectively. In the following we want to derive a good lower bound for the IMP by solving its LP relaxation. Therefore conditions (5) and (6) are replaced by $y_{ir} \geq 0$ and $\chi_{\omega\tau} \geq 0$, yielding the (linear)

master problem (MP). Due to the large number of variables (columns) corresponding to routes, this LP cannot be solved directly. Instead, we restrict ourself to a small number of initial columns $\Omega' \subset \Omega$. The corresponding LP is referred to as *restricted master problem* (RMP). Additional columns (routes) that are able to improve the current LP solution are generated by iteratively solving the so-called pricing subproblem.

3.1 Pricing Subproblem

The pricing subproblem resembles a *shortest path problem with resource constraints* (SPPRC) [7]. Regarding the quality of the theoretically obtainable lower bound it is beneficial to restrict the search to elementary paths, hence only considering the *elementary SPPRC* (ESPPRC). Unfortunately, this condition renders the problem NP hard. Following ESPPRC pricing subproblem holds for each day $\tau \in T$ and is solved on the auxiliary graph $G' = (V', A')$, with $V' = V \cup \{v_{n+1}\}$ and $A' = \{(v_0, i), (i, v_{n+1}) : i \in V_C\} \cup \{(i, j) : i, j \in V_C,\, i \neq j\}$, where v_{n+1} is a copy of the (starting) depot v_0 and acts as target node, we further assume a homogeneous fleet of vehicles:

$$\min \sum_{i \in V'} \sum_{j \in V'} \hat{c}_{ij\tau}\, x_{ij} \tag{7}$$

$$\text{s.t.} \quad \sum_{j \in V_C} x_{0j} = 1 \tag{8}$$

$$\sum_{i \in V'} x_{ik} - \sum_{j \in V'} x_{kj} = 0 \qquad \forall k \in V_C \tag{9}$$

$$\sum_{i \in V_C} x_{i,n+1} = 1 \tag{10}$$

$$\sum_{i \in V_C} \sum_{j \in V'} q_i\, x_{ij} \leq Q \tag{11}$$

$$a_{n+1} - w_0 \leq D \tag{12}$$

$$a_i + w_i + d_i + c_{ij} - M_{ij}(1 - x_{ij}) \leq a_j \qquad \forall (i, j) \in A' \tag{13}$$

$$e_i \leq (a_i + w_i) \leq l_i \qquad \forall i \in V' \tag{14}$$

$$w_i \geq 0 \qquad \forall i \in V' \tag{15}$$

$$a_i \geq 0 \qquad \forall i \in V' \setminus \{v_0\} \tag{16}$$

$$a_0 = 0 \tag{17}$$

$$x_{ij} \in \{0, 1\} \qquad \forall (i, j) \in A' \tag{18}$$

Variables x_{ij}, $\forall (i, j) \in A'$ denote which arcs from A' are used, and $\hat{c}_{ij\tau}$ is the *reduced cost* of using arc (i, j) on day τ:

$$\hat{c}_{ij\tau} = \begin{cases} c_{ij} - \rho_\tau & \text{if } i = v_0,\ j \in V_C \\ c_{ij} - \pi_{i\tau} & \text{if } i \in V_C,\ j \in V' \end{cases}$$

with ρ_τ and $\pi_{i\tau}$ being the corresponding dual variable values of constraints (3) and (4), respectively. Constraints (8)–(10) are the flow constraints, (11) and (12) guarantee feasibility regarding capacity and duration constraints, respectively. Finally, (13) and (14) are time constraints, with variable a_i denoting the arrival time at customer i and w_i being the waiting time occurring after this visit.

The ESPPRC subproblem is solved by a dynamic programming approach based on [2, 5]. We use a label correcting algorithm and expand the partial paths from the depot v_0 to the target node v_{n+1}, thereby retaining only non-dominated labels. Due to space limitations we only describe the label and the dominance criteria suitable for our problem setting in more detail, which are an important part of the whole algorithm. We are faced both with the existence of time windows and restrictions on route duration. To our knowledge this combination was not part of any work in the context of this algorithm

before, though it is highly relevant in practice. Due to these constraints it is a non-trivial task to find non-dominated paths, since on the one hand partial paths arriving earlier might be beneficial regarding following time windows, whereas on the other hand partial paths arriving later might be able to reach more customers afterwards.

To minimize route duration we adhere to the concept of *forward time slack* [12] and maximize the initial waiting time w_0 at the (start) depot without introducing a time window violation. When building a path we need to determine the minimum of this time slack, also used to calculate the minimal route duration.

A label associated with a partial path p at node v_i holds the following resource information: accumulated cost $\mathcal{C}_i$, load $\mathcal{L}_i$, and overall waiting time $\mathcal{W}_i$, as well as the arrival time $\mathcal{A}_i$, the actual minimal forward time slack $\mathcal{F}_i$, and a set $\overline{V_i}(p)$ containing already visited nodes and those unreachable due to the numerous restrictions. Having this information, we define two more resources which are calculated on-the-fly: the start of service time $\mathcal{S}_i = \max\{\mathcal{A}_i, e_i\}$ and the current minimal route duration $\mathcal{D}_i = \mathcal{S}_i - \min\{\mathcal{F}_i, \mathcal{W}_i\}$.

When moving from node v_i to node v_j the label resources are updated as follows:

- $\mathcal{C}_j = \mathcal{C}_i + c_{ij}$
- $\mathcal{L}_j = \mathcal{L}_i + q_i$
- $\mathcal{A}_j = \mathcal{A}_i + w_i + c_{ij} + d_i$
- $\mathcal{W}_j = \mathcal{W}_i + \max\{0, e_j - \mathcal{A}_j\}$
- $\mathcal{F}_j = \min\{\mathcal{F}_i, \mathcal{W}_j + (l_j - \mathcal{S}_j)\}$

When a path reaches the target node v_{n+1} the initial waiting time at the depot is set to $w_0 = \mathcal{F}_{n+1}$ in order to yield the overall minimal route duration $\mathcal{D}_{n+1}$.

Finally, based on these resources we define three different dominance rules stating whenever partial path p^1 dominates partial path p^2, both ending at the same node v_i:

R1: $\mathcal{C}_i^1 \leq \mathcal{C}_i^2 \ \wedge \ \mathcal{D}_i^1 \leq \mathcal{D}_i^2 \ \wedge \ \mathcal{L}_i^1 \leq \mathcal{L}_i^2 \ \wedge \ \mathcal{A}_i^1 \leq \mathcal{A}_i^2$

R2: $\mathcal{C}_i^1 \leq \mathcal{C}_i^2 \ \wedge \ \mathcal{D}_i^1 \leq \mathcal{D}_i^2 \ \wedge \ \mathcal{L}_i^1 \leq \mathcal{L}_i^2 \ \wedge \ \mathcal{A}_i^1 \leq \mathcal{A}_i^2 \ \wedge \ \overline{V_i}(p^1) \subseteq \overline{V_i}(p^2)$

R3: $\mathcal{C}_i^1 \leq \mathcal{C}_i^2 \ \wedge \ \mathcal{D}_i^1 \leq \mathcal{D}_i^2 \ \wedge \ \mathcal{L}_i^1 \leq \mathcal{L}_i^2 \ \wedge \ \mathcal{A}_i^1 \leq \mathcal{A}_i^2 \ \wedge \ \overline{V_i}(p^1) \subseteq \overline{V_i}(p^2) \ \wedge \ \mathcal{F}_i^1 \geq \mathcal{F}_i^2$

A dominated partial path is discarded from further consideration. Dominance rule R1 is quite simple and fast, a similar one was used in [2]. However, it sometimes also filters out (near) optimal least cost paths. Contrary to rule R1, rules R1 and R3 are more refined and also consider sets $\overline{V_i}(p^1)$ and $\overline{V_i}(p^2)$. Especially, rule R3 additionally takes the actual forward time slack into account, which is the critical resource connecting the arrival time and the duration. Due to rules R1 and R2 being too strict and therefore of heuristic nature, rule R3 must be applied at last to guarantee finding all least cost paths. In order to decrease computation time we apply the following sequential dominance rule scheme:

(a) use rule R1 as long as more than 100 new columns could be found, else switch permanently to (b)

(b) use rule R2 until no new columns could be found, then switch to (c)

(c) finally use rule R3, returning to (b) in the next run whenever new columns could be found; terminate the column generation otherwise.

The dynamic programming algorithm can also be stopped after a certain number of negative cost paths have been found, i.e. applying a "forced early stop" [9].

We further use a metaheuristic to generate new columns which can be regarded a greedy randomized adaptive search procedure (GRASP) [11]: In each iteration we start with the "empty" path (v_0, v_{n+1}) with zero cost and successively try to add arcs with negative cost, always selecting one at random in case there are more available; afterwards we apply up to ten random moves out of the set of inserting, deleting, moving, replacing and exchanging customers; finally, we perform a local search also based on these moves, whereas they are applied in a random fashion and we always accept the first improving change. Whenever an iteration results in a negative cost path it is stored and returned at the end of the heuristic.

As soon as new columns are generated for one of the daily subproblems they are inserted in 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 in 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.

Table 1: Experimental results of different subproblem algorithm settings.

Instance				UB	LB	%-gap	DP	DP^S	GRASP+DP	
No.	n	m	t				t[s]	t[s]	min t[s]	med t[s]
1a	48	3	4	2909.02	2882.01	0.94	10.10	5.60	9.44	14.65
2a	96	6	4	5032.06	4993.48	0.77	112.93	58.67	55.15	58.64
3a	144	9	4	7138.65	6841.44	4.34	787.67	399.88	319.92	404.92
$4a_{r1}$	160	10	4	6929.84	6641.67	4.34	1920.73	1422.78	823.79	900.33
7a	72	5	6	6784.71	6641.39	2.16	29.95	20.40	17.32	21.52
$9a_{r1}$	96	7	6	8545.80	8035.09	6.36	278.77	121.24	97.28	103.83
$9a_{r2}$	120	8	6	8598.40	8140.15	5.63	1822.78	994.40	730.27	829.76
8a	144	10	6	9721.25	9153.79	6.20	1493.19	720.00	418.85	505.83
$2b_{r1}$	32	2	4	2709.15	2682.52	1.00	53.43	49.93	42.91	76.43
1b	48	3	4	2277.44	2258.85	0.82	189.32	159.54	25.15	127.84
$2b_{r2}$	64	4	4	2771.68	2733.55	1.40	209.66	232.45	166.73	217.94
$3b_{r1}$	72	4	4	3306.86	3241.90	2.00	543.81	618.91	496.99	548.20
$7b_{r1}$	24	2	6	3776.25	3677.21	2.70	0.71	0.69	1.98	2.25
$8b_{r1}$	36	2	6	3640.79	3476.43	4.73	13.39	9.00	10.75	13.16
$7b_{r2}$	48	3	6	3723.18	3599.72	3.43	45.93	35.75	34.37	39.01
$8b_{r2}$	60	3	6	4606.17	4324.87	6.50	2084.47	1448.49	1163.47	1586.17

4 Experimental Results

The test instances were taken from [3]. They are divided in types 'a' and 'b' having narrow and large time windows, respectively. We reduced some of them by selecting only a random subset of the customers and appropriately adapting the number of vehicles; in this case we give a subscript denoting the index of the reduced instance. The initial set of columns is provided by taking the routes of feasible solutions of our metaheuristic reported in [10]. The algorithms have been implemented in C++, compiled with GCC 4.1 and executed on a single core of a 2.2 GHz Dual-Core AMD Opteron 2214 PC with 4 GB RAM. Ilog CPLEX 11.2 was used as LP solver. The ESPPRC subproblem is solved in three ways: dynamic programming (DP), dynamic programming with forced early stop after generating more than 1000 columns (DP^S), and a hybrid method composed of the GRASP-based metaheuristic with up to 10000 iterations—in case it did not find new columns in the first 1000 iterations—as long as it finds more than 100 new columns and applying DP afterwards (GRASP+DP). In Table 1 we state the instances, the initially provided upper bounds (UB), the derived lower bounds (LB), the percentage gaps between them, i.e. $\%\text{-gap} = (\text{UB} - \text{LB})/\text{LB} \cdot 100\%$, the CPU times of settings DP and DP^S, as well as the minimal and median times of setting GRASP+DP over 10 runs. It can be observed that applying a forced early stop (DP^S) is in general faster than using none, especially for instances with narrow time windows where the runtime is often halved. Using the GRASP+DP hybrid is beneficial for larger instances, when the faster heuristically generated columns outweigh the probably higher quality columns of the DP algorithm. Again, this difference is more obvious for instances having narrow time windows. The resulting gaps are clearly smaller for instances with a period of four days, though this is also affected by the method providing the initial solutions.

5 Conclusions

In this work we presented an ILP formulation for the periodic vehicle routing problem with time windows (PVRPTW) based on a set-covering model. The LP relaxation is solved via column generation, whereas the pricing subproblem resembles an elementary shortest path problem with resource constraints

(ESPPRC) for which we apply an exact dynamic programming algorithm. An important issue are the simultaneous restrictions on time windows as well as on route duration. For this purpose we proposed suitable label resources, their extension function, and dominance rules, incorporating the concept of forward time slack in order to minimize overall route duration. Furthermore a GRASP-based metaheuristic is proposed for the ESPPRC. Computational results indicate the advantage of applying a forced early stop as well as the metaheuristic in combination with the dynamic programming algorithm to save computation time on larger instances. For many instances only small remaining gaps were achieved.

We are already working on a a suitable column management to speed up the whole process, which is especially appealing as we are adding multiple columns per iteration. Future work will be dedicated to applying other (meta-)heuristics for solving the ESPPRC, adding suitable cuts, and designing appropriate primal heuristics used during or after the column generation process. Finally, we intend to embed this column generation approach in a branch-(and-cut)-and-price framework to narrow the resulting gap or even find proven optimal solutions.

References

[1] J. Bramel and D. Simchi-Levi. On the effectiveness of set covering formulations for the vehicle routing problem with time windows. *Operations Research*, 45:295–301, 1997.

[2] A. Chabrier. Vehicle routing problem with elementary shortest path based column generation. *Computers & Operations Research*, 33(10):2972–2990, 2006.

[3] 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.

[4] J. Desrosiers and M. E. Lübbecke. A primer in column generation. In G. Desaulniers et al., editors, *Column Generation*, chapter 1, pages 1–32. Springer, 2005.

[5] D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229, 2004.

[6] 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.

[7] S. Irnich and G. Desaulniers. Shortest path problems with resource constraints. In G. Desaulniers et al., editors, *Column Generation*, chapter 2, pages 33–65. Springer, 2005.

[8] B. Kallehauge. Formulations and exact algorithms for the vehicle routing problem with time windows. *Computers & Operations Research*, 35(7):2307–2330, 2008.

[9] J. Larsen. *Parallelization of the Vehicle Routing Problem with Time Windows*. PhD thesis, Technical University of Denmark, 1999.

[10] 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.

[11] M. Resende and C. Ribeiro. Greedy randomized adaptive search procedures. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, pages 219–249. Kluwer, 2003.

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