Discrete Applied Mathematics 163 (2014) 53–64

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

A set-covering based heuristic algorithm for the periodic vehicle routing problem

V. Cacchiani [a], V.C. Hemmelmayr [b,c,*], F. Tricoire [b,d]

[a] DEIS, University of Bologna, Viale Risorgimento 2, 40136 Bologna, Italy

[b] Department of Business Administration, University of Vienna, Bruenner Strasse 72, 1210 Vienna, Austria

[c] Institute for Transport and Logistics Management, WU, Vienna University of Economics and Business, Nordbergstraße 15, 1090 Vienna, Austria

[d] NICTA / UNSW, Sydney, Australia

ARTICLE INFO

Article history:
Received 17 February 2011
Received in revised form 11 June 2012
Accepted 28 August 2012
Available online 19 September 2012

Keywords:
Periodic vehicle routing
Matheuristics
Column generation

ABSTRACT

We present a hybrid optimization algorithm for mixed-integer linear programming, embedding both heuristic and exact components. In order to validate it we use the periodic vehicle routing problem (PVRP) as a case study. This problem consists of determining a set of minimum cost routes for each day of a given planning horizon, with the constraints that each customer must be visited a required number of times (chosen among a set of valid day combinations), must receive every time the required quantity of product, and that the number of routes per day (each respecting the capacity of the vehicle) does not exceed the total number of available vehicles. This is a generalization of the well-known vehicle routing problem (VRP). Our algorithm is based on the linear programming (LP) relaxation of a set-covering-like integer linear programming formulation of the problem, with additional constraints. The LP-relaxation is solved by column generation, where columns are generated heuristically by an iterated local search algorithm. The whole solution method takes advantage of the LP-solution and applies techniques of fixing and releasing of the columns as a local search, making use of a tabu list to avoid cycling. We show the results of the proposed algorithm on benchmark instances from the literature and compare them to the state-of-the-art algorithms, showing the effectiveness of our approach in producing good quality solutions. In addition, we report the results on realistic instances of the PVRP introduced in Pacheco et al. (2011) [24] and on benchmark instances of the periodic traveling salesman problem (PTSP), showing the efficacy of the proposed algorithm on these as well. Finally, we report the new best known solutions found for all the tested problems.

1. Introduction

Routing problems have been widely studied due to the economic importance of developing efficient techniques for optimization in transportation. Many real-world applications in transportation systems require finding a set of minimum cost routes to be performed by a fleet of vehicles, that satisfy orders of products requested by customers, over a given planning horizon (e.g. one week). In some applications, it can be useful to visit customers a different number of times, depending on the amount of products requested. In addition, certain days of the planning horizon can be specified by the

* Corresponding author at: Department of Business Administration, University of Vienna, Bruenner Strasse 72, 1210 Vienna, Austria. Fax: +43 1 313 36 716.

E-mail addresses: vera.hemmelmayr@univie.ac.at, vera.hemmelmayr@wu.ac.at (V.C. Hemmelmayr).

0166-218X
doi:10.1016/j.dam.2012.08.032

customers as available for the delivery of the product. Taking into account several days of planning for routing problems is an interesting variant of the classical vehicle routing problem (VRP), known as the periodic vehicle routing problem (PVRP). This problem is very important in real-world applications such as distribution for bakery companies [24], or blood product distribution [21].

In this work we propose a hybrid optimization heuristic algorithm that can be applied to mixed integer linear programming problems. It is based on the solution of the linear programming (LP) relaxation of an integer linear programming formulation of the problem. The LP-relaxation can be solved by a general purpose solver or by column generation techniques, with exact or heuristic methods. The LP-solution can be then used to guide the heuristic algorithm for obtaining an integer solution.

This heuristic algorithm iteratively fixes and releases variables from the LP. Variables are fixed when they are deemed contextually promising, and released after some time in order to diversify the search. In the context of this paper, heuristic column generation is used for obtaining an LP-solution: the subproblem is solved by an iterated local search (ILS) procedure.

In order to validate the proposed heuristic algorithm, we apply it to the periodic vehicle routing problem. This is a very well-studied problem for which many heuristic algorithms and an exact approach have already been proposed. It is an extension of the well-known VRP, and takes into account a given planning horizon. Each customer must be visited a required number of times along the planning horizon. In particular, a set of valid day combinations is given for each customer. In addition each customer has a given demand of product, that must be satisfied every time he is visited. A fleet of homogeneous vehicles is also given, each one with the same capacity. The PVRP calls for determining a set of minimum cost routes for each day of the planning horizon, satisfying the following constraints:

1. each customer is visited the required number of times, assigned a combination chosen among a set of appropriate day combinations for this customer, and is delivered the required quantity of product
2. the number of routes on each day does not exceed the total number of available vehicles
3. the capacity of each vehicle is not exceeded.

The PVRP generalizes the VRP since it calls for determining an appropriate day combination for each customer and simultaneously finding a set of minimum cost routes for each day.

The paper is organized as follows: in the next section, we briefly review the main works on PVRP and on periodic traveling salesman problem (PTSP), that is a special case of the PVRP where only one vehicle is available every day; in Section 3 we present the formal description of the problem and provide an integer linear programming (ILP) formulation in a set-covering form with additional constraints, together with the arising subproblem for the solution of the LP-relaxation of the model by column generation; in Section 4 we present the solution method, which consists of solving the LP-relaxation of the SC-formulation in a heuristic way, and of fixing and releasing variables, based on this solution. Experimental results are presented in Section 5 and compared with the state-of-the-art algorithms. Finally, some conclusions are drawn in Section 6.

2. Related work

The PVRP has received much attention in the literature, since it also arises in many real-world applications. A relatively recent survey on the PVRP and its extensions was written by Francis et al. [16]. Due to the difficulty of the problem, most of the works present heuristic approaches, although recently an exact method has been proposed by Baldacci et al. [2]. It is based on a set partitioning integer linear programming formulation of the problem and on three different relaxations used to derive powerful lower bounds.

The first work on the PVRP is proposed by Beltrami and Bodin [3]. Then there are other works by Christofides and Beasley [7], Tan and Beasley [31], Russel and Gribbin [28] and Gaudioso and Paletta [17]. Chao et al. [6] are the first to provide a two-phase heuristic that allows escaping from local optima. They use an integer linear program to assign visit day combinations to customers in order to initialize the system. Moreover, the capacity limit of the vehicles is temporarily relaxed.

In [8] a tabu search (TS) heuristic is proposed that can solve the PVRP, the multi-depot vehicle routing problem (MDVRP) and the periodic traveling salesman problem (PTSP). Two types of neighborhood operators are considered: the first one considers moving a customer from one route to another route on the same day. The second one considers assigning a new visit day combination to a customer. Insertions and removals of customers are performed with the generalized insertion (GENI) operator [18]. Infeasibility with respect to capacity and tour length constraints is allowed during the search process using an adaptive penalty function.

Alegre et al. [1] consider a real world application that arises in periodic pick-up of raw materials for a manufacturer of automobile parts. They propose a Scatter Search to solve this problem and the standard benchmark instances available in the literature. The algorithm is based on a two-phase approach, that first assigns orders to days and then constructs routes for each day.

Furthermore, special implementations for real-world problems are provided by Hadjiconstantinou and Baldacci [19] who propose a heuristic for a multi-depot period vehicle routing problem that arises in the utility sector. Francis et al. [15] solve the PVRP with service choice, where service frequency is a decision of the model. They propose an exact solution method and heuristic variations of it for larger instances. Hemmelmayr et al. [20] propose a variable neighborhood search (VNS) algorithm using three neighborhood structures. Those are moving a sequence of customers from one route to another route,

crossing sequences of customers between routes and changing the visit day combination of one or more customers. For a local search 3-opt is used and the solution acceptance is based on simulated annealing (SA).

Very recently, Vidal et al. [32] propose a hybrid genetic algorithm: the population consists of individuals representing feasible and infeasible solutions; the population evolves by applying different operators with the aim of having high quality solutions while maintaining diversity. They test the algorithm on benchmark instances for the PVRP and the MDVRP and on a new set of instances for the multi-depot periodic vehicle routing problem (MDPVRP), providing very good solutions.

Another very recent approach is proposed in [24]: the authors study the real problem of a bakery company in northern Spain. In order to minimize the total distance traveled for the daily routes over the week, the bakery company allows some flexibility in the dates of delivery. The authors propose a mixed-integer linear model and solve the problem through a two-phase algorithm. In the first phase, a set of good and diverse solutions is generated, based on GRASP. In the second phase, path-relinking is applied in order to improve the solutions. Computational experiments are performed on real-data-based instances. In addition, the authors apply the necessary modifications to treat the problem as a PVRP, in order to compare their algorithm with state-of-the-art algorithms for the PVRP.

The periodic traveling salesman problem (PTSP) is a special case of the PVRP where only one vehicle is available every day. A mathematical formulation of the PTSP can be found in [8]. Heuristic algorithms were proposed in [25,5]. The tabu search by Cordeau et al. [8] and the VNS approach by Hemmelmayr et al. [20], previously described, are very effective for the PTSP as well. In [4], a heuristic algorithm is proposed for the PTSP, that contains an improved version of the algorithm by Paletta [25].

3. Problem description

We consider the following input:

- T, the total number of days in the planning horizon,
- m, the number of vehicles available every day, each one with capacity K,
- n, the total number of customers to visit,
- C_i, the set of valid day combinations for customer $i \in \{1, \ldots, n\}$
- $G = (V', E)$, an undirected graph with a set of vertices $V' = \{0\} \cup V$, where 0 is the depot and V corresponds to the set of customers, with cost d_{ij}^t of traversing edge $(i, j) \in E$ on day $t \in \{1, \ldots, T\}$.

Let Ω be the set of all feasible routes, and d_r^t the cost of route $r \in \Omega$ when performed on day t. Let also a_{ir} be equal to 1 if customer i is visited by route r, 0 otherwise ($i \in \{1, \ldots, n\}$, $r \in \Omega$). Finally, let b_p^t be equal to 1 if day t is in combination p, 0 otherwise ($t \in \{1, \ldots, T\}$, $p \in C_i$, $i \in \{1, \ldots, n\}$).

We present a set-covering-like ILP formulation for the problem, with additional constraints. We introduce two sets of binary variables, x_r^t equal to 1 if and only if route r is performed on day t, and y_{ip} equal to 1 if and only if combination p is selected for customer i. A SC-formulation for the problem reads as follows:

$$\min \sum_{r \in \Omega} \sum_{t=1}^{T} x_r^t d_r^t \tag{1}$$

s.t.

$$\sum_{p \in C_i} y_{ip} \geqslant 1 \quad i \in \{1, \ldots, n\} \tag{2}$$

$$\sum_{r \in \Omega} x_r^t a_{ir} - \sum_{p \in C_i} y_{ip} b_p^t \geqslant 0 \quad i \in \{1, \ldots, n\},\ t \in \{1, \ldots, T\} \tag{3}$$

$$\sum_{r \in \Omega} x_r^t \leqslant m \quad t \in \{1, \ldots, T\} \tag{4}$$

$$x_r^t \in \{0, 1\}, \quad r \in \Omega,\ t \in \{1, \ldots, T\} \tag{5}$$

$$y_{ip} \in \{0, 1\}, \quad p \in C_i,\ i \in \{1, \ldots, n\}. \tag{6}$$

The objective function aims at minimizing the overall cost of the selected routes. Constraints (2) guarantee that at least one combination is selected for each customer. Constraints (3) ensure that every customer is visited by at least one route on each day belonging to its selected combination. Constraints (4) guarantee that the fleet size limit is respected for every day in the planning horizon. The proposed formulation is similar to the one presented in [2], that however is a set-partitioning-like model, i.e. constraints (2) and (3) are equality constraints. The same sets of variables are introduced and constraints (4) are expressed in the same way. Constraints (2) are expressed by considering the frequency of visit of the customers instead of the set of day combinations.

One of the main ingredients of the proposed heuristic approach lies on the solution of the LP-relaxation of the SC-formulation (1)–(6). We adopt column generation techniques for solving the LP-relaxation. In the next section, we will

first describe the arising subproblem and a metaheuristic algorithm that we developed for solving it. The proposed *SC-heur* algorithm makes use of the heuristic LP-solution and applies iteratively fixing and releasing of route variables. A more detailed description of *SC-heur* will be given in the next section.

3.1. Subproblem generation

We focus on the LP-relaxation of model (1)–(6) and we use a column generation approach to solve it. The arising subproblem consists of the so-called elementary shortest path problem with resource constraints (ESPPRC). The objective is to minimize the arc costs, that besides the original costs take into account a *prize* given by the dual variables associated to constraints (3) for each visited customer. The constraints of ESPPRC impose to have a path without cycles, going from the depot to a copy of it added for convenience (i.e. going back to the depot), taking into account the limit on the maximum available capacity of the vehicle. For further details on the derivation of the ESPPRC see e.g. [10,30]. ESPPRC is NP-hard (see [11]), when the underlying graph may have negative cost cycles (as in the studied case). Many works have proposed methods for dealing with ESPPRC (see e.g. [13,26,27]). In this paper, we propose a heuristic algorithm based on iterated local search, that will be described in the next section.

4. Solution method

As already mentioned, we propose a heuristic algorithm (*SC-heur*) based on the LP-relaxation of the model (1)–(6). The LP-relaxation is solved (in a heuristic way) by applying a column generation approach, where the subproblem consists of an ESPPRC and CPLEX is used as general purpose solver. The subproblem is solved heuristically by means of an *iterated local search* (ILS) method. ILS is a metaheuristic consisting in generating a sequence of local optima. In the last decades, it has proved to be quite effective for solving a variety of problems, including routing problems. A good introduction to ILS is that of Lourenço et al. [23]. It is an iterative method, working on an incumbent solution, say x, and considering a neighborhood of such a solution. The goal in ESPPRC is to find the shortest path between two nodes (*start* and *finish* that represent two copies of the depot), which is achieved by going through negative cost arcs. Since the dual variables from LP (1)–(6) are associated to nodes, it is quite clear that if a node is associated with a high dual variable, then all arcs entering this node are likely to have a negative cost. Therefore, all the neighborhoods we take into account are based on node operations. In particular we consider the following possible neighborhoods:

1. Insertion consists in adding a new node to the path.
2. Removal consists in removing a node from the path.
3. Moving consists in removing a node from the path, then inserting it at another position.
4. Replacing consists in removing a node from the path, then inserting another one, previously absent from the path, at the same position.
5. Swapping consists in swapping the respective positions of two different nodes in the path.

One iteration of ILS consists of the following 3 steps:

1. $x' \leftarrow Perturbation(x)$
2. $x'' \leftarrow LocalSearch(x')$
3. $x \leftarrow AcceptanceDecision(x, x'')$.

In particular, we consider the neighborhood obtained by insertion and removal of nodes (neighborhoods 1 and 2) in the *LocalSearch* phase and all the moves in the *Perturbation* phase. For the local search we adopt a steepest descent on the neighborhoods, i.e. at every iteration we perform the move bringing the most significant improvement on the path cost, be it an insertion or a removal. The acceptance criterion is the so-called random walk, where a new local optimum is systematically accepted as new incumbent solution. After a certain stopping criterion (to be described in the following) is met, the search is stopped. The method uses a fixed number k of random moves for the perturbation phase (in practice $k = 10$). Additionally, at each iteration we check if the current incumbent is better than the best solution found so far, and update it if necessary.

Concerning stopping criteria, we introduce an original mechanism to adapt automatically the number of iterations to the difficulty of the problem. Using a fixed number of iterations leads to a non-negligible problem: if the number of iterations is high then the heuristic provides good-quality solutions, but takes more time. Typically, at the beginning of the column generation process, it is useless to allow a lot of iterations, since optimal solutions to the ESPPRC can be found very quickly. So for those instances, it is not a good idea to have a lot of iterations; on the other hand, at the end of the column generation process, a lot of iterations can be required, since the ESPPRC instances generated by the master problem are much harder. We therefore propose an adaptive stopping criterion: if after a certain number of iterations a negative cost path has been found, then the procedure exits; else, the total number of iterations is increased, to allocate more time to the heuristic. A total limit on the number of iterations, that can never be exceeded, is also necessary. Algorithm 1 gives an overview of how the ILS algorithm works.

Algorithm 1 ILS algorithm for the ESPRRC.

```
iter ← 0
x ← (start, finish)
x* ← x
while iter < iterMax do
    x ← Perturbation(x)
    x ← LocalSearch(x)
    if cost(x) < cost(x*) then
        x* ← x
    end if
    if cost(x*) ≥ 0 ∧ iterMax < maxIterMax then
        iterMax ← 10.iterMax
    end if
    iter ← iter + 1
end while
```

The set of columns is initialized by applying the variable neighborhood search algorithm by Hemmelmayr et al. [20] (described in Section 1). However, we only want a starting solution so we use a limited number of iterations in order to provide good feasible solutions in a few seconds.

In order to produce an integer solution to the PVRP, we propose a constructive procedure which consists in heuristically fixing the value of selected x_r^t variables to 1. Fixing a variable is performed by setting its lower bound to 1, and it has an impact on the restricted master problem and on the dual variables, therefore on the subproblem. Hence, after fixing variables it makes sense to generate new columns taking into account these new dual values, rather than directly fixing more variables. In this context, a first solution approach consists in iteratively repeating the following two steps:

1. Heuristically solve the linear relaxation of the restricted master problem with ILS.
2. Fix a set of variables.

The algorithm then simply consists in repeating these two steps until an integer solution is found. Step 1 can be interrupted prematurely in order to accelerate the search: we generate columns at most 1000 times every time step 1 is called. In practice this condition is only matched in a few special cases.

Regarding the selection of variables to fix, we use a simple rule: fix all those variables equal to 1 in the solution, plus the fractional variable of highest value. One should note that fixing a route also has consequences on the available combinations for the customers visited by the chosen route. In particular, some days for a visited customer can become infeasible, due to the choice we made; this also means that combinations using these days become infeasible. Thus, we forbid the routes that are visiting customers on forbidden days, and we also prevent the use of forbidden combinations. Finally, we avoid generating new routes that visit a customer on a forbidden day. This is done by using a list of forbidden customers for each day in the ILS algorithm for the subproblem.

This heuristic framework allows to quickly find integer solutions, but might suffer from too restrictive heuristic choices, in the absence of a mechanism to cancel previous decisions: it is a direct rush to a local optimum. We therefore introduce a second mechanism in order to escape such local optima: previously fixed variables can also be released. This offers new possibilities, for instance moving from an integer solution to a fractional solution, then generating columns again. Regarding the selection of variables to release, we make use of the search history by systematically choosing the one variable that has been fixed for the longest time. Another possibility would be to select the most recently fixed variable, which would intensify the search around the same local optimum; we rather decide to diversify it, considering that the intensification aspect is already provided by generating columns every time a variable is fixed.

When an integer solution is found during the fixing phase, the best solution found (UB) is updated accordingly. The releasing phase starts when in the fixing phase the value of the LP objective function returned by CPLEX gets larger than UB. The releasing phase is executed until the LP-solution gets below the value of the best solution found. The list of forbidden customers, routes and combinations is also updated when releasing variables. In order to avoid cycling between the fixing and releasing phases, we use a tabu list (TL) that does not allow fixing of a variable that was released recently (i.e. in the last 40 variables that have been released).

The whole fixing–releasing heuristic stops after a stopping criterion is met: either a maximum number of iterations (of fixing and releasing) or a time limit is reached. In our experiments we fixed a time limit as stopping criterion. Algorithm 2 gives an overview of how this algorithm works. The proposed algorithm corresponds to a diving heuristic that includes backtracking, applied through releasing of some variables, and adopts a tabu list for avoiding cycling. Recently, some work has been done on primal heuristic algorithms based on column generation (see e.g. [22]). As also mentioned in the latter paper, setting bounds on column values is involved in the context of column generation: indeed this means to include additional constraints in the pricing problem which make it harder. Thus, effective primal heuristics such as Relaxation Induced Neighborhood Search by Danna et al. [9] or Local Branching by Fischetti and Lodi [14], developed in the context of general MIP solvers, cannot be easily extended to the Branch-and-Price setting.

Algorithm 2 Hybrid tabu search for the PVRP (*SC-heur*).

```
initialize the UB
TL ← ∅
while stopping criterion not reached do
  LB, UB ← SolveRestrictedMasterProblem()
  if LB < UB then
    for all x do
      if x ∉ TL ∧ x = 1 then
        fix x to 1
        propagate forbidden routes and combinations
      end if
    end for
    fix the highest non-tabu fractional x to 1
    propagate forbidden routes and combinations
  else
    release the oldest fixed x
    update TL
  end if
end while
```

5. Experimental results

In this section, we present computational experiments on benchmark instances of the PVRP and the PTSP from the literature, as well as on realistic instances presented in [24]. All the tests were performed on an Intel Xeon 2.67 GHz CPU. *SC-heur* was executed with a time limit of 2 h.

5.1. PVRP instances

We used a set of instances provided in the literature. Instances p1–p10 were proposed by Eilon et al. [12] for the VRP and adapted to the PVRP by Christofides and Beasley [7]. Russel and Igo [29] provided instance p11. Instance p12 and p13 are taken from [28]. Instances p14–p32 were introduced by Chao et al. [6].

In Table 1 we present a comparison of *SC-heur* with a heuristic algorithm (*NoRel*) which does not allow releasing route variables, but only consists of the fixing phase, that is repeated until an integer solution is found. The aim is to show that releasing of variables is a fundamental component of the proposed algorithm. We present the average objective value and computing time to reach the best solution (in min) over 10 runs. It can be easily seen that removing the releasing component leads to worse results.

In Table 2, we compare our approach with state-of-the-art algorithms and with a very recent approach by Vidal et al. [32]. The first column represents the instance name, then we show the results obtained by the tabu search of Cordeau et al. [8] (CGL), by the scatter search of Alegre et al. [1] (ALP), by the variable neighborhood search of Hemmelmayr et al. [20] (HDH), by the hybrid genetic algorithm of Vidal et al. [32] (HGSADC) and by our *SC-heur*. These are average results over 10 runs for *SC-heur*, as well as for HDH and HGSADC. We mark in boldface the best results. In the last row we show the average percentage gap between *SC-heur* and each of the other algorithms. As it can be seen from the results in Table 2, *SC-heur* brings an improvement over previous state-of-the-art algorithms (i.e. CGL, ALP and HDH) and obtains comparable results with the very recent approach HGSADC [32]. Finally, the last two columns show the previous best known solutions (*prev BKS*) and the ones (*SC-heur BKS*) found by our algorithm among all the experiments. We mark with an asterisk '*' the solutions that are optimal as proved in [2], and in boldface the best results. *SC-heur* was able to find 3 new best solutions and to find the best known solution 14 times.

Table 3 shows the computing times in minutes of each algorithm. ALP is run on a Pentium III, 600 MHz, CGL and HDH are run on a 3.2 GHz pc, and HGSADC is run on a 2.4 GHz AMD Opteron 250 CPU workstation but they report only the computing time converted to a 3.0 GHz Pentium IV by using the Dongarra factor. We report the time to reach the best solution for *SC-heur* and the total run time for CGL, ALP, HDH and HGSADC. The total run time for *SC-heur* is 2 h. The computing times of *SC-heur* are larger than those in the previous approaches (i.e. CGL, ALP and HDH), however we obtain better solutions on average. Compared to HGSADC, it should be noted that *SC-heur* uses the same parameter setting for all instances, while in HGSADC larger time limits are used for the larger instances.

5.2. Instances by Pacheco et al. [24]

Another set of experiments is performed on instances presented in [24], described in Section 1. These are realistic instances, provided by a bakery company.

Table 1
Comparison with a variant of our approach, not allowing for releasing of variables.

Instance	*NoRel*		*SC-heur*		% Dev
	Value	T^* (min)	Value	T^* (min)	
p01	524.61	0.00	524.61	0.00	0.00
p02	1333.44	0.00	1326.16	35.18	−0.55
p03	525.67	0.32	524.61	3.54	−0.20
p04	845.92	0.00	837.04	33.88	−1.05
p05	2090.17	0.72	2048.29	32.52	−2.00
p06	894.52	0.53	841.96	47.00	−5.88
p07	834.14	0.00	834.14	0.00	0.00
p08	2045.87	0.00	2042.02	16.63	−0.19
p09	847.38	2.08	847.03	8.71	−0.04
p10	1698.57	0.47	1664.9	61.72	−1.98
p11	805.14	4.95	790.77	84.57	−1.78
p12	1258.57	12.14	1228.14	97.45	−2.42
p13	3540.78	84.37	3549.3	79.28	0.24
p14	954.81	0.00	954.81	0.00	0.00
p15	1862.63	0.00	1862.63	0.00	0.00
p16	2875.24	0.00	2875.24	0.00	0.00
p17	1597.75	0.03	1597.75	0.03	0.00
p18	3155.41	0.01	3149.09	49.68	−0.20
p19	4843.79	5.38	4839.87	29.15	−0.08
p20	8367.40	0.01	8367.4	0.01	0.00
p21	2187.67	0.23	2170.8	17.73	−0.77
p22	4265.28	2.71	4232.5	81.50	−0.77
p23	6633.36	6.50	6573.33	57.86	−0.90
p24	3693.50	0.00	3687.46	6.27	−0.16
p25	3781.33	0.04	3777.15	5.49	−0.11
p26	3798.43	0.00	3795.32	0.27	−0.08
p27	22 008.86	1.56	21 963.23	53.67	−0.21
p28	22 372.24	1.69	22 271.91	84.07	−0.45
p29	22 654.80	1.22	22 564.8	80.30	−0.40
p30	75 470.31	6.10	75 193.09	75.27	−0.37
p31	77 068.45	4.04	76 496.67	93.26	−0.74
p32	78 916.65	4.11	78 065.24	86.22	−1.08
Average	11 367.27	4.35	11 296.79	38.16	−0.69

The authors made the necessary modifications to treat the problem as a classical PVRP, in order to be able to compare their algorithm with the state-of-the-art algorithms for the PVRP.

In Table 4, we present a comparison of *SC-heur* with state-of-the-art algorithms on these realistic instances. In particular, we consider the following algorithms, as in [24]: the tabu search of Cordeau et al. [8] (CGL), the scatter search of Alegre et al. [1] (ALP), the variable neighborhood search of Hemmelmayr et al. [20] (HDH), and the GRASP algorithm of Pacheco et al. [24] (GRASP). We show in boldface the best results. In the last row we show the average percentage gap between *SC-heur* and each of the other algorithms. In this case, we initialized the set of columns in the set-covering-like model for *SC-heur* with dummy routes, one for each customer. As one can see, *SC-heur* obtains on average an improvement over all the state-of-the-art algorithms. Finally, in the last two columns, we show the previous best known solutions (*prev BKS*) and the ones (*SC-heur BKS*) found by our algorithm among all the experiments. *SC-heur* finds 9 new best solutions, and finds the best known solution for all the other instances.

Table 5 shows the computing times in minutes. We report the computing times as in [24]: CGL was run on a Dell Precision T7400 with four processors at 3.0 GHz CPU (a single processor was used to run the tests), ALP was run on a Corel 8400 PC at 2.66 GHz, HDH algorithm was run on a 64 bits dual core PC, at 3.2 GHz, and GRASP was run on a Corel 8400 PC with a 2.66 GHz CPU.

5.3. PTSP instances

A last set of experiments was conducted on instances of the PTSP from the literature. Instances p1–p10 were given by Eilon et al. [12] for the VRP and adapted to the PTSP by Christofides and Beasley [7]. Instances p11–p23 were introduced by Chao et al. [5] and instances pr1–pr10 are taken from [8].

In Table 6, we present a comparison of *SC-heur* with state-of-the-art algorithms on PTSP instances. In particular, we consider the following algorithms: the heuristic algorithms of Chao et al. [5] (CGW) and of Paletta [25] (P), the tabu search of Cordeau et al. [8] (CGL), the heuristic algorithm of Bertazzi et al. [4] (BPS) and the VNS approach of Hemmelmayr et al. [20] (HDH). We show in boldface the best results. In the last row we show the average percentage gap between *SC-heur* and each of the other algorithms. We wish to mention that we have not considered the additional constraint that at least one

Table 2
Comparison of *SC-heur* with state-of-the-art algorithms on PVRP instances.

Instance	CGL	ALP	HDH	HGSADC	*SC-heur*	prev BKS	SC-heur BKS
p01	**524.61**	531.02	**524.61**	**524.61**	**524.61**	**524.61***	**524.61***
p02	1330.09	1324.74	1332.01	**1322.87**	1326.16	**1322.87**	**1322.87**
p03	**524.61**	537.37	528.97	**524.61**	**524.61**	**524.61***	**524.61***
p04	837.93	845.97	847.48	**836.59**	837.04	**835.26***	835.43
p05	2061.36	2043.75	2059.74	**2033.72**	2048.29	**2024.96**	2031.49
p06	840.3	**840.1**	884.69	842.48	841.96	**835.26***	835.41
p07	829.37	829.65	829.92	**827.02**	834.14	**826.14**	829.63
p08	2054.9	2052.21	2058.36	**2022.85**	2042.02	**2022.47**	2027.35
p09	829.45	829.65	834.92	**826.94**	847.03	**826.14**	833.83
p10	1629.96	1621.21	1629.76	**1605.22**	1664.9	**1593.43**	1619.52
p11	817.56	782.17	791.18	**775.84**	790.77	**770.89**	781.75
p12	1239.58	1230.95	1258.46	**1195.29**	1228.14	**1186.47**	1198.79
p13	3602.76	–	3835.9	3599.86	**3549.3**	3492.89	**3483.24**
p14	**954.81**	**954.81**	**954.81**	**954.81**	**954.81**	**954.81***	**954.81***
p15	**1862.63**	**1862.63**	**1862.63**	**1862.63**	**1862.63**	**1862.63***	**1862.63***
p16	**2875.24**	**2875.24**	**2875.24**	**2875.24**	**2875.24**	**2875.24***	**2875.24***
p17	**1597.75**	**1597.75**	1601.75	**1597.75**	**1597.75**	**1597.75***	**1597.75***
p18	3159.22	3157	3147.91	**3131.09**	3149.09	**3131.09**	3134.19
p19	4902.64	4846.49	4851.41	**4834.50**	4839.87	**4834.34**	**4834.34**
p20	**8367.40**	8412.02	**8367.40**	**8367.40**	**8367.40**	**8367.40**	**8367.40**
p21	2184.04	2173.58	2180.33	**2170.61**	**2170.61**	**2170.61***	**2170.61***
p22	4307.19	4330.59	4218.46	**4194.23**	4232.5	**4193.95**	4195.66
p23	6620.5	6813.45	6644.93	**6434.10**	6573.33	**6420.71**	6432.26
p24	3704.11	3702.02	3704.6	**3687.46**	**3687.46**	**3687.46***	**3687.46***
p25	3781.38	3781.38	3781.38	**3777.15**	**3777.15**	**3777.15***	**3777.15***
p26	**3795.32**	3795.33	**3795.32**	**3795.32**	**3795.32**	**3795.32***	**3795.32***
p27	23 017.45	22 561.33	22 153.31	**21 885.70**	21 963.23	**21 833.87**	21 899.10
p28	22 569.4	22 562.44	22 418.52	22 272.6	**22 271.91**	**22 242.51***	22 244.60
p29	24 012.92	23 752.15	22 864.23	**22 564.05**	22 564.8	**22 543.76***	**22 543.76***
p30	77 179.33	76 793.99	75 579.23	**74 534.38**	75 193.09	**73 875.19**	74 358.60
p31	79 382.35	77 944.79	77 459.14	76 686.65	**76 496.67**	76 001.57	**75 957.62**
p32	80 908.95	81 055.52	79 487.97	78 168.82	**78 065.24**	77 598.00	**77 591.23**
SC-heur Av. gap%	vs. CGL −0.91	vs. ALP −0.72	vs. HDH −0.72	vs. HGSADC 0.53			

customer has to be visited every day. This constraint was introduced by Chao et al. [5]. As shown in [20], this constraint leads to best known solutions with a route visiting many customers plus one single-customer route on each day of the horizon. We believe that these kinds of solutions are not useful in practice, so we decide to neglect this additional constraint. Thus, the *SC-heur* results for instances p03, p06 and p09 are not feasible with respect to this constraint. We mark the results we found for these instances with an asterisk '*'. For a fair comparison, we did not consider these instances in the computation of the average percentage gap. As one can see, *SC-heur* is competitive compared to HDH and improves the results obtained by the other algorithms. Finally, in the last two columns, we show the previous best known solutions (*prev BKS*) and the ones (*SC-heur BKS*) found by our algorithm among all the experiments. Without considering the instances for which the additional constraint that at least one customer has to be visited every day is not taken into account, *SC-heur* finds 2 new best solutions, and finds the best known solution for 10 instances.

In Table 7, we report the run times in minutes executed on a Sun 4/370 (CGW), on a Sun Sparcstation 10 (CGL), on a Sun Sparcclassic (P), and on a 3.2 Dual Core (HDH). For all the algorithms we report the total run time for each instance, while for *SC-heur* we report the time to reach the best solution. The total run time for *SC-heur* is 2 h. For BPS we do not report the running time since it is not available; quoting the authors, their "algorithm is able to tackle the test instances in an acceptable computational time (about 0.05 min on average)".

6. Discussion

In this paper we have proposed a new heuristic algorithm based on the linear programming relaxation of a set-covering-like model with additional constraints for the periodic vehicle routing problem. The proposed algorithm can be generalized in order to solve mixed integer linear programming problems. Column generation is applied in order to solve the linear programming relaxation. In particular, we solve it in a heuristic way, by means of an iterated local search algorithm. In order to obtain a feasible solution to the periodic vehicle routing problem, fixing and releasing of the route variables is performed. The developed algorithm has been tested on benchmark instances from the literature and on realistic instances of the PVRP. In addition, benchmark instances of the periodic traveling salesman problem have been solved. The comparison with state-of-the-art algorithms shows that the proposed heuristic algorithm is effective and competitive. In addition, some new best known solutions have been obtained.

Table 3
Computing times in minutes for the PVRP instances. T refers to the total time used to execute the algorithm and T^* is the time needed to obtain the best solution during search process.

Instance	CGL T	ALP T	HDH T	HGSADC T	*SC-heur* T^*
p01	0.49	4.47	1.64	0.22	0.00
p02	0.59	8.23	1.36	0.44	35.18
p03	0.54	0.75	1.68	0.18	3.54
p04	0.78	23.77	1.12	1.05	33.88
p05	0.88	21.33	1.13	2.27	32.52
p06	0.95	29.95	1.27	0.89	47.00
p07	1.28	3.32	3.05	0.88	0.00
p08	2.06	59.73	2.38	2.54	16.63
p09	1.59	16.17	3.22	1.01	8.71
p10	2.06	157.78	2.83	1.80	61.72
p11	3.44	108.20	4.23	4.60	84.57
p12	3.94	8.58	5.91	5.34	97.45
p13	24.86	–	2.13	40.00	79.28
p14	0.16	0.08	0.62	0.08	0.00
p15	0.39	0.02	1.57	0.17	0.00
p16	0.69	0.03	3.63	0.32	0.00
p17	0.37	1.60	0.95	0.27	0.03
p18	1.07	6.68	2.38	0.89	49.68
p19	2.26	0.33	4.31	2.26	29.15
p20	3.87	1.00	14.82	4.01	0.01
p21	0.55	6.22	1.21	0.90	17.73
p22	2.20	8.80	2.83	4.27	81.50
p23	5.70	0.70	5.69	4.29	57.86
p24	0.53	1.91	0.87	0.32	6.27
p25	0.52	1.15	0.78	0.59	5.49
p26	0.52	0.13	0.75	0.33	0.27
p27	1.38	3.65	1.10	3.52	53.67
p28	1.34	7.25	1.08	4.67	84.07
p29	1.27	0.32	0.99	3.86	80.30
p30	2.85	0.33	1.30	9.99	75.27
p31	2.68	127.50	1.29	10.00	93.26
p32	2.43	138.60	1.17	10.00	86.22

Table 4
Comparison of *SC-heur* with state-of-the-art algorithms for instances by Pacheco et al. [24].

Instance	CGL	ALP	HDH	GRASP	*SC-heur*	prev BKS	SC-heur BKS
41-1	**6425.17**	**6425.17**	**6425.17**	**6425.17**	**6425.17**	**6425.17**	**6425.17**
41-2	**6173.31**	6314.66	6176.67	**6173.31**	**6173.31**	**6173.31**	**6173.31**
41-3	**6355.62**	**6355.62**	**6355.62**	**6355.62**	**6355.62**	**6355.62**	**6355.62**
41-4	**6902.71**	**6902.71**	6921.95	**6902.71**	6903.83	**6902.71**	**6902.71**
41-5	**6832.63**	**6832.63**	**6832.63**	**6832.63**	**6832.63**	**6832.63**	**6832.63**
62-1	9298.06	9317.89	9335.90	9297.57	**9296.86**	9297.57	**9296.55**
62-2	**8765.20**	8823.87	8785.07	8867.92	8765.35	**8765.20**	**8765.20**
62-3	8833.46	8885.98	8849.44	**8813.65**	8829.99	**8813.65**	**8813.65**
62-4	**9089.24**	**9089.24**	**9089.24**	**9089.24**	**9089.24**	**9089.24**	**9089.24**
62-5	**8838.03**	8851.54	8878.02	**8838.03**	8849.65	**8838.03**	**8838.03**
82-1	10 779.09	10 912.78	10 826.58	11 180.33	**10 755.29**	10 779.09	**10 697.73**
82-2	10 459.80	10 480.60	10 501.43	10 430.65	**10 430.43**	10 430.65	**10 388.92**
82-3	10 147.79	**10 075.53**	10 138.96	10 128.56	10 082.58	**10 075.53**	**10 075.53**
82-4	10 555.07	**10 550.06**	10 619.69	**10 550.06**	10 568.74	**10 550.06**	**10 550.06**
82-5	10 611.28	10 592.93	10 618.36	10 631.00	**10 576.08**	10 592.93	**10 562.74**
102-1	13 312.92	13 285.43	13 191.64	13 125.98	**13 119.16**	13 125.98	**13 077.15**
102-2	13 887.50	13 834.50	13 893.59	13 827.75	**13 825.63**	13 827.75	**13 790.06**
102-3	11 920.98	11 957.09	11 981.48	11 898.51	**11 896.05**	11 898.51	**11 869.85**
102-4	12 515.18	12 423.94	12 482.99	12 415.18	**12 336.89**	12 415.18	**12 303.52**
102-5	12 591.00	12 573.39	12 640.49	**12 534.88**	12 536.88	12 534.88	**12 468.84**
SC-heur	vs. CGL	vs. ALP	vs. HDH	vs. GRASP			
Av. gap%	−0.26	−0.42	−0.40	−0.31			

Acknowledgments

Financial support from the Austrian Science Fund (FWF), by grant #P20342, as well as from the Austrian Research Promotion Agency (FFG), by grant #822739, is gratefully acknowledged. The authors would also like to thank Karl F. Doerner

Table 5
Computing times in minutes for the instances by Pacheco et al. [24]. T refers to the total time used to execute the algorithm and T^* is the time needed to obtain the best solution during search process.

Instance	CGL T	ALP T	HDH T	GRASP T	*SC-heur* T^*
41-1	0.60	0.55	1.33	0.49	2.36
41-2	0.61	0.52	1.38	0.48	1.08
41-3	0.68	0.54	1.36	0.49	0.02
41-4	0.58	0.58	1.40	0.45	7.61
41-5	0.49	0.51	1.44	0.40	0.17
62-1	0.96	1.63	2.77	1.36	13.25
62-2	1.03	1.52	2.86	1.35	21.45
62-3	0.97	1.58	2.95	1.18	22.47
62-4	0.68	1.29	2.90	1.20	10.70
62-5	1.14	1.95	2.71	1.60	42.46
82-1	1.49	3.43	4.61	2.72	33.02
82-2	1.55	3.62	4.72	3.39	54.54
82-3	1.64	3.50	4.78	3.32	26.24
82-4	1.20	3.89	4.82	2.95	31.92
82-5	1.52	3.62	5.12	3.49	47.36
102-1	1.86	6.28	7.24	5.82	41.93
102-2	1.53	6.48	7.75	5.42	30.82
102-3	1.95	7.80	7.59	6.49	79.46
102-4	2.10	7.57	7.30	6.70	63.76
102-5	2.13	8.65	7.32	6.94	54.65

Table 6
Comparison of *SC-heur* with state-of-the-art algorithms for PTSP instances.

Instance	CGW	P	CGL	BPS	HDH	*SC-heur*	prev BKS	SC-heur BKS
p01	442.1	436.50	439.02	436.50	**432.10**	**432.10**	**432.10**	**432.10**
p02	1106.7	1122.44	1111.93	1122.44	1106.84	**1105.81**	**1105.81**	**1105.81**
p03	474.0	469.16	469.69	469.64	467.42	**446.17***	466.71	**440.62***
p04	554.2	559.68	556.21	559.49	552.39	**550.07**	**549.05**	550.07
p05	1394.0	1387.90	1389.54	1384.75	1384.58	**1384.15**	**1382.33**	1384.15
p06	657.3	643.59	651.28	655.06	652.65	**581.94***	643.50	**572.16***
p07	662.4	–	660.41	**646.65**	649.17	658.09	**643.80**	658.09
p08	1635.2	–	1634.68	1633.92	1615.51	**1612.60**	**1611.96**	1612.60
p09	735.3	–	734.16	733.13	729.33	**698.04***	720.72	**684.26***
p10	1248.8	–	1240.01	1249.15	**1237.72**	1239.96	**1233.53**	1239.96
p11	491.0	**490.97**	**490.97**	**490.97**	**490.97**	**490.97**	**490.97**	**490.97**
p12	**664.1**	**664.10**	**664.10**	**664.10**	**664.10**	**664.10**	**664.10**	**664.10**
p13	**830.8**	**830.80**	**830.80**	**830.80**	**830.80**	**830.80**	**830.80**	**830.80**
p14	994.6	994.60	**994.60**	**994.60**	**994.60**	**994.60**	**994.60**	**994.60**
p15	1157.1	**1157.07**	**1157.07**	**1157.07**	**1157.07**	1157.12	**1157.07**	**1157.07**
p16	726.8	660.12	660.12	660.12	660.12	**649.96**	660.12	**643.42**
p17	776.5	776.43	776.43	776.43	776.71	**774.54**	776.43	**774.54**
p18	**873.7**	876.44	**873.73**	876.44	875.82	887.05	**873.73**	887.05
p19	974.6	**958.51**	958.88	**958.51**	965.54	974.60	**958.51**	974.60
p20	1053.6	**1033.58**	1034.51	**1033.58**	1035.51	1053.59	**1033.58**	1053.59
p21	1379.1	–	1375.08	**1375.07**	**1375.07**	1375.08	**1375.07**	**1375.07**
p22	4323.6	–	4319.72	4323.49	**4312.31**	4312.32	**4312.31**	**4312.31**
p23	8753.3	8390.53	8553.10	8498.00	**8349.26**	8405.10	**8308.48**	8405.10
pr01	–	**2064.84**	2068.46	**2064.84**	**2064.84**	**2064.84**	**2064.84**	**2064.84**
pr02	–	3232.72	3293.50	3231.50	3208.49	**3208.22**	**3205.94**	3208.22
pr03	–	4084.75	4106.72	4118.63	**4045.73**	4065.15	**4027.71**	4065.15
pr04	–	4636.67	4661.97	4621.36	**4547.77**	4557.92	**4538.19**	4557.92
pr05	–	4757.90	4698.83	4682.54	4628.24	**4623.86**	**4613.58**	4623.86
pr06	–	5688.42	5699.96	5595.45	**5529.68**	5559.11	**5521.24**	5559.11
pr07	–	4479.65	4453.15	4474.17	**4436.31**	4446.60	**4435.39**	4446.60
pr08	–	–	5405.40	5475.70	**5370.59**	5383.44	**5366.53**	5383.44
pr09	–	7405.52	7469.73	7346.32	**7244.02**	7256.65	**7234.35**	7256.65
pr10	–	8394.52	8493.74	8415.31	**8216.48**	8243.32	**8199.55**	8243.32
SC-heur	vs. CGW	vs. P	vs. CGL	vs. BPS	vs. HDH			
Av. gap%	−1.02	−0.58	−0.68	−0.46	0.19			

and Paolo Toth for their valuable comments and the support of the project. We are grateful to the two anonymous referees for their helpful comments.

Table 7
Computing times in minutes for PTSP instances. T refers to the total time used to execute the algorithm and T^* is the time needed to obtain the best solution during search process.

Instance	CGW T	P T	CGL T	HDH T	*SC-heur*T^*
p01	0.03	0.04	4.04	2.30	0.00
p02	1.56	0.05	4.57	2.10	0.00
p03	0.27	0.11	4.45	1.30	25.47
p04	0.76	0.14	7.33	3.50	0.00
p05	5.22	0.16	8.72	3.70	0.00
p06	0.66	0.64	9.91	1.40	54.03
p07	1.75	–	12.32	5.10	0.00
p08	12.41	–	14.14	5.70	0.00
p09	0.69	–	13.48	2.00	33.44
p10	2.68	–	14.13	6.90	0.00
p11	0.46	0.05	5.6	1.70	0.00
p12	1.3	0.12	8.88	2.70	0.00
p13	2.43	0.23	12.46	3.70	0.00
p14	3.05	0.4	17.63	5.10	0.01
p15	7.41	0.62	24.06	6.70	0.01
p16	0.2	0.06	3.88	1.70	50.32
p17	0.2	0.15	6.18	3.20	0.15
p18	1.87	0.31	9.17	4.70	0.00
p19	1.52	0.55	12.85	6.80	0.00
p20	2.97	0.88	16.87	8.40	0.00
p21	1.26	–	7.29	2.30	0.01
p22	11.67	–	25.2	7.60	0.00
p23	47.59	2.14	56.31	19.60	0.00
pr01	–	0.04	3.69	2.20	0.16
pr02	–	0.35	11.83	6.70	0.00
pr03	–	1.14	24.42	14.90	0.00
pr04	–	2.75	41.31	27.50	0.00
pr05	–	5.62	65.66	48.70	0.02
pr06	–	9.86	92.48	81.10	0.03
pr07	–	0.28	9.26	5.70	0.00
pr08	–	–	29.72	19.90	0.00
pr09	–	8.18	65.77	50.80	0.00
pr10	–	19.05	113.72	112.70	0.04

References

[1] J. Alegre, M. Laguna, J. Pacheco, Optimizing the periodic pick-up of raw materials for a manufacturer of auto parts, European Journal of Operational Research 179 (2007) 736–746.
[2] R. Baldacci, E. Bartolini, A. Mingozzi, A. Valletta, An exact algorithm for the period routing problem, Operations research 59 (1) (2011) 228–241.
[3] E. Beltrami, L. Bodin, Networks and vehicle routing for municipial waste collection, Networks 4 (1974) 65–94.
[4] L. Bertazzi, G. Paletta, M. Speranza, An improved heuristic for the period traveling salesman problem, Computers & Operations Research 31 (8) (2004) 1215–1222.
[5] I. Chao, et al., A new heuristic for the period traveling salesman problem, Computers & Operations Research 22 (5) (1995) 553–565.
[6] I. Chao, B. Golden, E. Wasil, An improved heuristic for the period vehicle routing problem, Networks 26 (1995) 25–44.
[7] N. Christofides, J. Beasley, The period routing problem, Networks 14 (1984) 237–256.
[8] J. Cordeau, M. Gendreau, G. Laporte, A tabu search heuristic for periodic and multi-depot vehicle routing problems, Networks 30 (2) (1997) 105–119.
[9] E. Danna, E. Rothberg, C. Pape, Exploring relaxation induced neighborhoods to improve mip solutions, Mathematical Programming 102 (1) (2005) 71–90.
[10] G. Desaulniers, J. Desrosiers, M. Solomon, Column Generation, Vol. 5, Springer Verlag, 2005.
[11] M. Dror, Note on the complexity of the shortest path models for column generation in vrptw, Operations Research 42 (1994) 977–978.
[12] S. Eilon, W. Gandy, N. Christofides, Distribution Management: Mathematical Modeling and Practical Analysis, Griffin, London, 1971.
[13] D. Feillet, P. Dejax, M. Gendreau, C. Gueguen, An exact algorithm for the elementary shortest path with resource constraints: application to some vehicle routing problems, Networks 44 (2004) 216–229.
[14] M. Fischetti, A. Lodi, Local branching, Mathematical Programming 98 (1) (2003) 23–47.
[15] P. Francis, K. Smilowitz, M. Tzur, The period vehicle routing problem with service choice, Transportation Science 40 (4) (2006) 439–454.
[16] P. Francis, K. Smilowitz, M. Tzur, The period vehicle routing problem and its extensions, in: The Vehicle Routing Problem: Latest Advances and New Challenges, Vol. 43, Springer, 2008 (Chapter).
[17] M. Gaudioso, G. Paletta, A heuristic for the periodic vehicle routing problem, Transportation Science 26 (1992) 86–92.
[18] M. Gendreau, A. Hertz, G. Laporte, New insertion and postoptimization procedures for the traveling salesman problem, Operations Research 40 (6) (1992) 1086–1094.
[19] E. Hadjiconstantinou, R. Baldacci, A multi-depot period vehicle routing problem arising in the utilities sector, Journal of the Operational Research Society 49 (1998) 1239–1248.
[20] V. Hemmelmayr, K. Doerner, R. Hartl, A variable neighborhood search heuristic for periodic routing problems, European Journal of Operational Research 195 (3) (2009) 791–802.
[21] V. Hemmelmayr, K. Doerner, R. Hartl, M. Savelsbergh, Delivery strategies for blood products supplies, OR Spectrum 31 (4) (2009) 707–725.
[22] C. Joncour, S. Michel, R. Sadykov, D. Sverdlov, F. Vanderbeck, Column generation based primal heuristics, Electronic Notes in Discrete Mathematics 36 (2010) 695–702.
[23] H. Lourenço, O. Martin, T. Stützle, Iterated local search, in: Handbook of Metaheuristics, Springer, 2003 pp. 321–353. (Chapter).
[24] J. Pacheco, A. Alvarez, I. García, F. Angel-Bello, Optimizing vehicle routes in a bakery company allowing flexibility in delivery dates, Journal of the Operational Research Society (2011) http://dx.doi.org/10.1057/jors.2011.51.

[25] G. Paletta, The period traveling salesman problem: a new heuristic algorithm, Computers & Operations Research 29 (10) (2002) 1343–1352.
[26] G. Righini, M. Salani, Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints, Discrete Optimization 30 (3) (2006) 255–273.
[27] G. Righini, M. Salani, New dynamic programming algorithms for the resource constrained elementary shortest path problem, Networks 51 (2008) 155–170.
[28] R. Russel, D. Gribbin, A multiphase approach to the period routing problem, Networks 21 (1991) 747–765.
[29] R. Russel, W. Igo, An assignment routing problem, Networks 9 (1979) 1–17.
[30] M. Salani, Branch-and-Price algorithms for vehicle routing problems, Ph.D. Thesis, Universitá degli Studi di Milano, 2006.
[31] C. Tan, J. Beasley, A heuristic algorithm for the periodic vehicle routing problem, Omega 12 (1984) 497–504.
[32] T. Vidal, T. Crainic, M. Gendreau, N. Lahrichi, W. Rei, A hybrid genetic algorithm for multi-depot and periodic vehicle routing problems, Operations Research 60 (3) (2012) 611–624.