Variable Neighborhood Search Coupled with ILP-based Very Large Neighborhood Searches for the (Periodic) Location-Routing Problem

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. This work deals with the application of a variable neighborhood search (VNS) to the capacitated location-routing problem (LRP) as well as to the more general periodic LRP (PLRP). For this, previous successful VNS algorithms for related problems are considered and accordingly adapted as well as extended. The VNS is subsequently combined with three very large neighborhood searches (VLNS) based on integer linear programming: Two operate on whole routes and do a rather coarse, yet powerful optimization, with the more sophisticated one also taking the single customers into account, and the third operates on customer sequences to do a more fine-grained optimization. Several VNS plus VLNS combinations are presented and very encouraging experimental results are given. Our method clearly outperforms previous PLRP approaches and is at least competitive to leading approaches for the LRP.

1 Introduction

The *location-routing problem* (LRP) combines two classical NP-hard problems: the *facility location problem* (FLP) and the *vehicle routing problem* (VRP). The LRP occurs when it is necessary to place some facilities at given locations, assign customers to them, and serve these customers by a fleet of vehicles, often imposing a limit on the maximal load (capacity) of a vehicle. Additionally, also capacity constraints on the facilities can and will be considered in the following. Hence, contrary to the simple out-and-back routes visiting single customers in the classical FLP, one is faced here with multi-stop routes. Solving the LRP demands a strategic (facility placement) and tactical (routing) planning task at the same time, contributing to the potential practical relevance of the LRP. Considering both aspects simultaneously is in favor of solving them in a subsequent way (usually starting with placing the facilities first), since the latter is more prone to yield suboptimal solutions [1].

An additional interesting strategic level can be incorporated by considering the LRP for a given planning period, resulting in the *periodic LRP* (PLRP). Here, specific customers must be served several times during the planning period.

The following definitions are targeted to the PLRP only since the LRP can be considered as special case when having only a single day planning horizon. A planning horizon of t days, referred to by $T = \{1, \ldots, t\}$, is considered and it is defined on a complete, undirected, weighted graph $G = (V, E)$, with $V = V_C \cup V_D$ being the set of nodes, composed of n customers $V_C = \{0, \ldots, n-1\}$ and m potential depots $V_D = \{n, \ldots, n+m\}$, and $E = \{\{i,j\} \mid i,j \in V_C,\ i \neq j\} \cup \{\{i,j\} \mid i \in V_C,\ j \in V_D\}$ being the set of edges. Travel cost $c_{ij} \geq 0$, independent of the day, are given for any pair of nodes $i, j \in V$. Each depot $i \in V_D$ has an associated capacity W_i and opening costs O_i. Further, a homogeneous fleet of K vehicles, each having capacity Q, is available per depot and day. The fixed cost of using a single vehicle at least once during T is given by F, and each vehicle is limited to perform one single route per day. Further, each customer $j \in V_C$ has defined a total demand d_j, a visit frequency f_j, and a non-empty set $C_j \subseteq \{T' \mid T' \subseteq T,\ |T'| = f_j\}$ of allowed combinations of visit days. The actual demand of customer j on day l using visit combination $r \in C_j$ is assumed to be given by d_{jlr}.

The PLRP then aims at selecting facilities (depots) to be opened as well as a single visit combination per customer, and finding (at most) $N \leq K$ vehicle routes on each of the t days on G such that:

- Each route starts and ends at the same opened depot within the same day,
- each customer j belongs to f_j routes over the planning horizon at those days belonging to the selected visit day combination, overall satisfying its demand d_j and respecting the given d_{jlr} values,
- the total load of each route does not exceed vehicle capacity limit Q,
- for each opened depot i the total load of each route assigned to it on any day does not exceed depot capacity limit W_i,
- the total costs of opening depots, fixed costs for used vehicles, and corresponding travel costs are minimized.

In this contribution we present a variable neighborhood search (VNS) suited for the periodic as well as the non-periodic LRP. For this we combine successful VNS variants/concepts for similar problems. To further improve the overall performance, very large neighborhood searches based on integer linear programming are introduced, and they are combined with the VNS in a fruitful way.

The remainder is organized as follows: Related work is presented in the next section, the VNS is the topic of Section 3, and the very large neighborhood searches are detailed in Section 4. Experimental results for both, LRP and PLRP, are given in Section 5. Section 6 finishes the work with concluding remarks.

2 Related Work

The LRP with capacitated vehicles and depots was primarily dealt with in recent years only, earlier work often considered one of both restrictions exclusively. Among the currently leading methods for the so-called *general LRP* are (hybrid) metaheuristics by Prins et al. [2, 3] and Duhamel et al. [4, 5] as well as combinations of exact techniques and heuristics, e.g. [6]. A recent survey of different LRPs and solution methods can be found in Nagy and Salhi [7]. The authors emphasize the high practical relevance of the LRP in supply chain management as well as a substantially increasing interest in the last years.

The PLRP was introduced only recently by Prodhon together with an iterative algorithm for solving it [8]. At the HM 2008 workshop, the same author(s) presented a sophisticated genetic algorithm [9] yielding improved results. Finally, at the HM 2009 workshop the so far best performing algorithm for the PLRP was again introduced by Prodhon [10], being a hybrid of evolutionary local search and path relinking, even more concentrating on the periodic aspect.

A similar VNS as the one applied here has been used for the periodic VRP (PVRP) [11], the PVRP with time windows (PVRPTW) [12], as well as for the related multi depot VRP with time windows [13]. As already mentioned, we draw upon the experience described therein and obtained by ourselves.

Furthermore, a very large neighborhood search similar to the one we present in Section 4.2 was introduced by De Franceschi et al. [14] for VRPs in general. The very large neighborhood search of Section 4.1 is related to it, but operates on a coarser level of the problem. A variant of it was recently successfully applied to the PVRPTW by us [15].

Finally, for a survey on ongoing trends of combining metaheuristics and exact methods, especially integer linear programming techniques, we refer to [16].

3 Variable Neighborhood Search for the (P)LRP

The *variable neighborhood search* (VNS) [17] metaheuristic 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, among others for very related problems, see Section 2. In the following we give an overview on our VNS for the (P)LRP.

To smooth the search space, the VNS relaxes the vehicle load as well as the depot load restrictions and adds penalties corresponding to the excess of these constraints to the cost function; whereat we settled to use a constant weighting of 10000 for all instances considered in this work.

An initial solution is created in the following way:

1. Choose a single visit day combination $r \in C_j$ per customer $j \in V_C$ at random.
2. Determine the actual lower bound for the accumulated depot capacity based on the previously selected visit combinations: $W_{\text{LB}} = \underset{l \in T}{\text{argmax}} \sum_{j \in V_C} d_{jlr}$, for the LRP this equals the sum of the customer demands: $\sum_{j \in V_C} d_j$.
3. Select depots to be opened at random one by one until their combined capacity is greater or equal than W_{LB}.
4. Repeat the following steps for each day l in T:
   (a) Assign each customer to be visited on day l to its nearest opened depot, thereby considering penalized depot load violations.
   (b) Apply the well-known Clarke and Wright savings algorithm [18] until no more routes can be feasibly merged due to the vehicle load restriction.
   (c) In the event of ending up with more than N routes, least customer routes are selected and customers contained therein are re-added in a greedy

way on that day, allowing (penalized) excess of vehicle load. [Note that due to the structure of the considered instances this never happened.]

We deem the initialization procedure rather straightforward, being not that sophisticated as those described in [10]. Nevertheless, we felt (and also experienced in preliminary tests) that having initial solutions of high(er) quality is often not beneficial for the subsequent optimization process, and we apply the procedure solely for initialization purposes, too.

In the shaking phase we utilize five different neighborhood structures, each with several moves of increasing perturbation size (denoted by δ), yielding a total of 18 shaking neighborhoods (i.e. $k_{\max} = 18$) for both LRP variants. Next, these five basic neighborhood structures are described.

Exchange-segments: Exchange two random segments of variable length between two routes at the same depot and on the same day. One of the segments might be empty, realizing a customer relocation.

Exchange-segments-two-depots: In principle similar to the previous *exchange-segments* but both segments are located at two distinct depots. This neighborhood structure facilitates the partitioning of customers to the opened depots. *Exchange-segments* is applied in case only a single depot is opened.

Change-two-depots: A previously closed depot is opened and subsequently another opened depot is closed such that the actual lower bound regarding the depot capacity W_{LB} is still satisfied. After opening the selected depot it is tried to relocate routes to it in a cost saving manner, which means to place the new depot at minimum cost between two consecutive customers. Afterwards the routes of the depot to be closed are relocated to an opened depot one by one in the least expensive way (as before also using the penalized objective function). After such a neighborhood move the number of opened/closed depots stays the same. A prerequisite is that at least one of all the available depots is not opened yet, the following *change-depot* is applied otherwise.

Change-depot: Here the status of a single depot is changed, i.e. from closed to opened or vice versa. The necessary route relocation operations are applied as for *change-two-depots*. Finally, this neighborhood move allows to alter the number of opened/closed depots. However, also here it is guaranteed that the actual lower bound regarding the depot capacity W_{LB} is still satisfied.

Change-visit-combinations: For tackling the periodic aspect of the PLRP it is necessary to enable the VNS to change the selected visit combination per customer. As for the periodic VRPs, it turned out that randomly changing several visit combinations with greedy insertion for the new visit days (also allowing to reassign the same visit combination) performs very well.

In this work we only consider a fixed shaking neighborhood order, which is detailed in Table 1 for the LRP and in Table 2 for the PLRP. Apart from the obvious difference of using *change-visit-combinations* for the PLRP only, a greater focus is laid on exchanging segments for the LRP instead.

For intensification we apply the well-known 3-opt intra-route exchange procedure in a best improvement fashion, only considering routes changed during shaking, and re-applying the operator until no more improvement is possible.

Table 1. (Fixed) Shaking neighborhood order used for the LRP.

k	$\mathcal{N}_k$
1–5	*Exchange-segments* of maximal length $\delta = k$
6	*Exchange-segments* of maximal lengths bounded by corresponding route size ($\delta = 6$)
7–11	*Exchange-segments-two-depots* of maximal length $\delta = k - 6$
12	*Exchange-segments-two-depots* of maximal lengths bounded by corresponding route size ($\delta = 6$)
13–15	*Change-two-depots* is applied up to $\delta = k - 12$ times
16–18	*Change-depot* is applied up to $\delta = k - 15$ times

Table 2. (Fixed) Shaking neighborhood order used for the PLRP.

k	$\mathcal{N}_k$
1–6	*Change-visit-combinations* for up to $\delta = k$ customers
7–9	*Exchange-segments* of maximal length $\delta = k - 6$
10–12	*Exchange-segments-two-depots* of maximal length $\delta = k - 9$
13–15	*Change-two-depots* is applied up to $\delta = k - 12$ times
16–18	*Change-depot* is applied up to $\delta = k - 15$ times

Afterwards, each new incumbent solution is also subject to a 2-opt* inter-route exchange heuristic [19]. Hereby for each pair of routes of the same day and depot all possible exchanges of the routes' end segments are tried, also applied repeatedly. For the LRP this is further applied with a probability of 0.2 to each newly derived solution lying within 3% to the current incumbent.

To often enhance the overall VNS performance quite substantially, in addition to better solutions sometimes also solutions having a worse objective value are accepted. As in [11, 12] this is done in a systematic way using the Metropolis criterion like in simulated annealing [20]. A linear cooling scheme is used in a way such that the acceptance rate of worse solutions is nearly zero in the last iterations. Though this is somewhat of a hybrid variant on its own, we still denote it as VNS.

4 ILP-based Very Large Neighborhood Searches

The general idea of *very large(-scale) neighborhood search* (VLNS) [21] is to apply a more sophisticated procedure than naive enumeration to search for a best (or better) solution within a reasonably large but restricted part of the whole search space induced by an incumbent solution. Various techniques especially including (mixed) integer programming methods, dynamic programming, and constraint programming have been successfully used in VLNS as embedded optimization procedures. In the following we will introduce two VLNS procedures based on *integer linear programming* (ILP) applicable to the LRP as well as to the PLRP.

4.1 VLNS Operating on Routes

The first VLNS deals with (re-)locating whole routes to depots, as well as opening/closing depots in the course of the application. In fact, we implemented a simpler version (denoted as V_1) and a more sophisticated one (denoted as V_2) of it, the latter building upon a set covering formulation which is to some degree similar to the one used for the PVRPTW in [15] and being especially appealing for the PLRP in principle. V_1 is a special case of V_2 and basically resembles the procedure applied in [6] for the LRP. There the authors (approximately) solve a Lagrangian relaxation formulation of the FLP subproblem via considering the aggregated routes as supercustomers. For this several subproblems need to be solved many times, as well as a lower and upper bound be computed. Contrary, we directly solve the resulting ILP model which is presented in the following, and is also applicable to the PLRP:

$$\min \quad \sum_{i\in V_D} O_i\, y_i + \sum_{l\in T}\sum_{i\in V_D}\sum_{j\in R(l)} C^L_{ij}\, x_{ij} + \sum_{i\in V_D} F\, z_i \tag{1}$$

$$\text{subject to} \quad \sum_{i\in V_D} x_{ij} = 1 \qquad \forall l\in T;\ \forall j\in R(l) \tag{2}$$

$$\sum_{j\in R(l)} x_{ij} \le z_i \qquad \forall i\in V_D;\ \forall l\in T \tag{3}$$

$$\sum_{j\in R(l)} L_j\, x_{ij} \le W_i \qquad \forall i\in V_D;\ \forall l\in T \tag{4}$$

$$\sum_{i\in V_D} W_i\, y_i \ge W_{\mathrm{LB}} \tag{5}$$

$$z_i \ge y_i \qquad \forall i\in V_D \tag{6}$$

$$x_{ij}\in\{0,1\} \qquad \forall i\in V_D;\ \forall l\in T;\ \forall j\in R(l) \tag{7}$$

$$y_i\in\{0,1\} \qquad \forall i\in V_D \tag{8}$$

$$z_i\in\mathbb{N} \qquad \forall i\in V_D \tag{9}$$

The objective (1) is to minimize costs for opening depots, routing costs (here only route location costs), as well as vehicle fixed costs. The set of all considered (aggregated) routes per day l is denoted by $R(l)$, the least cost of locating it at depot j is C^L_{ij}. We introduce following binary variables: x_{ij} (7) indicating whether or not depot i hosts route j, y_i (8) if depot i is opened, as well as integer variables z_i (9) stating the maximum number of routes located at depot i of all days, used for the vehicle fixed costs in (1). The following restrictions are applied: Each route must be located at one depot (2), the value of the z_i variables is determined by (3), and the load of a depot must be respected (4), with L_j denoting the load of route j. The last two constraints are to strengthen the model: the accumulated capacity of the selected depots must be at least the actual corresponding lower bound (5), and the depot with its corresponding maximum number of routes are coupled via (6); refer to [22], though they used the minimal number of depots in (5).

The previous model is built for a given feasible solution and the solution's routes are used for the corresponding day only.

As already mentioned, the more sophisticated variant formulates a similar, yet potentially much larger neighborhood as a set covering model. Therefore, the main difference between V_1 and V_2 is that although both operate on whole routes, the latter takes the single customers into account. The whole model including constraints for both the LRP and the PLRP can be stated as (also refer to [22] for a similar model for the LRP only):

$$\min \sum_{i \in V_D} O_i \, y_i + \sum_{l \in T} \sum_{i \in V_D} \sum_{j \in R(l)} (C^L_{ij} + C^R_j) \, x_{ij} + \sum_{i \in V_D} F \, z_i \tag{10}$$

$$\text{subject to} \qquad (3)\text{–}(4)$$

$$\sum_{r \in C_n} p_{nr} \geq 1 \qquad \forall n \in V_C \tag{11}$$

$$\sum_{i \in V_D} \sum_{j \in R(l)} a_{nj} \, x_{ij} - \sum_{r \in C_n} b_{nrl} \, p_{nr} \geq 0 \qquad \forall n \in V_C;\ \forall l \in T \tag{12}$$

$$\sum_{i \in V_D} \sum_{j \in R(l)} x_{ij} \leq N \qquad \forall l \in T \tag{13}$$

$$\sum_{j \in R(l)} a_{nj} \, x_{ij} - y_i \sum_{j \in R(l)} a_{nj} \leq 0 \qquad \forall n \in V_C; \forall i \in V_D; \forall l \in T \tag{14}$$

$$\sum_{i \in V_D} W_i \, y_i \geq W_{\mathrm{LB}} \tag{15}$$

$$\sum_{i \in V_D} W_i \, y_i - \sum_{n \in V_C} \sum_{r \in C_n} b_{nrl} \, p_{nr} \, d_{nlr} \geq 0 \qquad \forall l \in T \tag{16}$$

$$\sum_{l \in T} \sum_{i \in V_D} \sum_{j \in R(l)} x_{ij} - \left\lceil \frac{\sum_{n \in V_C} d_n}{Q} \right\rceil \geq 0 \tag{17}$$

$$\sum_{l \in T} \sum_{i \in V_D} \sum_{j \in R(l)} d'_{nj} \, x_{ij} \geq d_n \qquad \forall n \in V_C \tag{18}$$

$$\sum_{i \in V_D} \sum_{j \in R(l)} d'_{nj} \, x_{ij} - d_{jlr} \, b_{nrl} \, p_{nr} \geq 0 \qquad \forall n \in V_C;\ \forall r \in C_n;\ \forall l \in T \tag{19}$$

$$(6)\text{–}(9)$$

$$p_{nr} \in \{0, 1\} \qquad \forall n \in V_C;\ \forall r \in C_n \tag{20}$$

Beside the adopted constraints and variables from V_1 following additions were made: For each customer $n \in V_C$, binary variables p_{nr} (20) indicate whether or not visit combination $r \in C_n$ is chosen. The objective function (10) now also includes routing costs C^R_j for visiting the customers in route j (without the depot connection). Cover constraints (11) guarantee that at least one visit day combination is selected per customer, visit constraints (12) link the routes and the visit combinations, whereat a_{nj} and b_{irl} are binary constants indicating whether or not route j visits customer n and if day l belongs to visit combination $r \in C_n$ of customer n, respectively, and the number of routes per day may not exceed N (13). Again, constraints (14)–(17) strengthen the model: the routes containing customer n located at depot j and the depot itself are coupled in (14), (15) is like for V_1 and only used in case of the LRP, further (16) is a special variant instead of the latter for the PLRP, incorporating the periodic aspect, and finally, the minimal amount of vehicles (routes) necessary is set by (17). The motivation for a set covering model was to be able to exploit the routes of more than one feasible solution. A consequence with respect to the PLRP is that the selected visit combination of several customers might (or better need) to eventually change. However, the fact that the daily demand of a customer depends on the chosen visit combination does not "suit the model very well“: In order to build a feasible PLRP solution out of the ILP solution it might be necessary to change the amount delivered to a customer, which is a potential problem if the amount has to be increased due to given vehicle load constraints. At least we alleviate this problem by introducing constraints (18) and (19), reducing the chance of a conflict by forcing a certain amount to be delivered per customer (and chosen visit combination), with d'_{nj} denoting the amount delivered to customer n in route j. If all fails, the customer is tried to be added in a feasibly way via greedy insertion. Finally, also over-covered customers need to be dealt with: we simply remove all but their first occurrence.

The model of V_2 is created for a given set of feasible solutions, again using the solution's routes for the corresponding day only. This solution set always contains the current incumbent solution as well as a preferred number of optional solutions which are selected via binary tournament from the set of all improved solutions (found during the run up to that time). This way of handling it has the advantage of keeping a certain diversity (assuming enough available solutions) yet still favoring good solutions, as well as having to store no additional solutions. The current incumbent further acts as a starting solution for the ILP solver.

Basically, this model could be applied as the master problem of classical column generation as well [23], of course a suitable subproblem for generating new columns would have to be defined. However, our intention is to have a (relatively) fast supplementary neighborhood. Therefore we restrict ourselves to producing the columns (routes) with the VNS only, i.e. having a pure metaheuristic column generation. Regarding runtime, there is the possibility to restrict the number of depots a route can be located to (the x% least costly ones), whereas the routes of the best solution in the given set are allowed to be located at any depot.

4.2 VLNS Operating on Customers

Our second type of VLNS (denoted by V_3) operates on the customer level, it is therefore responsible for finer-grained optimization, yet it bears quite some resemblance with V_2 in that it also makes use of a set covering formulation. It is similar to the so-called *ILP-based refinement heuristic* presented by De Franceschi et al. [14]. The idea is to extract sequences of customers from given routes, reconnect the disconnected route parts, and then find an optimal allocation of these sequences to the possible insertion points, i.e. between any two (remaining) consecutive customers. Whenever sequences' customer sets are not disjoint, this neighborhood is hard to solve and an ILP formulation as a set covering model is appropriate. A suitable one for our setting is the following:

$$\min \quad \sum_{j \in R} C_j^R + \sum_{s \in S} \sum_{i \in I_s} C_{si}^I x_{si} \tag{21}$$

$$\text{subject to} \quad \sum_{s \in S : n \in s} \sum_{i \in I_s} x_{si} \geq 1 \qquad \forall n \in V_C \tag{22}$$

$$\sum_{s \in S : i \in I_s} x_{si} \leq 1 \qquad \forall i \in I \tag{23}$$

$$L_j + \sum_{s \in S} \sum_{i \in I_s \cap I(j)} L'_s x_{si} \leq Q \qquad \forall j \in R \tag{24}$$

$$\sum_{j \in R(i)} L_j + \sum_{s \in S} \sum_{i \in I_s \cap I(j)} L'_s x_{si} \leq W_i \qquad \forall i \in V_D \tag{25}$$

$$x_{si} \in \{0, 1\} \qquad \forall s \in S; \forall i \in I_s \tag{26}$$

The objective (21) is to minimize the insertion costs C_{si}^I of the sequences $s \in S$ in their possible or allowed insertion points $i \in I_s$. Each extracted customer must be covered by at least one of the sequences containing it (22). Further, each insertion point can hold at most one sequence (23). Constraints (24) and (25) are responsible to enforce limits on vehicle and depot load, respectively. Here, L_j again denotes the load of route j and L'_s is the load of sequence s, $I(j)$ are all insertion points provided by route j, and $R(i)$ are all routes located at depot i.

Several methods were proposed in the literature for similar or related neighborhoods to extract customers from given routes. Among them to pick the customers inducing the greatest detour [24], select a seed node and neighbored nodes, to select all odd or even customers of a route, or just select them at random on a per route basis; refer to [14] for more details. Here we settle for a simple model: select the customers independent of the actual routes, since it is to be expected that no selection scheme is generally better as all others, and preliminary results showed no great difference, too. Therefore we proceed in the following way: choose a preferred number of consecutive iterations $iter_{V_3}$ of V$_3$, randomly partition all customers into equally sized sets N_i with $|N_i| = n/iter_{V_3}$, and finally perform V$_3$ $iter_{V_3}$ times, for each iteration i using set N_i as customers to be extracted as well as considering a specific set of feasible solutions. The latter is created as for V$_2$ described in Section 4.1. The routes of the current incumbent represent the considered set of routes R, and together with the sequences contained therein they are used as a starting solution for the ILP solver. Initially all possible sequences from the given solutions' routes are extracted adhering to the pre-selected customers, and the potential insertion points are determined while processing routes of set R. This way we omit the costly generation of sequences via making use of the linear programming dual values as in [14], yet (most likely) also obtain non-disjoint sequences, hence like for V_2 we again exploit the information of several solutions. Additionally, for each sequence of length more than one we create all possible sequences containing one of the customers each. Again, a customer is only accepted in the resulting solution on its first occurrence. Note that V_3 is applied on a per day basis only, which has to be considered when using it for the PLRP. The possible insertion points I_s per sequence s are determined by taking the x% least costly ones, also omitting infeasible pairings.

5 Experimental Results

The algorithms have been implemented in C++, compiled with GCC 4.3 and executed on a single core of a 2.83 GHz Intel Core2 Quad Q9550 with 8 GB RAM. The general purpose MIP solver CPLEX version 12.1 is used to solve the VLNS' ILP models.

For both problem variants we took freely available benchmark data sets comprising 30 instances each, provided by Caroline Prodhon [25]. The instances differ in the number of customers n, depots m, and clusters, as well as in the vehicle capacity ('a' denotes low, and 'b' high) and are named: n-m-#clusters[a,b]. The PLRP instances further span a working week, i.e. five working days and two idle days and customers must be visited one, two, or three times, offering five, three, and one visit combinations, respectively. We refer to [25, 10] for computing the daily demands d_{jlr}. More details about the instances can be found in [6] regarding the LRP and in [10] for the PLRP.

VNS is always run for 10^5 iterations, setting an initial temperature of 500 and apply linear cooling every 100 iterations. The combination with the VLNS variants is performed in the following way: V_1 is applied on each new incumbent solution since it can be considered quite fast (denoted by VNS+V_1). The other two, more demanding VLNS are executed at fixed times, namely after every 10^4 iterations of the VNS, resulting in 10 applications per run. Thereby we either solely apply V_2 (resulting in VNS+$V_{1,2}$) or V_2 directly followed by V_3 (denoted by VNS+$V_{1,2,3}$). Naturally, more combinations would be possible, but for now we solely investigated the effect of gradually adding all neighborhoods. The number of additional feasible solutions used for V_2 is three and five for V_3, $iter_{V_3}$ is set to four. For both, possible depots for routes as well as possible insertion points for sequences we only consider the set of the least costly 33%, whereas the initial depot location is always retained. Basically, the runtime per application of each VLNS is restricted to two seconds, although this limit is seldom reached. Each new VLNS incumbent is subject to 2-opt*, those of V_3 also to 3-opt.

For each algorithm setting we perform 30 runs per instance, and state the average costs (avg.), the corresponding standard deviations in percentage (sdv.[%]), and the runtimes in seconds (t[s]). The results for the LRP are shown in Table 3, those for the PLRP in Table 4. Furthermore, %-gap$_{\text{min, BKS}}$ and %-gap$_{\text{avg., BKS}}$ state the average gap to the so far best known solutions (mostly up to date values can be found at [25], some newer values for the LRP were taken from [5]) of

Table 3. Results of VNS and VNS plus VLNS combinations on the LRP.

Instance	VNS			VNS+V_1			VNS+$V_{1,2}$			VNS+$V_{1,2,3}$		
	avg.	sdv. [%]	t [s]	avg.	sdv. [%]	t [s]	avg.	sdv. [%]	t [s]	avg.	sdv. [%]	t [s]
20-5-1a	54881.27	0.56	1.1	54905.70	0.59	1.1	54873.67	0.51	1.2	54890.33	0.39	1.3
20-5-1b	39104.00	0.00	1.2	39104.00	0.00	1.2	39104.00	0.00	1.2	39104.00	0.00	1.3
20-5-2a	48920.13	0.12	1.1	48933.90	0.29	1.1	48932.53	0.27	1.1	48933.90	0.29	1.2
20-5-2b	37542.00	0.00	1.2	37542.00	0.00	1.2	37542.00	0.00	1.2	37542.00	0.00	1.4
50-5-1a	90732.47	1.05	1.5	90780.50	0.96	1.5	90684.70	0.92	1.6	**90388.87**	0.71	1.9
50-5-1b	64197.23	1.50	2.3	64348.83	1.07	2.2	64125.30	1.20	2.3	**63471.57**	0.67	2.7
50-5-2a	89931.77	0.89	1.6	**89553.50**	0.82	1.7	**89484.20**	0.93	1.7	89858.77	1.20	2.0
50-5-2b	68884.60	1.17	3.7	**68060.00**	0.79	3.4	**68108.50**	0.65	3.5	**68013.17**	1.00	3.7
50-5-2aBIS	84582.17	0.39	2.2	84500.33	0.33	2.2	**84437.20**	0.35	2.3	**84208.23**	0.26	2.6
50-5-2bBIS	52564.40	0.88	4.4	**52350.83**	0.76	4.4	**52326.63**	0.63	4.5	**52131.20**	0.64	4.9
50-5-3a	87216.20	0.83	1.3	87039.93	0.72	1.3	87095.17	0.96	1.5	**86727.53**	0.33	1.7
50-5-3b	62132.77	0.94	2.3	62236.80	0.95	2.3	62208.67	1.00	2.4	62095.30	0.75	2.6
100-5-1a	279817.40	0.52	3.6	**279017.83**	0.41	3.6	**279038.27**	0.34	3.8	**278291.83**	0.33	4.9
100-5-1b	217985.50	0.55	6.4	217940.50	0.57	6.5	218012.67	0.68	6.6	**216285.93**	0.50	8.0
100-5-2a	196410.30	0.58	2.4	196388.63	0.42	2.5	196147.17	0.41	2.7	**195022.27**	0.44	4.0
100-5-2b	158949.53	0.65	3.1	**158355.97**	0.45	3.2	158682.53	0.49	3.3	**158217.13**	0.56	4.5
100-5-3a	203481.40	0.65	2.4	**202460.07**	0.38	2.4	**202131.23**	0.51	2.7	**201748.07**	0.34	3.9
100-5-3b	156289.00	0.72	3.1	**155749.70**	0.93	3.1	155873.37	0.95	3.1	**154917.30**	0.64	4.4
100-10-1a	316581.83	1.67	4.2	**292857.77**	0.65	5.5	**293227.17**	0.68	7.3	**291775.00**	0.45	8.5
100-10-1b	268297.33	4.00	6.8	**240989.67**	1.26	6.8	**241682.67**	2.81	8.3	**238058.60**	0.98	9.8
100-10-2a	248775.67	0.67	3.5	**246791.30**	0.35	3.8	**246363.47**	0.44	4.6	**245614.20**	0.44	5.6
100-10-2b	208692.87	0.79	5.2	**207346.30**	0.68	5.3	**206373.40**	0.46	5.6	**205718.67**	0.64	6.9
100-10-3a	259271.30	0.52	3.5	**256511.00**	0.72	3.7	**255896.97**	0.66	5.2	**255140.30**	0.40	6.6
100-10-3b	209812.43	0.99	5.0	209047.90	0.76	5.1	**208041.30**	0.61	5.5	**207410.27**	0.68	6.7
200-10-1a	489068.47	0.71	11.8	**483895.40**	0.33	11.7	**483788.30**	0.53	11.7	**481141.73**	0.24	18.4
200-10-1b	387485.03	0.72	9.5	**385752.33**	0.70	8.7	**385085.97**	0.59	9.8	**381516.50**	0.28	16.2
200-10-2a	462234.10	0.88	10.7	**454203.83**	0.43	10.7	**453625.67**	0.28	11.6	**452374.37**	0.20	16.7
200-10-2b	387271.63	0.87	9.3	**381417.83**	0.47	8.8	**380320.33**	0.44	9.6	**376836.03**	0.27	15.5
200-10-3a	481857.00	0.58	10.4	**477278.80**	0.40	11.0	**477010.47**	0.38	13.2	**475344.83**	0.32	18.3
200-10-3b	377792.13	0.70	7.3	**372415.27**	0.69	6.7	**370046.37**	0.81	7.9	**365705.17**	0.34	15.2
%-gap $_{\text{min, BKS}}$	0.65			0.28			0.18			0.01		
%-gap $_{\text{avg., BKS}}$	2.29			1.16			1.06			0.64		

the best solutions obtained in the 30 runs each as well as of the average solution values of these runs of the corresponding algorithm. Average results are marked bold if they yield a significant improvement over solely applying the VNS; verified with a Wilcoxon rank sum test with an error level of 5%. We will start by comparing our enhanced VNS methods to solely applying VNS, followed by a comparison to leading methods of the corresponding problem variant.

It can be observed that the VNS performs very well in general, but especially on the PLRP. Adding V_1 (VNS+V_1) induces only a slight increase in runtime, yet for the LRP the average gap is nearly halved, and for the PLRP the negative gap is further decreased by 0.25%. Additionally applying V_2 (VNS+$V_{1,2}$) further improves the results for both variants, this time showing a greater impact on the PLRP, which was partly expected due to the incorporation of the periodic aspect. Not surprisingly the increase in runtime is also greater for the PLRP, and since the model's size depends on the number of depots, this is especially notable on the instances with $m = 10$. Finally, also using V_3 (VNS+$V_{1,2,3}$) increases the runtimes contrary to before, since its model's size depends on the number of customers (per day), which mainly concerns the LRP, where longer routes appear

Table 4. Results of VNS and VNS plus VLNS combinations on the PLRP.

Instance	VNS			VNS+V_1			VNS+$V_{1,2}$			VNS+$V_{1,2,3}$		
	avg.	sdv. [%]	t [s]	avg.	sdv. [%]	t [s]	avg.	sdv. [%]	t [s]	avg.	sdv. [%]	t [s]
20-5-1a	79454.10	0.79	1.6	79411.53	0.67	1.6	79361.47	0.75	1.9	79282.57	0.72	2.1
20-5-1b	78343.80	2.18	1.6	79629.37	2.85	1.6	78318.60	2.53	1.8	78105.87	2.52	2.1
20-5-2a	79804.87	1.24	1.6	79556.57	1.36	1.5	**79028.73**	1.40	1.8	**78970.33**	1.27	2.1
20-5-2b	63257.53	0.90	1.5	63238.87	0.74	1.5	63223.27	0.80	1.8	63096.60	0.80	2.0
50-5-1a	152972.60	1.74	3.7	152477.50	1.58	3.9	152329.27	1.77	4.5	152177.43	1.00	5.0
50-5-1b	142449.97	1.52	3.8	142229.13	1.48	4.1	141794.13	1.64	4.8	**141262.33**	1.69	5.2
50-5-2a	143731.17	0.78	3.6	143547.83	1.37	3.6	**142796.73**	0.94	4.4	**142491.27**	0.92	4.8
50-5-2b	116350.80	1.09	3.4	116245.43	1.60	3.6	**115572.63**	1.67	4.2	114849.53	1.92	4.6
50-5-2aBIS	175978.60	1.68	3.0	176993.17	1.88	3.0	174541.90	1.55	3.5	175577.97	1.96	3.9
50-5-2bBIS	106017.00	2.19	3.2	104976.10	1.93	3.3	**105021.07**	2.32	3.6	**105173.47**	2.11	4.0
50-5-3a	157683.73	1.30	3.6	157828.03	1.20	3.7	**156203.50**	1.06	4.3	**156515.37**	1.02	4.7
50-5-3b	112633.10	1.65	3.6	112255.33	1.60	3.9	112791.17	1.96	4.3	112162.07	1.67	4.7
100-5-1a	356720.93	0.99	6.2	355023.53	1.18	6.3	355883.87	1.02	7.3	355150.70	1.25	8.1
100-5-1b	238598.30	1.50	7.0	237310.50	1.45	7.4	238005.00	1.51	8.7	237438.70	1.78	9.5
100-5-2a	272547.67	0.82	6.0	**271416.67**	1.05	6.0	**269891.33**	1.14	7.6	271363.50	1.28	8.1
100-5-2b	169677.70	1.74	6.0	169351.50	1.52	5.7	169601.23	1.46	6.9	170040.70	2.03	7.4
100-5-3a	227452.33	1.09	6.1	227125.63	1.15	6.3	**225138.40**	1.20	7.7	226444.73	1.45	8.2
100-5-3b	176998.30	1.71	7.5	176197.67	1.38	7.9	176313.03	1.48	9.1	**176004.53**	1.61	9.9
100-10-1a	268807.80	0.97	10.7	**265693.07**	1.19	12.3	**263990.00**	0.87	18.7	**265438.37**	0.94	19.9
100-10-1b	217358.80	1.47	9.8	216326.27	1.47	11.3	216803.00	1.46	17.5	217084.20	1.27	18.9
100-10-2a	267882.93	1.33	7.8	267811.00	1.27	8.9	267365.00	1.09	13.4	268575.77	1.22	13.6
100-10-2b	175829.90	1.58	8.0	175681.90	1.81	8.4	175196.30	1.66	11.1	176078.47	1.70	11.7
100-10-3a	265910.37	1.32	8.4	**263870.90**	1.34	9.9	**264044.60**	1.13	15.3	264714.53	1.47	15.0
100-10-3b	199922.63	1.67	9.3	198927.20	1.08	10.2	**198500.53**	1.64	13.8	199679.60	1.55	14.4
200-10-1a	453541.50	0.80	18.5	**451017.47**	1.02	20.0	**448584.57**	1.22	27.6	452401.77	0.94	29.6
200-10-1b	384547.63	1.07	20.0	384567.47	1.19	22.0	383839.33	0.89	29.9	386224.97	1.37	32.4
200-10-2a	397358.37	0.90	15.2	**393325.00**	0.98	15.5	**391387.60**	1.20	20.3	**393316.83**	0.75	22.3
200-10-2b	328169.77	1.27	16.3	327180.13	1.29	16.1	**325184.10**	1.17	20.8	329007.53	1.80	24.4
200-10-3a	557255.00	1.00	14.5	554921.90	1.03	16.5	555822.73	1.15	21.9	560297.17	1.00	25.0
200-10-3b	362268.50	1.47	16.5	361395.50	1.19	16.5	361268.60	1.38	22.0	365044.00	1.27	23.4
%-gap $_{\text{min, BKS}}$	-4.35			-4.68			-4.92			-4.58		
%-gap $_{\text{avg., BKS}}$	-1.99			-2.24			-2.54			-2.33		

in general, too. However, perhaps due to this, and also the single day planning horizon, an overall performance gain with V_3 only occurs for the LRP, yielding the best solutions on average. In case of the PLRP it even reduces the overall solution quality. Taking a closer look shows that the performance is more or less the same up to instances having 50 customers, whereas afterwards it deteriorates and somehow seems to work against the optimization process. Probably it leads to getting stuck in unfavorable local optima. However, the bad performance of V_3 for the PLRP is most likely also related to the structure of the instances considered, since they rather promote many small routes, offering less potential for more sophisticated intra-route improvement. Anyway, this fact needs to be investigated further, probably applying tests on altered or newly created instances. We remark that solely applying V_2 would often lead to a considerably larger increase in runtime, especially for the LRP this time. So it seems that V_1 achieves important preliminary work. Otherwise we deem the increase of runtime only moderate and hence omitted to state additional VNS runs with more iterations.

The best found solutions of our algorithm variants over all runs performed are compared to the so far best known solutions in Table 5, stating the individual

Table 5. Comparison of previous best known solutions (BKS) and best results obtained by our algorithm over all runs for the LRP and PLRP.

Instance	LRP			PLRP		
	BKS	best	%-gap	BKS	best	%-gap
20-5-1a	54793	54793	0.00	78851	**78477**	-0.47
20-5-1b	39104	39104	0.00	**75727**	76102	0.50
20-5-2a	48908	48908	0.00	80538	**77784**	-3.42
20-5-2b	37542	37542	0.00	62987	**62133**	-1.36
50-5-1a	90111	90111	0.00	157632	**146360**	-7.15
50-5-1b	63242	63242	0.00	142359	**136454**	-4.15
50-5-2a	88298	88298	0.00	152376	**138391**	-9.18
50-5-2b	67308	67308	0.00	119925	**110043**	-8.24
50-5-2aBIS	84055	84055	0.00	177649	**168721**	-5.03
50-5-2bBIS	51822	51822	0.00	102824	**100836**	-1.93
50-5-3a	86203	86203	0.00	162640	**152863**	-6.01
50-5-3b	61830	61830	0.00	114931	**108790**	-5.34
100-5-1a	275993	**275813**	-0.07	364324	**341083**	-6.38
100-5-1b	214392	**213973**	-0.20	239235	**224307**	-6.24
100-5-2a	194267	**193671**	-0.31	287093	**260182**	-9.37
100-5-2b	157173	**157157**	-0.01	174963	**163191**	-6.73
100-5-3a	200246	**200160**	-0.04	238031	**214341**	-9.95
100-5-3b	152528	**152466**	-0.04	181334	**166726**	-8.06
100-10-1a	290429	**288540**	-0.65	281896	**256892**	-8.87
100-10-1b	234641	**232230**	-1.03	224751	**207326**	-7.75
100-10-2a	243778	**243677**	-0.04	282437	**257388**	-8.87
100-10-2b	203988	203988	0.00	179047	**166810**	-6.83
100-10-3a	253344	**251128**	-0.87	272463	**253617**	-6.92
100-10-3b	**204597**	204706	0.05	208890	**188678**	-9.68
200-10-1a	479425	**478349**	-0.22	464596	**431373**	-7.15
200-10-1b	378773	**378631**	-0.04	401901	**362143**	-9.89
200-10-2a	450468	**449571**	-0.20	410514	**380262**	-7.37
200-10-2b	**374435**	375129	0.19	324121	**310110**	-4.32
200-10-3a	472898	**471024**	-0.40	574398	**530195**	-7.70
200-10-3b	364178	**363907**	-0.07	356618	**342944**	-3.83
average			-0.13			-6.26

Table 6. Showing average relative runtimes of LRP solution approaches and average percentage gaps to (previously) best known solutions.

n	m	GRASP [2]	MAPM [3]	LRGTS [6]	GRASP+ELS [5]	VNS	VNS+V$_{1,2,3}$
50	5	1.00	1.00	1.00	1.00	1.00	1.00
100	5	10.87	9.10	5.11	29.76	1.45	1.79
100	10	17.33	8.46	16.00	32.52	1.95	2.66
200	10	207.79	87.88	73.11	188.26	4.07	6.05
%-gap $_{1\times\text{run, BKS}}$		3.37	1.15	0.52			
%-gap $_{\text{avg, BKS}}$						2.29	0.64
%-gap $_{\text{min, BKS}}$					0.84	0.65	0.01

and the average percentage gaps between them. The encouraging results clearly show the potential of our method(s) for this problem variants.

A further interesting comparison is made with regard to the dependence of runtime on instance size for previous, leading (P)LRP approaches and our

Table 7. Showing average relative runtimes of PLRP solution approaches and average percentage gaps to (previously) best known solutions.

n	m	IM[8]	MAPM[9]	ELS[10]	VNS	VNS+$V_{1,2}$
50	5	1.00	1.00	1.00	1.00	1.00
100	5	8.01	11.26	5.50	1.85	1.88
100	10	17.97	13.45	10.15	2.58	3.56
200	10	145.14	250.78	60.95	4.82	5.65
%-gap $_{1\times\text{run, BKS}}$		12.32	9.37			
%-gap $_{\text{avg., BKS}}$				1.45	-1.99	-2.54

VNS as well as the best performing VNS plus VLNS variant; these results are shown in Table 6 for the LRP and in Table 7 for the PLRP. Here each algorithm's runtime is normalized by its average runtime on instances of type $n = 50$, $m = 5$, yielding a better picture as using the smallest instances with $n = 20$, $m = 5$ for it. Looking at the moderate increase of relative runtime our methods also seem very promising with respect to tackling even larger instances. Comparing the absolute runtimes with those of several LRP approaches presented in [5] using a similar computing environment (or giving scaled runtimes) is also clearly in favor of our methods. For completeness we also state in Tables 6 and 7 the overall average percentage gaps to the (previously) best known solutions as given in Table 5. Due to different practices of performing computational tests and/or presenting results we are not able to compare the different methods in a more direct way. Hence we denote the results where only a single run was executed with %-gap $_{1\times\text{run, BKS}}$ (those methods were claimed to be robust), and those of the average as well as best solution quality as before. We note that for GRASP+ELS[5] and ELS[10] five runs were performed. Anyway, it is obvious that our methods are at least competitive to leading LRP approaches, yielding 15 new best solutions for 30 instances, and clearly outperform previous solution approaches for the PLRP, yielding new best solutions for 29 out of 30 instances.

6 Conclusions

We presented a variable neighborhood search (VNS) for the periodic location-routing problem (PLRP) with capacitated vehicles and depots, which is also directly applicable to the LRP, i.e. the special case when having only a single day planning horizon The VNS is subsequently combined with integer linear programming-based very large neighborhood searches (VLNS). Two of them operate on a higher level via relocating whole routes to depots, considering all days at once, with one being a more sophisticated version using a set covering model. The third one deals with the location of customer sequences to insertion points in routes of a single day. Two of the VLNS are further designed to exploit the information contained in several solutions provided by the VNS. Experimental results on available benchmark test data show the excellent performance of our methods on the LRP and the PLRP when compared to corresponding leading approaches, both in terms of solution quality and dependence of runtime on instance size. The results further indicate almost always a gain in solution quality when gradually applying all VLNS, very often yielding significantly better results as applying VNS alone.

Next, we want to incorporate the periodicity in a suitably enhanced version of the third VLNS, as well as to especially test also on other available LRP data sets as well as on new or modified PLRP instances. Since we used only a moderate amount of iterations/runtime it would be interesting to allow more resources and investigate the behavior of our methods.

Acknowledgments

This work has been supported by the Austrian Science Fund (FWF) under contract number P20342-N13. The authors also want to thank the anonymous reviewers for their valuable comments.

References

1. Salhi, S., Rand, G.K.: The effect of ignoring routes when locating depots. European Journal of Operational Research **39**(2) (1989) 150 – 156
2. Prins, C., Prodhon, C., Calvo, R.W.: Solving the capacitated location-routing problem by a GRASP complemented by a learning process and a path relinking. 4OR **4**(3) (2006) 221–238
3. Prins, C., Prodhon, C., Calvo, R.W.: A memetic algorithm with population management (MA|PM) for the capacitated location-routing problem. In Gottlieb, J., Raidl, G.R., eds.: Evolutionary Computation in Combinatorial Optimization – EvoCOP 2006. Volume 3906 of LNCS. Springer (2006) 183–194
4. Duhamel, C., Lacomme, P., Prins, C., Prodhon, C.: A memetic approach for the capacitated location routing problem. In Prodhon, C., et al., eds.: Proceedings of the 9th EU/MEeting on Metaheuristics for Logistics and Vehicle Routing, Troyes, France (2008)
5. Duhamel, C., Lacomme, P., Prins, C., Prodhon, C.: A GRASP×ELS approach for the capacitated location-routing problem. Computers & OR **37**(11) (2010) 1912–1923
6. Prins, C., Prodhon, C., Ruiz, A., Sorianoa, P., Calvo, R.W.: Solving the capacitated location-routing problem by a cooperative Lagrangean relaxation-granular tabu search heuristic. Transportation Science **41**(4) (2007) 470–483
7. Nagy, G., Salhi, S.: Location-routing: Issues, models and methods. European Journal of Operational Research **177**(2) (2007) 649–672
8. Prodhon, C.: A metaheuristic for the periodic location-routing problem. In Kalcsics, J., Nickel, S., eds.: Operations Research Proceedings 2007, Springer (2007) 159–164
9. Prodhon, C., Prins, C.: A memetic algorithm with population management (MA|PM) for the periodic location-routing problem. In Blesa, M.J., Blum, C., Cotta, C., Fernández, A.J., Gallardo, J.E., Roli, A., Sampels, M., eds.: Hybrid Metaheuristics. Volume 5296 of LNCS., Springer (2008) 43–57

10. Prodhon, C.: An ELS×path relinking hybrid for the periodic location-routing problem. In Blesa, M.J., Blum, C., Gaspero, L.D., Roli, A., Sampels, M., Schaerf, A., eds.: Hybrid Metaheuristics. Volume 5818 of LNCS., Springer (2009) 15–29
11. 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
12. 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)
13. 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
14. De Franceschi, R., Fischetti, M., Toth, P.: A new ILP-based refinement heuristic for vehicle routing problems. Math. Program **105**(2-3) (2006) 471–499
15. 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., Blum, C., Gaspero, L.D., Roli, A., Sampels, M., Schaerf, A., eds.: Hybrid Metaheuristics. Volume 5818 of LNCS., Springer (2009) 45–59
16. Raidl, G.R., Puchinger, J.: Combining (integer) linear programming techniques and metaheuristics for combinatorial optimization. In Blum, C., et al., eds.: Hybrid Metaheuristics – An Emergent Approach for Combinatorial Optimization. Volume 114 of SCI. Springer (2008) 31–62
17. Hansen, P., Mladenović, N.: Variable neighborhood search. In Glover, F., Kochenberger, G., eds.: Handbook of Metaheuristics. Kluwer Academic Publishers, Boston MA (2003) 145–184
18. Clarke, G., Wright, J.W.: Scheduling of vehicles from a central depot to a number of delivery points. Operations Research **12**(4) (1964) 568–581
19. 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
20. Kirkpatrick, S., Gelatt Jr., C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220**(4598) (1983) 671–680
21. Ahuja, R.K., Ergun, Ö., Orlin, J.B., Punnen, A.P.: A survey of very large-scale neighborhood search techniques. Discrete Applied Mathematics **123**(1-3) (2002) 75–102
22. Akca, Z., Berger, R.T., Ralphs, T.K.: A branch-and-price algorithm for combined location and routing problems under capacity restrictions. In Chinneck, J.W., et al., eds.: Operations Research and Cyber-Infrastructure. Volume 47 of Operations Research/Computer Science Interfaces Series. Springer (2009) 309–330
23. Desrosiers, J., Lübbecke, M.E.: A primer in column generation. In Desaulniers, G., et al., eds.: Column Generation. Springer (2005) 1–32
24. Rousseau, L.M., Gendreau, M., Pesant, G.: Using constraint-based operators to solve the vehicle routing problem with time windows. Journal of Heuristics **8**(1) (2002) 43–58
25. Prodhon, C.: `http://prodhonc.free.fr/` (June 2010)