Available online at www.sciencedirect.com
ScienceDirect
Computers & Operations Research 34 (2007) 2403–2435
computers & operations research
www.elsevier.com/locate/cor

A general heuristic for vehicle routing problems

David Pisinger*, Stefan Ropke

DIKU, Department of Computer Science, University of Copenhagen, Universitetsparken 1, DK-2100 Copenhagen Ø, Denmark

Available online 24 October 2005

Abstract

We present a unified heuristic which is able to solve five different variants of the vehicle routing problem: the vehicle routing problem with time windows (VRPTW), the capacitated vehicle routing problem (CVRP), the multi-depot vehicle routing problem (MDVRP), the site-dependent vehicle routing problem (SDVRP) and the open vehicle routing problem (OVRP).

All problem variants are transformed into a rich pickup and delivery model and solved using the adaptive large neighborhood search (ALNS) framework presented in Ropke and Pisinger [An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation Science, to appear]. The ALNS framework is an extension of the large neighborhood search framework by Shaw [Using constraint programming and local search methods to solve vehicle routing problems. In: CP-98, Fourth international conference on principles and practice of constraint programming, Lecture notes in computer science, vol. 1520, 1998. p. 417–31] with an adaptive layer. This layer adaptively chooses among a number of insertion and removal heuristics to intensify and diversify the search. The presented approach has a number of advantages: it provides solutions of very high quality, the algorithm is robust, and to some extent self-calibrating. Moreover, the unified model allows the dispatcher to mix various variants of VRP problems for individual customers or vehicles.

As we believe that the ALNS framework can be applied to a large number of tightly constrained optimization problems, a general description of the framework is given, and it is discussed how the various components can be designed in a particular setting.

The paper is concluded with a computational study, in which the five different variants of the vehicle routing problem are considered on standard benchmark tests from the literature. The outcome of the tests is promising as the algorithm is able to improve 183 best known solutions out of 486 benchmark tests. The heuristic has also shown promising results for a large class of vehicle routing problems with backhauls as demonstrated in Ropke and Pisinger [A unified heuristic for a large class of vehicle routing problems with backhauls. European Journal of Operational Research, 2004, to appear].
© 2005 Elsevier Ltd. All rights reserved.

Keywords: Metaheuristics; Large neighborhood search; Vehicle routing problem

1. Introduction

Most scientific papers in the area of heuristic solution methods for vehicle routing problems target a specific vehicle routing problem, e.g. vehicle routing problems with time windows (VRPTW). In such papers a heuristic is designed, implemented and fine-tuned to fit this particular problem type. Only a few papers (see e.g. [1,2]) consider heuristics that "out-of-the-box" can be used to solve several problem types. We believe that general vehicle routing heuristics are

* Corresponding author. Tel.: +45 35 32 1354; fax: +45 35 32 1401.
E-mail addresses: pisinger@diku.dk (D. Pisinger), sropke@diku.dk (S. Ropke).

0305-0548/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2005.09.012

an important research area as such heuristics are needed for real life problems, in which the transportation needs of different companies often are different and thus call for various types of vehicle routing problems.

The heuristic in this paper is applied to five different problems: the VRPTW, the capacitated vehicle routing problem (CVRP), the multi-depot vehicle routing problem (MDVRP), the site-dependent vehicle routing problem (SDVRP) and the open vehicle routing problem (OVRP). In the CVRP one has to deliver goods to a set of customers with known demands on minimum-cost vehicle routes originating and terminating at a depot. The vehicles are assumed to be homogeneous and having a certain capacity. In some versions of the CVRP one also has to obey a route duration constraint that limits the lengths of the feasible routes. The VRPTW extends the CVRP by associating time windows with the customers. The time window defines an interval during which the customer must be visited. The OVRP is closely related to the CVRP, but contrary to the CVRP a route ends as soon as the last customer has been served as the vehicles do not need to return to the depot. The MDVRP extends the CVRP by allowing multiple depots. The SDVRP is another generalization of the CVRP in which one can specify that certain customers only can be served by a subset of the vehicles. Furthermore, vehicles do not need to have the same capacity in the SDVRP. In the CVRP, MDVRP and SDVRP one seeks to minimize the total traveled distance, whereas in the OVRP and VRPTW, the first priority is to minimize the number of vehicles and minimizing the traveled distance is the second priority. The choice of objective is not an intrinsic feature of the problems, but just the tradition in the metaheuristic literature. Most exact methods and some metaheuristics for the VRPTW minimize total traveled distance instead of minimizing number of vehicles used.

All problem types are transformed to a rich pickup and delivery problem with time windows (RPDPTW) and are solved using the adaptive large neighborhood search (ALNS) framework introduced by [3,4]. The heuristic presented in the two aforementioned papers has been reused, with some small improvements (summarized in Section 5), to solve the five problem types considered in this paper.

In the RPDPTW, we have a number of requests to be carried out by a given set of vehicles. Each request consists of picking up a quantity of goods at one location and delivering it to another location. The objective of the problem is to find a feasible set of routes for the vehicles so that all requests are serviced, and such that the overall travel distance is minimized. A feasible route of a vehicle must start at a given location, service a number of requests such that the capacity of the vehicle is not exceeded, and finally end at a given location. A pickup or delivery must take place within a given time window. Each request has an associated pickup precedence number, and a delivery precedence number. A vehicle must visit the locations in nondecreasing order of precedences (see e.g. [5] for various applications of precedence constraints). Since not all vehicles may be able to service all requests (e.g. due to their physical size or the absence of some cooling compartments) we need to ensure that every request is serviced by a given subset of vehicles. Between any two locations we have an associated, nonnegative distance and travel time. It is assumed that travel times satisfy the *triangle inequality*. This assumption implies that any removal of requests from a feasible route will keep the route feasible with respect to the imposed time windows.

The five vehicle routing problems considered in the present paper have all been intensively studied in the literature. The two best known problems are the VRPTW and the CVRP. The VRPTW has been the target of extensive research and almost any type of metaheuristic has been applied to the problem. For recent surveys on the state of the art in VRPTW research we recommend the survey by Cordeau et al. [6] that describes both exact and heuristic methods, and the survey by Bräysy and Gendreau [7] that focuses on metaheuristics. It is hard to single out a few VRPTW metaheuristics as the number of proposed heuristics is huge, and no heuristic dominates all the other heuristics in all aspects. We would, however, like to mention the metaheuristic by Mester and Bräysy [8] as it has achieved outstanding results on larger VRPTW instances with between 200 and 1000 customers. For the smaller VRPTW instances like the Solomon data set, some of the best heuristics in terms of solution quality achieved are the large neighborhood search by Bent and Van Hentenryck [9] and the hybrid genetic algorithm by Homberger and Gehring [10].

Solving the VRPTW to optimality has also received much attention. The current state of the art exact methods are proposed by Kallehauge et al. [11], Irnich and Villeneuve [12] and Chabrier [13], and all follow the branch-and-price framework. The two first mentioned approaches also strengthen the obtained lower bound by adding valid inequalities to the LP formulation. The size of the instances that consistently can be solved to optimality is rather limited as unsolved instances with 50 customers exist, but some large-scale instances can be solved. For example, Kallehauge et al. [11] report that a 1000 customer instance has been solved. Solving problems of this size is only possible by current techniques if the instance has a certain structure and the time constraints are very tight. These observations justify the research into heuristics for the VRPTW as industrial routing problems demand robust algorithms for large-sized instances.

The CVRP literature is also vast. Classic heuristics for the problem have been surveyed by Laporte and Semet [14], and metaheuristics have been surveyed by Gendreau et al. [15] and more recently by Cordeau et al. [16]. CVRP heuristics have typically been tested on 14 instances containing between 50 and 199 customers. In the early 1990s very good metaheuristics for the CVRP were developed such as parallel tabu search by Taillard [17]. Most of the solutions to the 14 classic instances found back then have still not been improved. More recently, some larger instances have been introduced containing between 240 and 1200 customers [18,19]. These new instances seem to have spurred a new interest into metaheuristics for the CVRP as indicated in the survey by Cordeau et al. [16].

Until recently, exact methods for the CVRP were dominated by branch-and-cut methods. One of the best branch-and-cut algorithms for the CVRP was developed by Lysgaard et al. [20]. Recent research results indicate that branch-and-cut-and-price algorithms are a more promising approach as shown by Fukasawa et al. [21]. For the CVRP, the largest problem that has been solved to optimality contains 135 customers.

The OVRP is a variant of the CVRP that has received less attention. The problem appears in various distribution problems, in which the vehicle simply stops after the last delivery. The problem was introduced by Sariklis and Powell [22] and they proposed a two-phase cluster first-route second heuristic. Recently, tabu search heuristics were proposed by Fu et al. [23] and Brandão [24].

Tabu search heuristics for the MDVRP have been proposed by Renaud et al. [25] and Cordeau et al. [1]. The last paper deserves special attention as it describes a general heuristic that also solves periodic vehicle routing problems (PVRP) and periodic traveling salesman problems. Earlier, Chao et al. [26] proposed a *record-to-record* improvement heuristic for the MDVRP.

The SDVRP was first studied by Nag et al. [27] who developed several simple heuristics for the problem. Chao et al. [28] developed a more advanced heuristic and constructed several new test instances. Cordeau and Laporte [29] showed that the problem could be seen as a special case of the PVRP and they presented computational results obtained by solving the problem using their PVRP tabu search heuristic.

The main contribution of this paper is to describe a general ALNS heuristic, that is able to solve all the above variants of the VRP problem. The computational results are promising as the ALNS, for the large-scale VRPTW instances suggested by Gehring and Homberger [30], on average use less vehicles compared to competing heuristics, and the method becomes even more attractive compared to other heuristics as the problem size increases. For the OVRP, MDVRP and SDVRP we are able to improve a large number of best known solutions. The ALNS heuristic is comparable to most recently proposed heuristics for the CVRP, but it is surpassed by the very best heuristic for the problem type.

Due to the promising results of ALNS, we give a general description of the paradigm to make it easier to adapt the framework to other problem types. Various strategies for designing construction and removal heuristics are discussed.

In Section 2, we give a formal mathematical definition of the RPDPTW and in Section 3, we describe how the considered problem variants are transformed into the RPDPTW. In Section 4, we give a general presentation of the ALNS algorithm forming the core of our solution approach. Section 5 describes how the general framework has been adapted to solve the RPDPTW. Section 6 presents a number of computational experiments which document that the proposed heuristic does not perform worse than state-of-the-art heuristics specialized to solve each problem variant. The paper is concluded in Section 7.

2. Formal problem definition

We now present a mathematical formulation of the RPDPTW problem. The mathematical model is used to describe the heuristic in details in later sections and to describe how the considered VRP variants are transformed to the RPDPTW.

Following the terminology of Desaulniers et al. [31], a problem instance of the pickup and delivery problem contains n requests and m vehicles. The problem is defined on a graph where $P = \{1, \ldots, n\}$ is the set of pickup nodes, and $D = \{n+1, \ldots, 2n\}$ is the set of delivery nodes. Request i is represented by node i and $i+n$. $K = \{1, \ldots, m\}$ is the set of all vehicles. Let $P_k \subseteq P$ and $D_k \subseteq D$ be the set of pickups and deliveries that can be served by vehicle k. Since a request is serviced by the same vehicle we may assume that $i \in P_k \Leftrightarrow i+n \in D_k$, i.e. that both the pickup and delivery can be serviced by vehicle k. Define $N = P \cup D$ and $N_k = P_k \cup D_k$. Let $\tau_k = 2n+k,\ k \in K$ and $\tau'_k = 2n+m+k,\ k \in K$ be the nodes that represent the start and end terminals of vehicle k. The directed graph $G = (V, A)$ consists of the nodes $V = N \cup \{\tau_1, \ldots, \tau_m\} \cup \{\tau'_1, \ldots, \tau'_m\}$ and the arcs $A = V \times V$. For each vehicle we have a sub-graph $G_k = (V_k, A_k)$,

where $V_k = N_k \cup \{\tau_k\} \cup \{\tau'_k\}$ and $A_k = V_k \times V_k$. For each edge $(i, j) \in A$ we assign a distance $d_{ij} \geqslant 0$ and a travel time $t_{ij} \geqslant 0$. Again, it is assumed that the travel times satisfy the *triangle inequality*, i.e. $t_{ij} \leqslant t_{il} + t_{lj}$ for all $i, j, l \in V$. We assign a service time s_i and a time window $[a_i, b_i]$ to each node $i \in V$. The service time represents the time needed for loading and unloading and the time window indicates when the visit at the particular site must start; a visit to node i can only take place between time a_i and b_i. A vehicle is allowed to arrive to a site before the start of the time window but it has to wait until the start of the time window before the visit can be performed. For each node $i \in N$, we define l_i to be the amount of goods that should be loaded onto the vehicle at the particular node. We have that $l_i \geqslant 0$ for $i \in P$ and $l_i = -l_{i-n}$ for $i \in D$. Each vehicle $k \in K$ has a certain capacity C_k. Each node has assigned a *precedence number* Π_i. Nodes with low precedence must always be visited before nodes with higher precedence.

Each vehicle k should follow a legal route from its start terminal τ_k to its destination terminal τ'_k. A legal route $\bar{r}$ is a simple (loop-free) path

$$\bar{r} = (\tau_k = v_1, v_2, \ldots, v_h = \tau'_k) \tag{1}$$

satisfying the precedences and time windows at the customers, the capacity of the vehicle, and ensuring that a pickup takes place before a delivery, and that only requests serviceable by vehicle k are carried out.

More formally, we demand that a vehicle only visits nodes that can be serviced by the vehicle, i.e.

$$v_i \in N_k, \quad i = 2, \ldots, h-1. \tag{2}$$

A pickup–delivery pair must be served by the same vehicle, and the pickup must take place before the delivery, hence we have

$$i \leqslant j, \quad v_i \in P_k, \quad v_j \in D_k, \quad v_j = v_i + n. \tag{3}$$

Precedences should be obeyed along the route, this is ensured by the constraints

$$i \leqslant j, \quad \Pi_{v_i} \leqslant \Pi_{v_j}. \tag{4}$$

To ensure that time windows are satisfied, we introduce $S_i \in \mathbb{R}_0^+$ to denote when the vehicle starts the service at site v_i. We then have the constraints

$$a_{v_i} \leqslant S_i \leqslant b_{v_i}, \quad i = 1, \ldots, h, \tag{5}$$
$$S_{i+1} \geqslant S_i + s_i + t_{v_i, v_{i+1}}, \quad i = 1, \ldots, h-1, \tag{6}$$
$$a_{\tau_k} \leqslant S_1 \leqslant b_{\tau_k}, \tag{7}$$
$$a_{\tau'_k} \leqslant S_h \leqslant b_{\tau'_k}, \tag{8}$$

where $[a_{\tau_k}, b_{\tau_k}]$ is the time window of terminal τ_k and $[a_{\tau'_k}, b_{\tau'_k}]$ is the time window of terminal τ'_k. Finally, the capacity of the vehicle should be respected throughout the path. For this purpose, we introduce $L_i \in \mathbb{R}_0^+$ to denote the load of the vehicle at node i after serving node i. Then we have

$$L_i \leqslant C_k, \quad i = 1, \ldots, h, \tag{9}$$
$$L_{i+1} = L_i + l_{i+1}, \quad i = 1, \ldots, h-1, \tag{10}$$
$$L_1 = 0, \tag{11}$$
$$L_h = 0. \tag{12}$$

The *travel cost* of a given route $\bar{r}$ is

$$c_{\bar{r}} = \sum_{i=1}^{h-1} d_{v_i, v_{i+1}}. \tag{13}$$

Situations may occur in which some requests cannot be serviced by the available vehicles. To model this situation we create n dummy routes, consisting of a single request. These routes do not make use of any vehicles but they have a large cost, denoted Γ. Requests that are not served by a vehicle are said to be located in the *request bank*.

The whole problem can now be formulated as follows: let R be the set of all feasible routes. The boolean matrix $(\alpha_{j\bar{r}})$ for $\bar{r} \in R$ and $j = 1, \ldots, n$ is used to indicate whether request j is serviced using route $\bar{r}$. The boolean matrix $(\beta_{k\bar{r}})$ for

$\bar{r} \in R$ and $k = 1, \ldots, m$ is used to indicate whether the route $\bar{r}$ is carried out by vehicle k. Using binary variables $x_{\bar{r}}$ to indicate whether route $\bar{r}$ is used in the solution we get the following model:

$$\min \quad f(x) = \sum_{\bar{r} \in R} c_{\bar{r}} x_{\bar{r}}, \tag{14}$$

$$\text{s.t.} \quad \sum_{\bar{r} \in R} \alpha_{j\bar{r}} x_{\bar{r}} = 1, \quad j = 1, \ldots, n, \tag{15}$$

$$\sum_{\bar{r} \in R} \beta_{k\bar{r}} x_{\bar{r}} = 1, \quad k = 1, \ldots, m, \tag{16}$$

$$x_{\bar{r}} \in \{0, 1\}, \quad \bar{r} \in R. \tag{17}$$

Note that a dummy route is not assigned to any vehicle, that is, for any dummy route $\bar{r}$ we have that $\beta_{k\bar{r}}=0$, $\forall k=1, \ldots, m$.

3. Problem transformations

The heuristic in this paper is applied to five different problems—VRPTW, CVRP, OVRP, MDVRP, SDVRP—which all are transformed to a RPDPTW. The conversions which will be described in the following paragraphs are extensions of the transformations presented by Ropke and Pisinger [4] for solving VRP problems with backhauls.

3.1. Vehicle routing problem with time windows

In order to transform a VRPTW instance to a RPDPTW instance we map every customer in the VRPTW to a request in the RPDPTW. Such a request consists of a pickup at the depot and a delivery at the customer site. The amount of goods that should be carried by the requests is equal to the demand of the corresponding customer. The time window of the pickup is set to $[a_d, a_d]$, where a_d is the start of the time window of the depot in the VRPTW and its service time is set to zero. The time window and service time of the delivery are copied from the corresponding customer in the VRPTW. In order to avoid routes that return to the depot for restocking we let all pickups and deliveries have precedence zero and one, respectively. All vehicles in the RPDPTW have the same start and end terminals corresponding to the depot in the VRPTW. Distances and travel times in the RPDPTW are set in the natural way.

3.2. Capacitated vehicle routing problem

A CVRP instance can easily be transformed to a VRPTW instance. This can, for example, be done by setting all travel and service times to zero and all time windows to [0,0]. If the CVRP contains a route duration constraint then travel times and durations should be set as in the CVRP. All time windows (including the ones at the end terminals) should be set to $[0, D]$, where D is the route duration. The VRPTW is transformed to a RPDPTW as described in Section 3.1.

3.3. Site-dependent vehicle routing problem

In the SDVRP a customer may only be serviced by a given subset of the vehicles, typically because the access paths to the node do not allow given vehicles to pass, or because specific facilities are demanded in the vehicle (e.g. a freezing compartment).

The SDVRP is easily modeled as a RPDPTW by using the transformation from CVRP to RPDPTW and noting that the RPDPTW allows us to specify the pickups P_k and deliveries D_k that can be carried out by vehicle k.

3.4. Open vehicle routing problem

The OVRP is very close to the CVRP. The difference between the two problems is that in the OVRP the vehicles do not have to return to the depot. Thus, an OVRP can be solved as an asymmetric CVRP by setting distances and travel times from every customer to the depot to zero.

The travel times in the resulting RPDPTW do not satisfy the triangle inequality, but our method is able to handle the problems anyway since $t_{ij} \leqslant t_{il} + t_{lj}$ is only violated when l is an end terminal. Our only reason for assuming that the triangle inequality is satisfied for the travel times is that we have to avoid situations in which the removal of one or more requests causes the travel time to increase. As the node sequence $i \rightarrow l \rightarrow j$, where $l \in \{\tau'_1, \ldots, \tau'_m\}$ never occurs in a valid route this violation of the triangle inequality does not cause any problems.

3.5. Multi-depot vehicle routing problem

In the MDVRP each customer may be serviced by a vehicle originating at any of the available depots. Even though our underlying RPDPTW model supports multiple depots, it requires that each request is assigned to a specific depot. In general, this is a hard optimization problem of its own which needs to be handled together with the routing problem. Hence, we use the following transformation.

Create a dummy base location where all routes start and end and where all ordinary requests are picked up. Also create a dummy request for each vehicle k in the problem. The pickup and delivery locations of these requests are located at the depot of the corresponding vehicle. A dummy request has demand zero, it does not have any service time and it can be served at any time. The set N_k of each vehicle k contains all ordinary requests and the dummy request corresponding to the vehicle. In this way, we ensure that each vehicle will carry precisely one dummy request.

The precedences Π_i of a pickup and a delivery corresponding to an ordinary request are set to zero and two, respectively. The precedence of the pickup and delivery of the dummy requests are set to one and three, respectively. This ensures that all ordinary deliveries will be surrounded by the pickup and delivery of a dummy request. The distance and travel time between a pickup of an ordinary request and any other location is set to zero. All other distances and travel times are set as defined by the original MDVRP.

In a solution to the RPDPTW that serves all requests we know that each vehicle will begin at a start terminal located at the dummy base location, then perform a number of pickups and then go to the pickup of the dummy request. Next, the ordinary deliveries will be served and the vehicle will return to the delivery of the dummy request and then to the end terminal of the route. Before starting the pickup of the dummy request and after the delivery of it all travel times and distances will be zero. Furthermore, travel times and distances are accumulated correctly while carrying the dummy request.

While solving MDVRP problems the cost of dummy routes Γ must be set to a sufficiently large number such that it will never be profitable to leave a dummy request in the request bank.

4. Adaptive large neighborhood search

We will now describe the ALNS framework used in the present paper. We believe that ALNS can be applied to a large class of difficult optimization problems, hence in the following we consider an optimization problem in the general IP form:

$$\min\{f(x) : Ax \leqslant b, x \in \mathbb{Z}^n\}. \tag{18}$$

ALNS is a local search framework in which a number of simple algorithms compete to modify the current solution. In each iteration an algorithm is chosen to destroy the current solution, and an algorithm is chosen to repair the solution. The new solution is accepted if it satisfies some criteria defined by the local search framework applied at the master level.

To be more formal, we extend the domain of each variable x_i to $\mathbb{Z} \cup \{\perp\}$, where $\perp$ means undefined. A *destroy heuristic* chooses at most q variables which are assigned the value $\perp$. A repair heuristic assigns feasible values $x_i \in \mathbb{Z}$ to the q variables.

The ALNS framework is an extension of the *large neighborhood search* presented by Shaw [32], where a large collection of variables are modified in each iteration. In ALNS, the neighborhoods are searched by simple and fast heuristics. ALNS is also based on the *Ruin and Recreate* paradigm presented by Schrimpf et al. [33], or the *Ripup and Reroute* paradigm applied in [34]. In each iteration the current solution is partially destroyed and then repaired using some heuristics. ALNS also has similarities with *very large scale neighborhood search* (VLSN) presented by

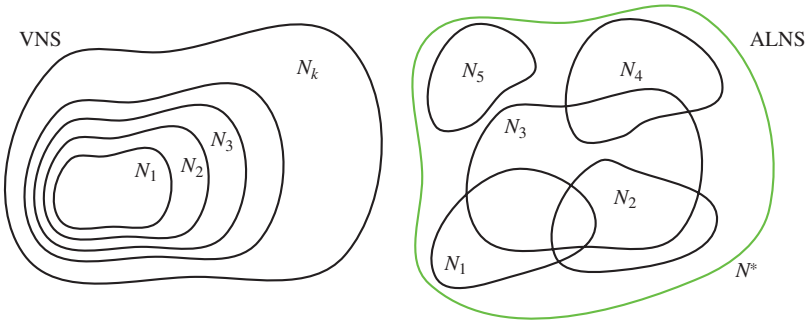

Fig. 1. Illustration of neighborhoods used by VNS and ALNS. VNS typically operates on one type of neighborhood with variable depth while ALNS operates on structurally different neighborhoods $N_1, \ldots, N_k$ defined by the corresponding search heuristics. All neighborhoods $N_1, \ldots, N_k$ in ALNS are a subset of the neighborhood N^* defined by modifying q variables.

Ahuja et al. [35]. In VLSN, the algorithm operates on very large neighborhoods chosen in a way so that they can still be searched efficiently.

Variable neighborhood search (VNS) was presented by Hansen and Mladenovic [36]. VNS makes use of a parameterized family of neighborhoods, typically obtained by using a given neighborhood with variable depth. When the algorithm reaches a local minimum using one of the neighborhoods, it proceeds with a larger neighborhood from the parameterized family. When the VNS algorithm gets out of the local minimum it proceeds with the smaller neighborhood. On the contrary, ALNS operates on a predefined set of large neighborhoods corresponding to the destroy (removal) and repair (insertion) heuristics. The neighborhoods are not necessarily well-defined in a formal mathematical sense—they are rather defined by the corresponding heuristic algorithm. The difference between VNS and ALNS is illustrated in Fig. 1. In the sections that follow, we will distinguish between a neighborhood and the heuristic searching it.

Instead of viewing the ALNS heuristic as a sequence of destroy and repair operations one can alternatively see it as a sequence of *fix* and *optimize* operations. The fix operation selects a number of variables that are fixed at their current value; the optimize operation seeks to find a near-optimal solution that respects the fixed variables, that is, only nonfixed variables can be changed. After the optimization operation, all variables are unlocked again. The fix operation is analogous to the destroy operation and the optimize operation is analogous to the repair operation. The fix/optimize view might be helpful when applying the heuristic to problems where the destroy and repair operations do not seem intuitive.

4.1. Outline of algorithm

ALNS can be based on any local search framework, e.g. simulated annealing, tabu search or guided local search. The general framework is outlined in Fig. 2, where lines 2–8 form the main loop of the local search framework at the master level. Implementing a simulated annealing algorithm is straightforward as one solution is sampled in each iteration of the ALNS. A simple tabu search could, for example, be implemented by randomly sampling a number of candidate solutions and choosing the best nontabu solution.

In each iteration of the main loop, we choose one destroy and one repair neighborhood (line 3). An adaptive layer stochastically controls which neighborhoods to choose according to their past performance (score). The more a neighborhood N_i has contributed to the solution process, the larger score π_i it obtains, and hence it has a larger probability of being chosen.

The adaptive layer uses roulette wheel selection for choosing a destroy and a repair neighborhood. If the past score of a neighborhood i is denoted π_i and we have ω neighborhoods, then we choose neighborhood N_j with probability

$$\frac{\pi_j}{\sum_{i=1}^{\omega} \pi_i}.$$

Note that the destroy and repair neighborhoods are selected independently, and hence two separate roulette wheel selections are performed.

Adaptive Large Neighborhood Search

1 Construct a feasible solution x; set $x^* := x$
2 Repeat
3 Choose a destroy neighborhood N^- and a repair neighborhood N^+ using roulette wheel selection based on previously obtained scores $\{\pi_j\}$
4 Generate a new solution x' from x using the heuristics corresponding to the chosen destroy and repair neighborhoods
5 If x' can be accepted then set $x := x'$
6 Update scores π_j of N^- and N^+
7 If $f(x) < f(x^*)$ set $x^* := x$
8 Until stop criteria is met
9 Return x^*

Fig. 2. Outline of the ALNS framework.

In most applications the neighborhoods are searched by fast heuristics, hence it is reasonable to assume that they are equally fast. But if some heuristics are significantly slower than others, one may normalize the score π_i of a neighborhood with a measure of the time consumption t_i of the corresponding heuristic. This ensures a proper trade-off between time consumption and solution quality.

In line 4 of the ALNS algorithm, we first destroy the current solution x using a heuristic searching the neighborhood N^- and then repair the solution using a heuristic corresponding to neighborhood N^+. It can be advantageous to use noising or randomization in the destroy and repair heuristics to obtain a proper diversification. In traditional local search heuristics the diversification is controlled implicitly by the local search paradigm (accept ratio, tabu list, etc.), but since we use large neighborhoods which are searched by simple heuristics, it is not sufficient to have a diversification operator at the master level. We also need a diversification operator at the sub-level to avoid stagnating search processes where the destroy and repair neighborhoods keep performing the same modifications to a solution.

Finally, in line 6 we update the scores π_i of the neighborhoods. A number of criteria can be used to measure how much a neighborhood contributes to the solution process: new best solutions are obviously given a large score, but also not previously visited solutions are given a score. Depending on the local search framework used on the master level, one may also give specific scores to accepted solutions e.g. in a simulated annealing framework. Since each step of the ALNS heuristic involves two neighborhoods (a destroy and a repair neighborhood), the score obtained in a given iteration is divided equally between them.

Every M iterations of the ALNS algorithm, the scores π_i are reset, and the probabilities for choosing the neighborhoods are recalculated. Each neighborhood is assigned a minimum probability for being chosen to ensure that statistical information about its performance can be collected. The probabilities for choosing a neighborhood can also be a weighted sum of the score during the last M iterations, and the overall score since the beginning of the algorithm.

4.2. Designing an ALNS algorithm

In order to design an ALNS algorithm for a given optimization problem one needs to

- Choose a number of fast construction heuristics which are able to construct a full solution given a partial solution (a solution where some variables are set to $\perp$ and some have a real value).
- Choose a number of destroy heuristics. It might be worthwhile to choose destroy heuristics that are expected to work well with the chosen construction heuristics, but it is not necessary.
- Choose a local search framework at the master level.

In each iteration the heuristic corresponding to a *destroy neighborhood* should remove a given number q of variables. The destroy neighborhoods (N^-) should be a proper mix of neighborhoods which can intensify and diversify the search.

To diversify the search, one may randomly choose q decision variables, i.e. using a *random removal* neighborhood. To intensify the search one may try to remove q "critical" variables, i.e. variables having a large cost or variables spoiling the current structure of the solution (e.g. edges crossing each other in a Euclidean traveling salesman problem). This is known as *worst removal* or *critical removal*. Concrete examples on *random removal* and *worst removal* neighborhoods in a VRP context are given in Sections 5.1.1–5.1.2.

One may also choose a number of related variables that are easy to interchange while maintaining feasibility of the solution. This *related removal* neighborhood was introduced by Shaw [32]. More formally, we can measure the relatedness r_{ij} of two variables x_i and x_j by the deviation of the corresponding coefficients in the constraint matrix A in problem (18). The smaller r_{ij} the more related are variables x_i and x_j. How exactly r_{ij} should be defined depends on the concrete problem at hand, and one may even have several simultaneous neighborhoods defined by various choices of the relatedness measure (r_{ij}). In order to choose the q most related variables, one needs to solve the NP-hard *dispersion-sum problem* given by

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} r_{ij}x_i x_j, \\ \text{subject to} \quad & \sum_{j=1}^{n} x_j = q, \\ & x_j \in \{0,1\}, \quad j = 1,\ldots,n. \end{aligned} \tag{19}$$

A greedy heuristic for this problem running in $O(n^3)$ was presented in [37] together with a more time-consuming exact algorithm. If n is large, it may be too time-consuming even to compute the whole matrix (r_{ij}) and one will instead choose related variables according to some heuristics. Shaw [32] presented an algorithm running in $O(qn)$ time by initially selecting a variable at random, and then repeatedly selecting an already selected variable i and finding a variable j which minimizes r_{ij} and adding j to the set of chosen variables. An alternative heuristic is based on a modified Kruskal's algorithm for the minimum spanning tree problem, using r_{ij} as edge weights, which stops when a connected component with q or more elements has been constructed. The variables in this component are set to $\perp$. The worst-case running time of this algorithm is $O(n^2 \log n)$ as we have n^2 edges in Kruskal's algorithm. Ropke and Pisinger [3] used a modified version of this algorithm in the VRP for splitting requests on a route into two strongly connected subsets. It should be noted that solving the dispersion sum problem (19) to optimality seldom would be a good idea even if it could be done in a very short time. If r_{ij} is independent of the current solution the destroy neighborhood obtained by solving the dispersion sum problem to optimality would always assign $\perp$ to the same set of variables. Concrete examples on various *related removal* neighborhoods are given in Sections 5.1.3–5.1.5.

Following the same idea as in *related removal* one may choose a number of variables having small coefficients in the resource constraints in (18), as these are generally easy to interchange and loosely speaking can fill up unused resource constraints. We denote this strategy *small removal*.

Finally, one may use *history-based removal* where the q variables are chosen according to some historical information as presented in [3]. The historical information could, for example, count how often setting a given variable (or set of variables) to a specific value leads to a bad solution. One may then try to remove variables that currently are assigned an improper value, based on the historical information. Variants of the *history-based removal* neighborhood are discussed in Sections 5.1.6–5.1.7.

Repair neighborhoods (N^+) are typically based on concrete well-performing heuristics for the given problem. These heuristics can make use of variants of the greedy paradigm, e.g. performing the locally best choice in each step, or performing the least bad choice in each step. An alternative variant of the greedy paradigm is to set all variables to their upper bound in problem (18), and repeatedly decrease the most expensive variable until a feasible solution is obtained. The repair heuristics can also be based on approximation algorithms or exact algorithms which have been relaxed to obtain faster solution times at the cost of solution quality. Shaw [32] and Bent and Van Hentenryck [9] proposed more expensive algorithms like searching N^+ based on relaxed branch-and-bound methods. Although ALNS mainly is intended to use cheap heuristics, more expensive search methods can be used if the scores of the corresponding neighborhoods are normalized with respect to the time consumption. In the context of VRP problems, repair neighborhoods are considered in more detail in Section 5.2 discussing both simple greedy approaches and variants of regret heuristics.

Some optimization problems can be split into a number of sub-problems, where each sub-problem can be solved individually. Such problems include the bin packing problem in which a number of bins are to be filled, or the vehicle routing problem in which a number of routes are to be constructed. For such problems, one should decide whether the sub-problems should be solved one by one (*sequential heuristics*) or all sub-problems should be solved at the same time (*parallel heuristics*). Sequential heuristics are easier to implement but may have the disadvantage that the last sub-problem solved is left with variables that do not fit well together. This is to some extent avoided in parallel heuristics.

A natural extension to the ALNS framework is to have *coupled neighborhoods*. In principle one may, for each destroy neighborhood N_i^-, define a subset $K_i \subseteq \{N^+\}$ of repair neighborhoods that can be used with N_i^-. The roulette wheel selection of repair neighborhoods will then only choose a neighborhood in K_i if N_i^- was chosen.

As a special case, one may have $K_i = \emptyset$ meaning that the neighborhood N_i^- takes care of both the destroy and repair steps. One could use an ordinary local search heuristic to compete with the other destroy and repair neighborhoods, ensuring that a thorough investigation of the solution space close to the current solution is made from time to time.

For some problems it may be sufficient to have a number of destroy and repair heuristics that are selected randomly with equal probability, that is without the adaptive layer. We will denote such a heuristic a *large multiple-neighborhood search* (LMNS). The LMNS heuristics share the robustness of the ALNS heuristics, while having considerably fewer parameters to calibrate.

4.3. Properties of the ALNS framework

The ALNS framework has several advantages. For most optimization problems we already know a number of well-performing heuristics which can form the core of an ALNS algorithm. Due to the large neighborhoods and diversity of the neighborhoods, the ALNS algorithm will explore large parts of the solution space in a structured way. The resulting algorithm becomes very robust, as it is able to adapt to various characteristics of the individual instances, and seldom is trapped in a local minima.

ALNS is particularly well suited for tightly constrained problems, in which small neighborhoods are not sufficient to escape a local minima or certain areas of the solution space. In such problems, the large neighborhood search makes it possible to change many variables each time to reach new feasible solutions.

The calibration of the ALNS algorithm is quite limited as the adaptive layer automatically adjusts the influence of each neighborhood used. It is still necessary to calibrate the individual sub-heuristics used for searching the destroy and repair neighborhoods, but one may calibrate these individually or even use the parameters used in existing algorithms.

In the design of most local search algorithms the researcher has to choose between a number of possible neighborhoods. In ALNS, the question is not "either-or" but rather "both-and". As a matter of fact, our experience is that the more (reasonable) neighborhoods the ALNS heuristic makes use of, the better it performs [3].

5. ALNS applied to the RPDPTW

We will now describe how the general ALNS framework has been adapted to the RPDPTW problem. The "variables" in the ALNS framework correspond to requests in the RPDPTW. A destroy neighborhood N^- consists of removing q requests from the existing routes and assigning them to the *request bank*.

The heuristic described in this section is almost identical to the heuristic used to solve a large class of vehicle routing problems with backhauls [4]. One more destroy heuristic has been added (see Section 5.1.5) and the formula determining the number of requests to remove has been changed (see Section 6.1.1).

For completeness, we will describe the various heuristics associated with the destroy neighborhoods in Section 5.1. A repair neighborhood N^+ inserts requests from the request bank into one or more legal routes. The associated insertion heuristics are described in Section 5.2. The local search framework used at the master level is simulated annealing to be described in Section 5.3. Section 5.4 describes the noising method used to diversify the search of the heuristics. Finally, the scheme used for adjusting the weights in the roulette wheel selection is described in Section 5.5.

5.1. Request removal

The ALNS heuristic for the RPDPTW makes use of seven different removal heuristics, each searching a given removal neighborhood N^-. The heuristics take as input a given solution x and output q requests that have been removed from the routes.

5.1.1. Random removal

The simplest removal heuristic, `random removal`, selects q requests at random and removes them from the solution. This obviously has the effect of diversifying the search.

5.1.2. Worst removal

The purpose of the `worst removal` heuristic is to choose a number of requests that are very expensive, or which somehow spoil the structure of the current solution. In the RPDPTW it seems reasonable to try to remove requests with high cost and insert them at another place in the solution to obtain a better solution value.

Given a request i and a solution x, define $f'(x, i)$ as the cost of the solution where request i has been removed completely (it is not even in the request bank). Define Δf_{-i} as $\Delta f_{-i} = f(x) - f'(x, i)$.

The `worst removal` heuristic now repeatedly chooses a new request i, having the largest cost Δf_{-i} until q requests have been removed. The removal heuristic is randomized, the randomization is controlled by the parameter p. If p is small, the most expensive request is selected, while less expensive requests may be chosen for larger values of p with a probability that decreases with the cost Δf_{-i}. We refer to [3] for additional details.

5.1.3. Related removal

The purpose of the `related removal` heuristic is to remove a set of requests that in some sense are *related* and hence easy to interchange. For the RPDPTW, we define the relatedness r_{ij} of two orders i and j solely by the distance between the requests, as introduced by Ropke and Pisinger [3]. Since each request i consists of a pickup node i and a delivery node $i + n$ we get the expression

$$r_{ij} = \frac{1}{D}(d'(i, j) + d'(i, j + n) + d'(i + n, j) + d'(i + n, j + n)), \tag{20}$$

where the distance measure $d'(u, v)$ between two nodes in this context is defined as

$$d'(u, v) = \begin{cases} d_{uv} & \text{if } u \text{ and } v \text{ are not located at a terminal,} \\ 0 & \text{if } u \text{ or } v \text{ is located at a terminal.} \end{cases} \tag{21}$$

The motivation for neglecting the distance from a terminal is that the terminal is going to be visited in any case, and hence should not contribute to the relatedness measure of two requests.

The denominator D is set to the number of nonzero terms in Eq. (20), i.e. the number of pickups and deliveries taking place at a site different from a terminal. Hence, if all nodes are different from a terminal we set $D := 4$ while if both requests have a pickup at a terminal we set $D := 1$.

The relatedness measure is used to remove customers as described in Shaw [32]. The algorithm initially selects a request i by random. Then it repeatedly chooses an already selected request j and selects a new request which is most related to j. The algorithm stops when q requests have been chosen. Like in the `worst removal` heuristic (Section 5.1.2) the process is controlled by a randomization parameter p. If p is zero, the most related request is always chosen in the inner loop. If $p > 0$ a less related request may be chosen, where the probability of choosing a request decreases with the relatedness measure r_{ij} and increases with p. The algorithm is described in more detail in [3].

5.1.4. Cluster removal

The `cluster removal` heuristic is a variant of the `related removal` heuristic in which we try to remove clusters of related requests from a few routes. As a motivation, consider a route where the requests are grouped into two geographical clusters. When removing requests from such a route it is often important to remove one of these clusters entirely as the insertion methods otherwise would be prone to insert the removed requests back into the route. The `related removal` heuristic from Section 5.1.3 has a tendency to leave requests from such a cluster on the original route so, therefore, we propose a heuristic that seeks to remove an entire cluster at once.

Although, we could use the same algorithm as above for selecting related requests—just restricted to a single route—we have chosen to use a heuristic based on strongly connected components, as described in Section 4.2. We simply run Kruskal's algorithm for the minimum spanning tree problem (using r_{ij} for the edge distances) and terminate the algorithm when two connected components remain. One of these clusters is chosen at random and the requests from the chosen cluster are removed. If less than q requests have been selected, we randomly pick a removed request and choose a request from a different route, that is most related to the given request. The route of the new request is partitioned into two clusters and so the process continues until the desired number of requests has been removed. We refer the reader to Ropke and Pisinger [4] for more details.

5.1.5. Time-oriented removal

The `time oriented removal` is another variant of the `related removal` heuristic. In this heuristic, we try to remove requests that are served at roughly the same time as we hope that these requests are easy to interchange.

The heuristic works as follows. A request $\tilde{r}$ is chosen at random and the B requests that are closest to $\tilde{r}$ (according to the distance r_{ij} defined in (20)) are marked. We define a time-oriented distance between two requests as

$$\Delta t_{ij} = |t_{p_i} - t_{p_j}| + |t_{d_i} - t_{d_j}|, \tag{22}$$

where t_{p_i} and t_{d_i} are the times of the pickup and the delivery of request i in the current solution. Among the B marked requests we select the $q-1$ that are closest to $\tilde{r}$ according to Δt_{ij}. The process is controlled by a randomization parameter p like in the `related removal` heuristic described in Section 5.1.3. These requests are removed together with $\tilde{r}$.

Before running the removal heuristic we first select a subset of all requests that are geographically close to the chosen request, as we observed that this selection made the heuristic perform better on large instances. The reason for this is that if the heuristic only considered requests that are close to the chosen request time-wise, then only one or two requests would be removed from each route in the larger problems, and this makes it hard to make any major improvements to the solution.

5.1.6. Historical node-pair removal

It is well-known from several metaheuristics that using historical information in the local search (e.g. the long term memory or the aspiration level in tabu search) may improve the performance of a local search algorithm. In the present heuristic, we look at the historical success of visiting two nodes right after each other in a route, while the heuristic in Section 5.1.7 looks at the historical success of servicing two requests by the same vehicle.

The `historical node-pair removal` heuristic (denoted the *neighbor graph removal heuristic* in [4]) makes use of both historical information and the present solution when removing the requests. With each pair of nodes $(u, v) \in A$ we associate a weight $f^*_{(u,v)}$ which indicates the best solution value found so far, in a solution which used edge (u, v). Initially $f^*_{(u,v)}$ is set to infinity, and each time a new solution is found, we update the weights $f^*_{(u,v)}$ of all edges used in the given solution, for which the edge weight can be improved.

We may use the edge weights $f^*_{(u,v)}$ to remove requests that seem to be misplaced. The removal heuristic simply calculates the cost of a request $(i, i+n)$ in the current solution by summing the weights of edges incident to i and $i+n$. The most costly request is removed, and the process is repeated until q requests have been extracted. To ensure some variation in the extracted requests, randomness is introduced in the removal process.

5.1.7. Historical request-pair removal

An alternative history-based removal heuristic can make use of the historical success of placing pairs of requests in the same route. We will call this approach `historical request-pair removal` (denoted *request graph removal* in [4]).

For this purpose, we introduce the weight $h_{(a,b)}$ for each pair of requests $(a, b) \in \{1, \ldots, n\} \times \{1, \ldots, n\}$. The weight $h_{(a,b)}$ denotes the number of times the two requests a and b have been served by the same vehicle in the B best unique solutions observed so far in the search. Initially, $h_{(a,b)}$ is set to zero, and each time a new unique top-B solution is observed, the weights are incremented and decremented according to the solutions entering and leaving the top-B solutions. An appropriate value for B was experimentally found to be 100.

The weights $h_{(a,b)}$ could be used in a similar way as in the historical node-pair removal heuristic described above, but initial experiments indicated that this was an unpromising approach. Instead, the graph is used to define the relatedness

between two requests, such that two requests are considered to be related if the weight of the corresponding edge in the request graph is high. This relatedness measure is used as in the related removal heuristic described in Section 5.1.3.

5.2. Inserting requests

The considered insertion heuristics all construct a number of routes for the vehicles. As each route can be considered as an individual sub-problem the heuristics can build the routes *sequentially* or *in parallel* as discussed in Section 4.2. The sequential heuristics build one route at a time while parallel heuristics construct several routes at the same time. The heuristics presented in this paper are all parallel, as they are used in a context where a number of partial routes $k \in R$ are given, and a number of unplaced requests U is inserted from the request bank.

5.2.1. Basic greedy heuristic

A simple greedy approach is to repeatedly insert a request in the cheapest possible route. More formally, let $\Delta f_{i,k}$ denote the change in the objective value incurred by inserting request i at the *cheapest* position in route k. We set $\Delta f_{i,k} = \infty$ if request i cannot be inserted in route k. Following the greedy approach, we calculate

$$(i, k) := \arg \min_{i \in U, k \in R} \Delta f_{i,k} \tag{23}$$

and insert request i in route k at its minimum cost position. This process continues until all requests have been inserted or no more requests are feasible. The time complexity of this `basic greedy` heuristic is decreased by tabulating all values of $\Delta f_{i,k}$ and noting that only one route is changed in each iteration.

5.2.2. Regret heuristics

An obvious problem with the `basic greedy` heuristic is that it often postpones the placement of difficult requests to the last iterations where we do not have much freedom of action. The `regret` heuristic tries to circumvent the problem by incorporating a kind of look-ahead information when selecting the request to insert. Regret heuristics have been used by Potvin and Rousseau [38] for the VRPTW and in the context of the generalized assignment problem by Trick [39].

Let Δf_i^q denote the change in the objective value incurred by inserting request i into its best position in the qth *cheapest* route for request i. For example, Δf_i^2 denotes the change in the objective value by inserting request i in the route where the request can be inserted *second cheapest*. In each iteration, the regret heuristic chooses to insert the request i according to

$$i := \arg \max_{i \in U} (\Delta f_i^2 - \Delta f_i^1). \tag{24}$$

The request is inserted in the best possible route at the minimum cost position. In other words, we maximize the difference of cost of inserting the request i in its best route and its second best route. We repeat the process until no more requests can be inserted.

The heuristic can be extended in a natural way to define a class of regret heuristics: the `regret-`q heuristic is the construction heuristic that in each construction step chooses to insert request i given by

$$i := \arg \max_{i \in U} \left(\sum_{h=2}^{q} \Delta f_i^h - \Delta f_i^1 \right). \tag{25}$$

Ties are broken by selecting the request with smallest insertion cost. The request i is inserted at its minimum cost position, in its best route.

The `regret` heuristic based on criteria (24) is obviously a `regret-2` heuristic and the `basic greedy` heuristic from Section 5.2.1 is a `regret-1` heuristic due to the tie-breaking rules. Informally speaking, heuristics with $q > 2$ investigate the cost of inserting a request on the q best routes and chooses to insert the request whose cost difference between inserting it into the best route and the $q - 1$ best routes is largest. Compared to a `regret-2` heuristic, `regret-`q heuristics with large values of q discover earlier when the possibilities for inserting a request at a favorable place becomes limited.

5.3. Master local search framework

At the master level we have chosen to use simulated annealing as our local search framework. Our acceptance criteria in line 5 of the main algorithm depicted in Fig. 2 thus becomes to accept a candidate solution x' given the current solution x with probability

$$e^{-(f(x')-f(x))/T}, \tag{26}$$

where $T > 0$ is the *temperature*. We use a standard exponential cooling rate, starting from the temperature T_{start} and decreasing T according to the expression $T = Tc$, where c is the *cooling rate*, $0 < c < 1$. We calculate T_{start} by inspecting our initial solution. The following method was developed in [4] and works well when the number of requests in the problems to be solved is relatively constant. First, the cost z' of the initial solution is calculated using a modified objective function. In the modified objective function, Γ (cost of having requests in the request bank) is set to zero. The start temperature is now set such that a solution that is w percent worse than the current solution is accepted with probability 0.5. The reason for setting Γ to zero is that typically this parameter is large and could cause us to set the starting temperature too high if the initial solution had some requests in the request bank. Now, w is a parameter that has to be set. We denote this parameter the *start temperature control parameter*. We have observed that this approach is better at coping with instances of different sizes if we divide the start temperature found by the number of requests in the instance.

5.4. Applying noise to the objective function

As mentioned in Section 4.1 it can be necessary to use noising or randomization in the destroy and repair heuristics, as a diversification operator at the master level is not sufficient.

For the RPDPTW problem we have chosen to add a noise term to the objective function of the insertion heuristics. Every time we calculate the cost C of a request insertion into a route, we add some noise δ and calculate a modified insertion cost $C' = \max\{0, C + \delta\}$. The noise δ is chosen as a random number in the interval $[-N_{\max}, N_{\max}]$, where $N_{\max} = \eta \max_{i,j \in V}\{d_{ij}\}$, and η is a parameter that controls the amount of noise. We use the maximum distance to make the noise level proportional to the objective value. The distances form part of the objective function in all problems considered, hence the noise level is somehow proportional to the objective function.

Every insertion heuristic is split into two heuristics—one using noise, and one using the original objective function only. After selecting which removal and insertion heuristic to use, it is decided if the clean or the noise imposed insertion heuristic should be used. This is again done using the roulette wheel selection principle as we keep track of how well the insertion heuristics with and without noise have been performing recently. Note that we do not keep track of how well each individual insertion heuristic is performing with and without noise, but only the insertion heuristics in general.

5.5. Adaptive weights adjustment

The roulette wheel selection mechanism in the ALNS framework presented in Section 4.1 is based on the scores π_i of the respective heuristics. A high score corresponds to a successful heuristic, and hence the heuristic should be chosen with larger probability.

The scores are collected during some small time segments, defined as 100 iterations. The *observed* score $\overline{\pi}_{i,j}$ of a heuristic i in time segment j is incremented with the following values depending on the new solution x':

- σ_1 The last remove–insert operation resulted in a new global best solution x'.
- σ_2 The last remove–insert operation resulted in a solution x' that has not been accepted before, and the cost of the new solution is better than the cost of current solution.
- σ_3 The last remove–insert operation resulted in a solution x' that has not been accepted before. The cost of the new solution is worse than the cost of current solution, but the solution was accepted.

We distinguish between the two latter situations since we prefer heuristics that are able to improve on the solution, but we also want to reward heuristics that can diversify the search to some extent. We keep track of visited solutions by assigning a hash key to each solution and storing the key in a hash table.

At the end of each segment we calculate the *smoothened* scores to be used in the roulette wheel selection as

$$\pi_{i,j+1} = \rho \frac{\overline{\pi}_{i,j}}{a_i} + (1-\rho)\pi_{i,j}, \tag{27}$$

where a_i is the number of times the heuristic has been called in the time segment. The *reaction factor* ρ controls how quickly the weight adjustment algorithm reacts to changes in the scores. If $\rho = 1$ then the roulette wheel selection is only based on the scores in the most recent segment, while if $\rho < 1$ the scores of past segments is also taken into account. For an illustration of how the scores evolve during a search we refer the reader to [3].

5.6. Minimizing the number of vehicles used

The presented heuristic minimizes the travel costs, hence in order to minimize the number of vehicles also, we use a two-stage approach.

Starting from a heuristic solution which makes use of m vehicles, we repeatedly remove one route and place the corresponding requests in the request bank. If the ALNS heuristic is able to find a solution that serves all requests we proceed with a lower number of routes. We assign a large cost Γ to requests in the request bank to encourage solutions with all requests serviced.

If the ALNS heuristic fails to find a solution with all requests serviced, the algorithm steps back to the last feasible solution encountered and proceeds with the second stage of the algorithm which consists of the ordinary ALNS heuristic with the last found feasible solution as a starting point. For additional detail on the two-stage algorithm see [3].

A different two-stage approach was used by Bent and Van Hentenryck [9], in which two distinct neighborhoods and metaheuristics were used for the two stages.

5.7. Initial solution

The initial solution used in the local search is found by a regret-2 heuristic. All requests are initially placed in the request bank, and the regret-2 heuristic is run in parallel for all vehicles.

6. Computational experiments

6.1. Parameter tuning

In order to keep the parameter tuning to a minimum we have used almost the same parameter setting as determined in [3], with the exception of the cooling rate c and the start temperature control parameter w. These were calibrated by selecting five reasonable values for each parameter and testing the 25 possible combinations on eight VRPTW instances with between 100 and 1000 customers. This was done separately for both the vehicle minimizing ALNS and the ordinary distance minimizing ALNS, so different values for c and w are used when trying to find a feasible solution and when minimizing the distance.

6.1.1. Selecting the number of requests to remove

In our past work [3,4], we have removed up to 100 requests in each iteration. Experiments indicated that we seldom accepted the moves resulting from such removals as the insertion heuristics are too weak. Consequently, the maximum number of requests that can be removed in a single iteration has been reduced to 60. It was also observed that moves resulting from removing a small number of requests often were accepted, but seldom lead to any major improvements of the solution. Therefore, we now remove at least $0.1n$ requests in each iteration. To be precise, the number of requests to remove is found as a random number between $\min\{0.1n, 30\}$ and $\min\{0.4n, 60\}$. That is, for small instances the number of requests to remove will be in the interval $[0.1n, 0.4n]$ while for larger instances the interval is $[30, 60]$.

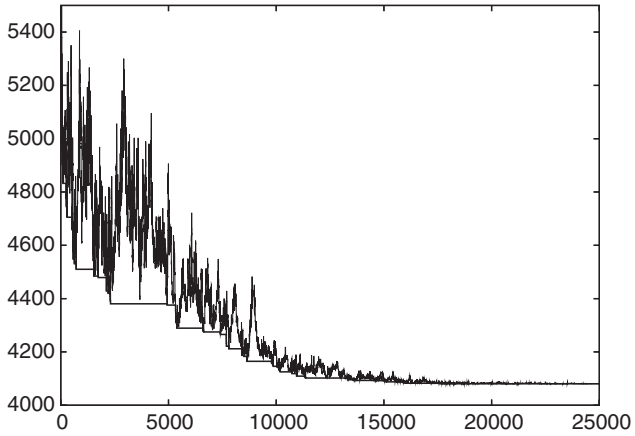

Fig. 3. Solution cost as function of iteration count. Along the x-axis we show the iteration count while the y-axis shows solution cost. The upper graph is the cost of the accepted solutions while the lower graph is the cost of the currently best known solution.

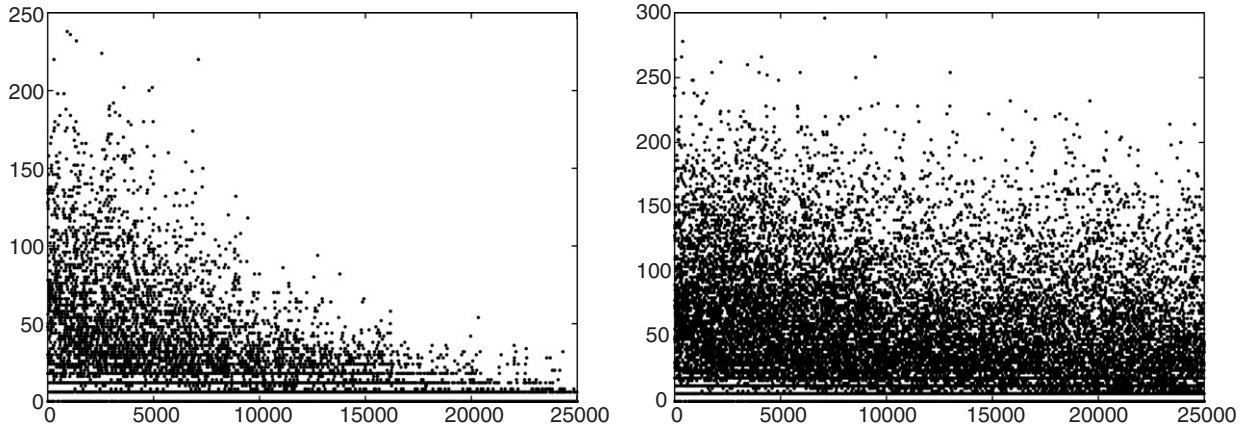

Fig. 4. (Left) Difference between accepted solutions. The figure shows the Hamming distance between an accepted solution and the last accepted solution. (Right) Difference between proposed solution and last accepted solution. The figure shows the Hamming distance between each proposed solution and the last accepted solution. The x-axis shows iteration count and the y-axis shows solution distance.

6.2. Analysis of typical search

In order to illustrate how the present ALNS heuristic works, we have produced a number of figures by running the heuristic on a 200 customer VRPTW instance minimizing the traveled distance. All figures are from the same search.

Fig. 3 shows the cost of the accepted solutions and the best known solution as a function of the iteration count. The figure is very typical for a simulated annealing metaheuristic. Initially, very poor moves are accepted and consequently the graph of accepted solutions is fluctuating wildly. As the temperature is decreased the fluctuations become smaller and they eventually nearly die out such that only improving solutions or very mildly deteriorating solutions are accepted.

The next sequence of figures all show the distance between selected solutions. We have chosen to define the distance between two solutions x and x' as the Hamming distance between the corresponding binary edge-variables. Fig. 4 (left) shows the distance between each new accepted solution and the previously accepted solution (the current solution). The figure illustrates that in the first half of the search the ALNS can make huge changes to the solution in a single move as discussed in Section 4.3. In the other half of the search only small moves are accepted. Fig. 4 (right) depicts the difference between each proposed solution and the last accepted solution. The figure shows that large moves are proposed throughout the search process, but towards the end of the search these large moves are not accepted.

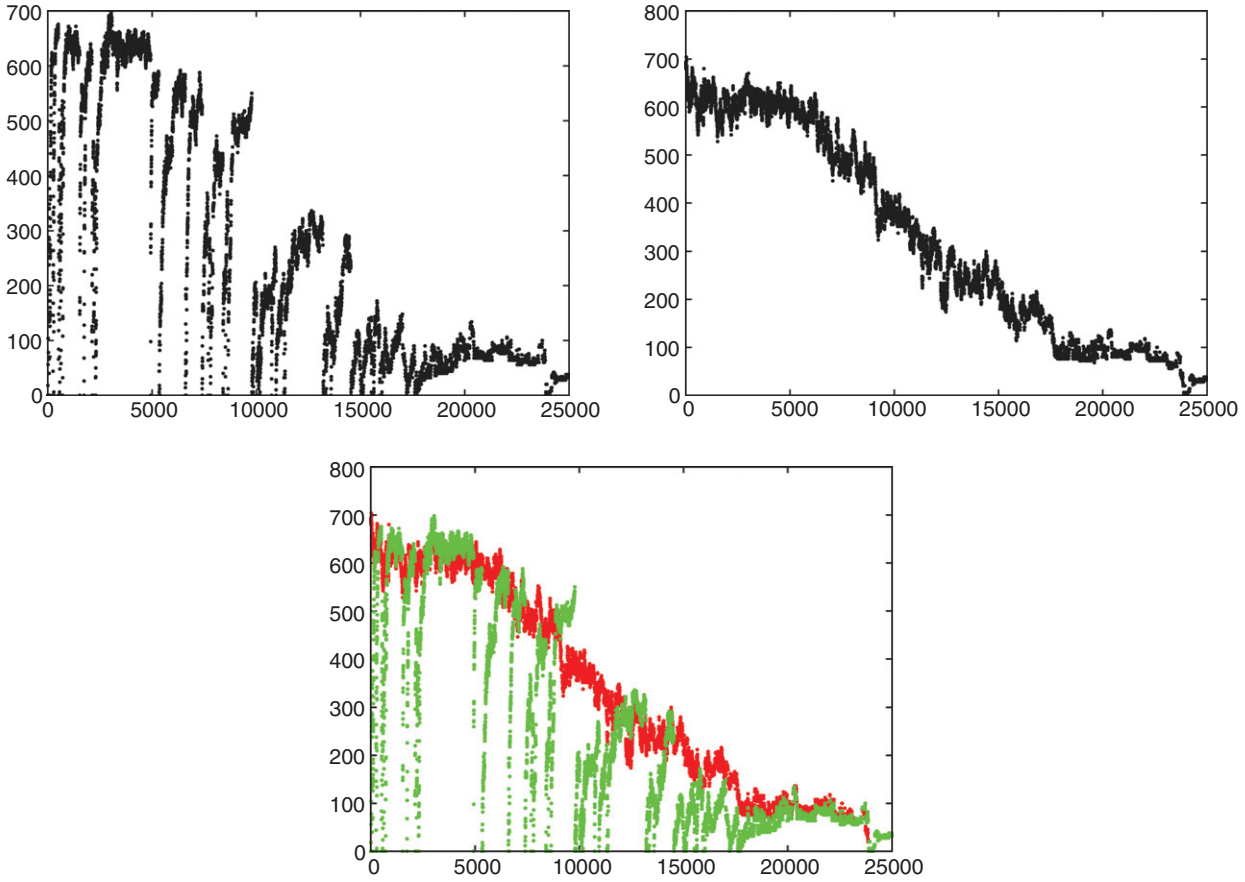

Fig. 5. (Top left) Hamming distance between accepted solutions and the currently best known solution. (Top right) Hamming distance between accepted solutions and the best solution found during the search. (Bottom) The two plots showed in the same diagram. The x-axis shows iteration count and the y-axis shows Hamming distance.

The above observations cause us to suggest some possible improvements to the algorithm: (1) Towards the end of the search it seems to be beneficial to reduce the number of requests q that are removed in each iteration as the simulated annealing framework generally will accept only minor changes. This could speed up the algorithm or allow us to perform more iterations within the same amount of time. (2) Several moves have distance zero, meaning that no changes were made to the solution vector. Obviously, such moves should be avoided, possibly by incorporating a tabu-like principle in the insertion heuristics.

Fig. 5 (top left) shows the Hamming distance between the accepted solutions and the previously best known solution. Every time the distance reaches zero, we have most likely found a new best solution (or we have returned to the previously best known solution). It is interesting to see how quickly the search moves away from the currently best known solution. This behavior is contrary to some of the ideas behind the variable neighborhood metaheuristics and the noising method, where one tries to stick around the currently best known solution or return to it if the current search direction seems fruitless. Also notice that we move very far away from the best solutions. This can be seen as the number of edges in a solution is equal to $2n + m$. The maximum Hamming distance between two solutions is therefore $2(2n + m)$. In the instance studied in this section $n = 200$ and $m = 20$, thus the maximum Hamming distance for this instance is 840.

Fig. 5 (top right) shows the Hamming distance from each accepted solution and the best solution found throughout the search. It is interesting to see that this plot is much more steady compared to the plot in Fig. 5 (top left) and that even though we are moving very far away from the previously best known solution, the distance to the overall best solution (which of course is unknown early in the search) remains roughly stable.

Fig. 5 (bottom) combines the two previous plots. The upper contours of the two plots fit each other surprisingly well. This indicates that the ALNS heuristic quickly moves away from the currently best known solution until the distance to the currently best known solution is roughly the same as the distance to the final best known solution. The search then visits solutions where the two distances are roughly the same until a new best solution is found. We believe that the simulated annealing framework is responsible for this behavior.

6.3. Application of the heuristic to standard benchmark problems

In this section, we examine how the proposed heuristic performs on standard benchmark instances for the five problem types considered in this paper. In order to investigate how much influence the number of LNS iterations has on the solution quality, we have tested two configurations of our algorithm. One version (*ALNS*-25*K*) that does 25 000 iterations while minimizing the total traveled distance and one that does 50 000 iterations (*ALNS*-50*K*). Both configurations use up to 25 000 iterations in the vehicle minimization stage. The cooling rate c in the simulated annealing algorithm described in Section 5.3 was adjusted such that both configurations go through the same temperature span.

We have applied the heuristic to each instance 5 or 10 times, depending on the instance size. We report the best solution value out of the 5 or 10 experiments as well as the average solution value.

All experiments were performed on a 3 GHz Pentium 4 computer. Detailed results from the experiments can be found in our technical report [40]. As mentioned before, the same parameter configuration has been used for all experiments.

6.3.1. Vehicle routing problems with time windows

A large number of metaheuristics have been proposed for solving the VRPTW. Bräysy and Gendreau [7] have surveyed most of these approaches, and their survey contains 47 metaheuristics. Most of these metaheuristics have been applied to the *Solomon data set* [41]. The Solomon data set contains 56 VRPTW instances that all contain 100 customers. The instances contain a variety of customer and time window distributions and have proved to be a challenge for both heuristics and exact methods since their introduction. Most of the proposed metaheuristics use vehicle minimization as primary objective and travel distance minimization as secondary objective, we prioritize our objectives in the same way. In this section, we compare the ALNS heuristic to the "best" of the previously proposed metaheuristics. It is hard to decide which of the previously proposed metaheuristics that are the best, as several criteria for comparing the heuristics could be used. In this paper, we have selected the metaheuristics that have been able to reach the minimum number of total vehicles used for all of the instances in the Solomon data set, as these in a certain sense can be regarded as the best heuristics in terms of solution quality. Table 1 summarizes this comparison.

The table shows that the ALNS heuristic is able to compete with the best heuristics for the VRPTW when considering the moderately sized Solomon instances, even though it was not specifically designed for this problem type. The heuristics by Homberger and Gehring [10] and Bent and Van Hentenryck [9] obtain slightly better results compared to the best solutions obtained by ALNS-25K, but the papers do not state how many experiments that were performed to reach these results. On the other hand, ALNS-25K reaches slightly better solutions than the three remaining heuristics and the computational time is reasonable. The column showing the average performance of ALNS-25K indicates that a single run of the heuristic can be performed quite fast but then one should not expect to reach the minimum number of vehicles. It does not seem worthwhile to spend 50 000 iteration instead of 25 000 for these rather small problems. During the calibration of the algorithm we discovered a new best solution to problem R207. This solution can be found in the Appendix.

When the VRPTW has been solved by exact methods in the literature one has usually considered minimizing the traveled distance without putting any limits on the number of vehicles. Furthermore, all distances are usually truncated to one decimal (see, for example, the work by Larsen [45]). In Table 2, we summarize the result of applying the ALNS-25K heuristic to the Solomon VRPTW instances using the same objective and rounding criteria as the exact methods. The heuristic has been applied to each instance 10 times and the table reports the best and average performances. The table shows that the heuristic is able to find solutions that are very close to the optimal solutions and in many cases the heuristic is able to identify the optimal solution in at least one of the test runs.

The optimal solutions have been collected from Chabrier [13], Cook and Rich [46], Danna and Le Pape [47], Feilet et al. [48], Irnich and Villeneuve [12], Kallehauge et al. [11], Kohl et al. [49] and Larsen [45].

Larger VRPTW instances have been proposed by Gehring and Homberger [30]. The Gehring/Homberger data set contains 300 instances with between 200 and 1000 customers. In Tables 3–7, we compare the ALNS heuristic to the

Table 1
Solomon instances with 100 customers

	BBB	HG	B	BH	IIKMUY	ALNS 25K		ALNS 50K	
						Best of 10	Avg. of 10	Best of 10	Avg. of 10
R1	11.92	11.92	11.92	11.92	11.92	11.92	12.03	11.92	12.03
	1221.10	1212.73	1222.12	**1211.10**	1217.40	1213.39	1216.93	1212.39	1215.16
R2	2.73	2.73	2.73	2.73	2.73	2.73	2.75	2.73	2.75
	975.43	955.03	975.12	**954.27**	959.11	958.60	968.01	957.72	965.94
C1	10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00
	828.48	**828.38**	**828.38**	**828.38**	**828.38**	**828.38**	**828.38**	**828.38**	**828.38**
C2	3.00	3.00	3.00	3.00	3.00	3.00	3.00	3.00	3.00
	589.93	**589.86**	**589.86**	**589.86**	**589.86**	**589.86**	**589.86**	**589.86**	**589.86**
RC1	11.50	11.50	11.50	11.50	11.50	11.50	11.60	11.50	11.60
	1389.89	1386.44	1389.58	**1384.17**	1391.03	1385.39	1386.91	1385.78	1385.56
RC2	3.25	3.25	3.25	3.25	3.25	3.25	3.25	3.25	3.25
	1159.37	**1108.52**	1128.38	1124.46	1122.79	1124.77	1140.06	1123.49	1135.46
CNV	405	405	405	405	405	405	407.5	405	407.5
CTD	57 952	**57 192**	57 710	57 273	57 444	57 360	57 641	57 332	57 550
CPU	P-400 MHz	P-400 MHz	P-200 MHz	SU 10	P3 1 GHz	P4 3 GHz	P4 3 GHz	P4 3 GHz	P4 3 GHz
T. (s)	1800	n/a	4950	7200	15 000	86	86	146	146
Exp.	3	n/a	1	>5	1	10	1	10	1

The table compares the ALNS heuristic to the heuristics by Berger et al. (BBB) [42], Homberger and Gehring (HG) [10], Bräysy (B) [43], Bent and Van Hentenryck (BH) [9] and Ibaraki et al. (IIKMUY) [44]. The data set is divided into six groups: R1, R2, C1, C2, RC1, RC2. For each group we report two numbers per heuristic. The top number is the number of vehicles used and the bottom number is the distance traveled. These numbers have been averaged over all the instance in the given group. The rows named CNV and CTD show the cumulative number of vehicles and distances, respectively. The row CPU shows the computer used in the experiment. "P" is an abbreviation for "Pentium", "SU" is an abbreviation for Sun Ultra. The row T. (s) shows the number of CPU seconds used for finding the solutions. The last row shows the number of experiments that were performed in order to obtain the results presented in the table (if multiple experiments were performed, the table shows the best results obtained). The two columns for the ALNS heuristic show the results obtained with the 25 000 iteration configuration and the 50 000 iteration configuration. For each configuration we show two columns. The first column shows the best result out of 10 experiments, and the second column show the average solution quality (averaged over the 10 experiments). Bold entries mark the best solution quality obtained among the heuristics in the comparison.

Table 2
Comparison of ALNS to exact methods

Customers	Instances	Solved to optimality	Optimums found	Avg. gap all (%)	Avg. gap opt. (%)	Avg. time (s)
25	56	56	56	0.02	0.02	5
50	56	53	48	0.19	0.13	15
100	56	37	27	0.36	0.26	47

The columns should be interpreted as follows: Customers—the number of customers in the test set, Instances—the number of instances in the test set, Solved to optimality—the number of instances that has been solved to optimality in the literature, Optimums found—the number of optimal solutions that were found by the heuristic, Avg. gap all (%)—the average gap over all instances, Avg. gap opt. (%)—the average gap over instances solved to optimality in the literature, Avg. time (s)—the average time in seconds spent on performing one experiment.

best heuristics that have been applied to these problems. The two heuristics that reach the best solution quality is the heuristic by Mester and Bräysy [8] and the ALNS heuristic. Overall the ALNS heuristic is better at minimizing the number of vehicles which is the primary objective of these problems. The heuristic of Mester and Bräysy is very good at minimizing the traveled distance though. The experiments show that the time used by the ALNS heuristic scales quite well with the problem size when the number of iterations is kept fixed. The 50 000 iteration ALNS configuration becomes worthwhile for the larger problems. For problems with 600 customers or more the difference in total traveled distance obtained by the ALNS-25K and ALNS-50K configurations become quite large, as the simulated annealing metaheuristic needs more iterations to obtain a good solution for large problems.

Table 3
Gehring/Homberger VRPTW instances with 200 customers

	GH99	GH01	BH	LL	LC	BHD	MB	ALNS 25K		ALNS 50K	
								Best of 10	Avg. of 10	Best of 10	Avg. of 10
R1	18.2	18.2	18.2	18.3	18.2	18.2	**18.2**	18.2	18.20	18.2	18.20
	3705.00	3855.03	3677.96	3736.20	3676.95	3718.30	**3618.68**	3635.94	3664.648	3631.226	3652.747
R2	4.0	4.0	4.1	4.1	4.0	4.0	**4.0**	4.0	4.05	4.0	4.05
	3055.00	3032.49	3023.62	3023.00	2986.01	3014.28	**2942.92**	2950.30	2950.04	2949.368	2942.594
C1	18.9	18.9	18.9	19.1	18.9	18.9	**18.8**	18.9	18.90	18.9	18.90
	2782.00	2842.08	2726.63	2728.60	2743.66	2749.83	**2717.21**	2723.10	2732.458	2721.522	2728.382
C2	6.0	6.0	6.0	6.0	6.0	6.0	6.0	6.0	6.00	**6.0**	6.00
	1846.00	1856.99	1860.17	1854.90	1836.10	1842.65	1833.57	1833.33	1836.4	**1832.947**	1834.675
RC1	18.0	18.1	18.0	18.3	18.0	18.0	18.0	18.0	18.00	**18.0**	18.00
	3555.00	3674.91	3279.99	3385.80	3449.71	3329.62	3221.34	3233.76	3282.989	**3212.282**	3257.168
RC2	4.3	4.4	4.5	4.9	4.3	4.4	4.4	4.3	4.33	**4.3**	4.33
	2675.00	2671.34	2603.08	2518.70	2613.75	2585.89	2519.79	2560.59	2592.39	**2556.874**	2578.575
CNV	694	696	697	707	694	695	**694**	694	694.8	694	694.8
CTD	176 180	179 328	171 715	172 472	173 061	172 406	**168 573**	169 370	170 589	169 042	169 941
CPU	P-200 MHz	P-400 MHz	SU 10	P-545 MHz	P-933 MHz	A-700 MHz	P4 2 GHz	P4-3 GHz	P4-3 GHz	P4-3 GHz	P4-3 GHz
T. (min)	4 × 10	4 × 2.1	n/a	182.1	5 × 10	2.4	8	4.3	4.3	7.7	7.7
Exp.	1	3	n/a	3	1	3	1	10	1	10	1

The table compares the ALNS heuristic to the heuristics by Gehring and Homberger (GH99) [30] and (GH01) [50], Bent and Van Hentenryck (BH) [9], Le Bouthillier and Cranic (LC) [51], Bräysy et al. (BHD) [52] and Mester and Bräysy (MB) [8]. The table should be interpreted like Table 1. Note that computing times are reported in minutes. Entries of the form $x \times y$ appearing in the T. (min) row indicate that the experiment was run for y minutes on a parallel computer with x processors. The "A" in the CPU row is an abbreviation for "AMD Athlon".

Table 4
Gehring/Homberger VRPTW instances with 400 customers

	GH99	GH01	BH	LL	LC	BHD	MB	ALNS 25K		ALNS 50K	
								Best of 10	Avg. of 10	Best of 10	Avg. of 10
R1	36.4	36.4	36.4	36.6	36.5	36.4	**36.3**	36.4	36.40	36.4	36.40
	8925.00	9478.22	8713.37	8912.40	8839.28	8692.17	**8530.03**	8609.38	8663.57	8540.04	8589.90
R2	8.0	8.0	8.0	8.0	8.0	8.0	**8.0**	8.0	8.00	8.0	8.00
	6502.00	6650.28	6959.75	6610.60	6437.68	6382.63	**6209.94**	6252.01	6309.84	6241.72	6277.07
C1	38.0	38.0	38.0	38.7	37.9	37.9	37.9	37.6	37.62	**37.6**	37.62
	7584.00	7855.82	7220.96	7181.40	7447.09	7230.48	7148.27	7369.88	7450.84	**7290.16**	7372.49
C2	12.0	12.0	12.0	12.1	12.0	12.0	**12.0**	12.0	12.00	12.0	12.00
	3935.00	3940.19	4154.40	4017.10	3940.87	3894.48	**3840.85**	3849.27	3884.44	3844.69	3875.95
RC1	36.1	36.1	36.1	36.5	36.0	36.0	**36.0**	36.0	36.00	36.0	36.00
	8763.00	9294.99	8330.98	8377.90	8652.01	8305.55	**8066.44**	8149.61	8240.28	8069.30	8148.81
RC2	8.6	8.8	8.9	9.5	8.6	8.9	8.8	8.5	8.64	**8.5**	8.64
	5518.00	5629.43	5631.70	5466.20	5511.22	5407.87	5243.06	5366.82	5388.76	**5335.09**	5351.56
CNV	1390	1392	1393	1414	1390	1391	1389	1385	1386.6	**1385**	1386.6
CTD	412 270	428 489	410 112	405 656	408 281	399 132	390 386	395 970	399 377	**393 210**	396 158
CPU	P-200 MHz	P-400 MHz	SU 10	P-545 MHz	P-933 MHz	A-700 MHz	P4 2 GHz	P4-3 GHz	P4-3 GHz	P4-3 GHz	P4-3 GHz
T. (min)	4 × 20	4 × 7.1	n/a	359	5 × 20	7.9	17	9.7	9.7	15.8	15.8
Exp.	1	3	n/a	3	1	3	1	5	1	5	1

Table 5
Gehring/Homberger VRPTW instances with 600 customers

	GH99	GH01	BH	LL	LC	BHD	MB	ALNS 25K		ALNS 50K	
								Best of 10	Avg. of 10	Best of 10	Avg. of 10
R1	54.5	54.5	55.0	55.2	54.8	54.5	**54.5**	54.5	54.50	54.5	54.50
	20 854.00	21 864.47	19 308.62	19 744.80	19 869.82	19 081.18	**18 358.68**	19 370.04	19 562.34	18 888.52	19 048.49
R2	11.0	11.0	11.0	11.1	11.2	11.0	11.0	11.0	11.00	**11.0**	11.00
	13 335.00	13 656.15	14 855.43	13 592.40	13 093.97	13 054.83	12 703.52	12 729.51	12 826.39	**12 619.26**	12 721.32
C1	57.9	57.7	57.8	58.2	57.9	57.8	57.8	57.5	57.56	**57.5**	57.56
	14 792.00	14 817.25	14 357.11	14 267.30	14 205.58	14 165.90	14 003.09	14 125.94	14 212.71	**14 065.89**	14 098.04
C2	17.9	17.8	17.8	18.2	17.9	18.0	17.8	17.5	17.80	**17.5**	17.80
	7787.00	7889.96	8259.04	8202.60	7743.92	7528.73	7455.83	7891.70	7834.72	**7801.296**	7682.61
RC1	55.1	55.0	55.1	55.5	55.2	55.0	**55.0**	55.0	55.00	55.0	55.00
	18 411.00	19 114.02	17 035.91	17 320.00	17 678.13	16 994.22	**16 418.63**	16 846.71	17 006.94	16 594.94	16 722.51
RC2	11.8	11.9	12.4	13.0	11.8	12.1	12.1	11.6	11.78	**11.6**	11.78
	11 522.00	11 670.29	11 987.89	11 204.90	11 034.71	11 212.36	10 677.46	10 922.44	10 938.30	**10 777.12**	10 828.45
CNV	2082	2079	2091	2112	2088	2084	2082	2071	2076.4	**2071**	2076.4
CTD	867 010	890 121	858 040	843 320	836 261	820 372	796 172	818 863	823 814	**807 470**	811 014
CPU	P-200 MHz	P-400 MHz	SU 10	P-545 MHz	P-933 MHz	A-700 MHz	P4 2 GHz	P4-3 GHz	P4-3 GHz	P4-3 GHz	P4-3 GHz
T. (min)	4 × 30	4 × 12.9	n/a	399.8	5 × 30	16.2	40	10.5	10.5	18.3	18.3
Exp.	1	3	n/a	3	1	3	1	5	1	5	1

Table 6
Gehring/Homberger VRPTW instances with 800 customers

	GH99	GH01	BH	LL	LC	BHD	MB	ALNS 25K		ALNS 50K	
								Best of 10	Avg. of 10	Best of 10	Avg. of 10
R1	72.8	72.8	**72.7**	73.0	73.1	72.8	72.8	72.8	72.80	72.8	72.80
	34 586.00	34 653.88	**33 337.91**	33 806.34	33 552.40	32 748.06	31 918.47	32 697.85	32 905.52	32 316.79	32 528.76
R2	15.0	15.0	15.0	15.1	15.0	15.0	**15.0**	15.0	15.00	15.0	15.00
	21 697.00	21 672.85	24 554.63	21 709.39	21 157.56	21 170.15	**20 295.28**	20 477.77	20 627.40	20 353.51	20 499.72
C1	76.7	76.1	76.1	77.4	76.3	76.3	76.2	75.6	75.66	**75.6**	75.66
	26 528.00	26 936.68	25 391.67	25 337.02	25 668.82	25 170.88	25 132.27	25 365.59	25 547.82	**25 193.13**	25 269.64
C2	24.0	23.7	24.4	24.4	24.1	24.2	**23.7**	23.7	23.98	23.7	23.94
	12 451.00	11 847.92	14 253.83	11 956.60	11 985.11	11 648.92	**11 352.29**	11 985.80	11 999.28	11 725.46	11 741.73
RC1	72.4	72.3	73.0	73.2	**72.3**	73.0	73.0	73.0	73.00	73.0	73.00
	38 509.00	40 532.35	30 500.15	31 282.54	**37 722.62**	30 005.95	30 731.07	29 864.06	30 016.05	29 478.3	29 625.04
RC2	16.1	16.1	16.6	17.1	15.8	16.3	15.8	15.7	15.82	**15.7**	15.82
	17 741.00	17 941.23	18 940.84	17 561.22	17 441.60	17 686.65	16 729.18	16 870.87	17 022.33	**16 761.95**	16 852.95
CNV	2770	2760	2778	2802	2766	2776	2765	2758	2762.6	**2758**	2762.2
CTD	1 515 120	1 535 849	1 469 790	1 416 531	1 475 281	1 384 306	1 361 586	1 372 619	1 381 184	**1 358 291**	1 365 178
CPU	P-200 MHz	P-400 MHz	SU 10	P-545 MHz	P-933 MHz	A-700 MHz	P4-2 GHz	P4-3 GHz	P4-3 GHz	P4-3 GHz	P4-3 GHz
T. (min)	4 × 40	4 × 23.2	n/a	512.9	5 × 40	26.2	145	13.5	13.5	22.7	22.7
Exp.	1	3	n/a	3	1	3	1	5	1	5	1

Table 7
Gehring/Homberger VRPTW instances with 1000 customers

	GH99	GH01	BH	LL	LC	BHD	MB	ALNS 25K		ALNS 50K	
								Best of 10	Avg. of 10	Best of 10	Avg. of 10
R1	**91.9**	91.9	92.8	92.7	92.2	92.1	92.1	92.2	92.30	92.2	92.30
	57 186.00	58 069.61	51 193.47	50 990.80	55 176.95	50 025.64	49 281.48	52 131.96	51 900.53	50 751.25	50 584.55
R2	19.0	19.0	19.0	19.0	19.2	19.0	19.0	19.0	19.00	**19.0**	19.00
	31 930.00	31 873.62	36 736.97	31 990.90	30 919.77	31 458.23	29 860.32	30 108.84	30 327.34	**29 780.82**	30 016.08
C1	96.0	95.4	95.1	96.3	95.3	95.8	95.1	94.6	94.72	**94.6**	94.72
	43 273.00	43 392.59	42 505.35	42 428.50	43 283.92	42 086.77	41 569.67	42 123.87	42 266.42	**41 877.00**	42 034.65
C2	30.2	29.7	30.3	30.8	29.9	30.6	**29.7**	29.7	29.90	29.7	29.86
	17 570.00	17 574.72	18 546.13	17 294.90	17 443.50	17 035.88	**16 639.54**	17 307.16	17 589.70	16 840.37	17 052.62
RC1	90.0	90.1	90.2	90.4	90.0	90.0	**90.0**	90.0	90.00	90.0	90.00
	50 668.00	50 950.14	48 634.15	48 892.40	49 711.36	46 736.92	**45 396.41**	47 735.43	48 168.74	46 752.15	47 081.64
RC2	19.0	18.5	19.4	19.8	18.5	19.0	18.7	18.3	18.46	**18.3**	18.46
	27 012.00	27 175.98	29 079.78	26 042.30	26 001.11	25 994.12	25 063.51	25 267.93	25 466.13	**25 090.88**	25 185.45
CNV	3461	3446	3468	3490	3451	3465	3446	3438	3443.8	**3438**	3443.4
CTD	2 276 390	2 290 367	2 266 959	2 176 398	2 225 366	2 133 376	2 078 110	2 146 752	2 157 189	**2 110 925**	2 119 550
CPU	P-200 MHz	P-400 MHz	SU 10	P-545 MHz	P-933 MHz	A-700 MHz	P4 2 GHz	P4-3 GHz	P4-3 GHz	P4-3 GHz	P4-3 GHz
T. (min)	4 × 50	4 × 30.1	n/a	606.3	5 × 50	39.6	600	16	16	26.6	26.6
Exp.	1	3	n/a	3	1	3	1	5	1	5	1

Table 8
Large VRPTW instances

No.	R1		R2		C1		C2		RC1		RC2	
	Veh.	Dist.	Veh.	Dist.	Veh.	Dist.	Veh.	Dist.	Veh.	Dist.	Veh.	Dist.
200 *customers*												
1	20	4785.96	4	4563.55	**20**	**2704.57**	**6**	**1931.44**	**18**	**3647.56**	6	3126.03
2	18	4059.57	4	3650.54	18	2943.83	**6**	**1863.16**	**18**	**3269.91**	5	2828.39
3	18	3387.64	**4**	**2892.07**	18	2710.21	6	1776.96	18	3034.45	**4**	**2613.12**
4	18	3086.11	4	1981.30	18	2644.92	**6**	**1713.46**	**18**	**2869.74**	**4**	**2052.74**
5	18	4125.19	4	3377.18	**20**	**2702.05**	**6**	**1878.85**	18	3430.03	**4**	**2912.13**
6	**18**	**3586.80**	4	2929.72	**20**	**2701.04**	**6**	**1857.35**	**18**	**3357.90**	**4**	**2975.13**
7	18	3160.44	4	2456.71	**20**	**2701.04**	**6**	**1849.46**	**18**	**3233.29**	**4**	**2539.85**
8	18	2971.66	**4**	**1849.87**	19	2775.48	**6**	**1820.53**	**18**	**3110.46**	**4**	**2314.61**
9	18	3802.55	4	3113.74	18	2687.83	**6**	**1830.05**	18	3114.02	4	2175.98
10	18	3312.44	4	2666.10	**18**	**2644.25**	6	1808.21	**18**	**3020.24**	4	2015.61
400 *customers*												
1	40	10 432.30	8	9338.49	40	7152.06	12	4116.33	**36**	**8813.43**	**11**	**6834.02**
2	**36**	**9115.68**	**8**	**7649.87**	**36**	**7733.55**	**12**	**3930.05**	**36**	**8118.43**	**9**	**6355.59**
3	36	7988.22	8	5998.04	**36**	**7082.13**	12	3775.32	**36**	**7663.73**	**8**	**5055.02**
4	36	7415.81	**8**	**4326.48**	**36**	**6816.17**	12	3543.60	**36**	**7368.47**	**8**	**3647.39**
5	**36**	**9479.10**	8	7252.64	40	7152.06	12	3946.14	**36**	**8426.57**	9	6119.44
6	36	8556.38	8	6212.37	40	7153.45	**12**	**3875.94**	36	8390.24	**8**	**5997.24**
7	36	7725.97	8	5136.74	**39**	**7546.78**	12	3894.98	36	8223.65	**8**	**5476.57**
8	**36**	**7390.76**	**8**	**4055.22**	**37**	**7546.32**	12	3796.00	36	7922.67	8	4877.39
9	36	8970.98	8	6507.40	36	7573.18	12	3881.21	36	7953.20	**8**	**4601.30**
10	36	8325.16	**8**	**5894.40**	36	7145.92	12	3687.13	36	7774.83	8	4355.52
600 *customers*												
1	59	21 677.41	11	18 837.28	**60**	**14 095.64**	18	7780.84	55	17 751.33	**15**	**13 163.03**
2	54	20 045.49	**11**	**15 069.24**	**56**	**14 174.12**	**17**	**8799.38**	55	16 548.43	**12**	**11 853.72**
3	54	17 733.91	**11**	**11 291.52**	**56**	**13 803.50**	**17**	**7604.00**	**55**	**15 499.02**	**11**	**9863.35**
4	54	16 374.29	**11**	**8163.24**	**56**	**13 578.66**	**17**	**6993.77**	**55**	**15 072.90**	**11**	**7231.64**
5	54	21 243.24	**11**	**15 418.00**	60	14 085.72	18	7578.12	**55**	**17 401.34**	12	12 560.43
6	54	18 948.53	**11**	**12 936.28**	**60**	**14 089.66**	18	7554.61	55	17 355.10	**11**	**12 282.52**
7	54	17 438.28	**11**	**10 269.96**	**58**	**15 017.03**	**18**	**7520.34**	55	17 058.40	**11**	**11 052.49**
8	54	16 146.17	**11**	**7752.78**	**57**	**14 343.05**	**17**	**8696.15**	55	16 510.65	11	10 488.75
9	54	20 375.70	11	13 885.52	56	13 767.45	18	7356.19	55	16 435.71	**11**	**9882.71**
10	54	18 902.19	**11**	**12 568.79**	**56**	**13 688.57**	17	7938.94	55	16 316.51	11	9340.06
800 *customers*												
1	80	37 492.04	15	28 822.48	80	25 184.38	24	11 664.00	**73**	**31 275.38**	**19**	**20 954.95**
2	72	33 816.69	**15**	**23 274.22**	**74**	**25 536.76**	24	11 428.07	73	29 172.08	**17**	**18 032.89**
3	72	30 317.49	15	18 078.82	**72**	**24 629.86**	24	11 184.67	73	28 164.66	**15**	**14 800.78**
4	72	28 568.78	**15**	**13 413.79**	**72**	**23 938.33**	23	10 999.42	73	27 201.39	**15**	**11 368.19**
5	72	35 503.63	15	25 077.09	**80**	**25 166.28**	24	11 451.57	73	30 548.23	16	19 180.13
6	72	32 360.07	15	20 969.81	**80**	**25 160.85**	24	11 403.57	73	30 511.07	15	19 075.89
7	72	29 979.63	**15**	**16 977.49**	79	25 425.92	24	11 412.08	73	30 007.82	**15**	**17 329.32**
8	**72**	**28 341.21**	**15**	**12 945.52**	**75**	**25 450.99**	**23**	**13 878.40**	73	29 547.96	**15**	**16 226.78**
9	**72**	**34 218.41**	15	22 877.21	**72**	**25 737.46**	24	11 650.10	73	29 360.93	**15**	**15 687.20**
10	72	32 569.97	**15**	**21 092.27**	**72**	**25 697.68**	23	12 103.56	73	28 993.52	15	14 944.14

Table 8 (*continued*)

No.	R1		R2		C1		C2		RC1		RC2	
	Veh.	Dist.	Veh.	Dist.	Veh.	Dist.	Veh.	Dist.	Veh.	Dist.	Veh.	Dist.
1000 *customers*												
1	100	54 720.19	19	43 264.68	**100**	**42 478.95**	**30**	**16 879.24**	90	48 933.68	**21**	**30 396.13**
2	**91**	**55 428.79**	**19**	**34 417.47**	**91**	**42 249.60**	29	17 563.06	90	46 165.33	**18**	**27 552.05**
3	91	49 634.84	**19**	**25 400.16**	**90**	**40 376.43**	30	16 109.71	90	44 014.81	18	20 811.18
4	91	45 303.47	**19**	**18 332.77**	**90**	**39 980.07**	**29**	**16 011.30**	90	42 607.34	**18**	**16 007.59**
5	92	53 089.15	19	37 746.01	**100**	**42 469.18**	30	16 596.69	90	48 934.53	**18**	**28 368.48**
6	91	54 555.32	19	30 778.85	**100**	**42 471.29**	**30**	**16 369.10**	90	48 766.98	18	28 746.61
7	91	48 141.47	**19**	**23 991.71**	**99**	**42 673.51**	31	16 590.48	90	48 005.94	18	26 765.43
8	91	44 853.70	**19**	**17 844.36**	**95**	**42 359.27**	29	18 407.27	90	47 122.61	**18**	**24 961.29**
9	92	52 015.72	19	34 349.70	**91**	**41 482.00**	**30**	**16 294.72**	90	46 889.79	18	24 113.72
10	92	49 769.85	19	31 682.52	90	42 214.60	29	17 582.15	90	46 080.51	18	23 056.75

The first column shows the problem number. The columns veh. and dist. show the number of vehicles and total distance traveled in the best solution found. The table is grouped by instance type and instance size. Bold entries indicate a best solution (either a tie with one of the heuristics from the literature or a new best solution).

The ALNS heuristic has been able to improve the best known solution for 122 out of the 300 large scale VRPTW instances. The best solutions for the large VRPTW instances obtained by the ALNS-25K and ALNS-50K configurations are shown in Table 8.

6.3.2. Multi-depot vehicle routing problem

Table 9 shows the results obtained on 33 MDVRP instances used by Cordeau et al. [1]. Both ALNS configurations have been applied 10 times to each instance. The results obtained by the ALNS heuristic are compared to the best results obtained by heuristics proposed by Chao et al. [26], Renaud et al. [25] and Cordeau et al. [1]. The heuristic that previously has achieved the best solution quality is the one proposed by Cordeau et al. The cost of a solution is defined as the total distance traveled by the vehicles. The table shows that the ALNS heuristic has been able to improve upon the best solution for a considerable number of instances. Each configuration has found 14 new best solutions, but as most of these overlap, the total number of new best solutions is 15. The individual improvements are typically rather small though. The table also shows that the ALNS heuristic is quite stable as the average gap from the best known solution never surpasses 2% and 1% in the ALNS-25K and ALNS-50K configurations, respectively. It should be mentioned that the ALNS heuristic is slower than the previously proposed heuristics. The ALNS-25K and ALNS-50K configurations use on average 2 and 4 min, respectively, to perform one experiment on a 3 GHz Pentium 4. The heuristic by Cordeau et al. on average used 11.7 min to perform one experiment on a Sun SPARCstation 10 which is considerably slower than our computer.

6.3.3. Site-dependent vehicle routing problem

The heuristic has been applied to the same test instances as used by Cordeau and Laporte [29]. The results obtained on the SDVRP instances are summarized in Table 10. The results are promising as the average solution quality of ALNS-25K overall is better than results previously published. Also the sum of the costs of the best known solutions found by the ALNS-50K configuration is more than 2% better than the previous best known solution and the best known solution was improved for 30 out of the 35 instances. The computational time needed for performing one experiment with the ALNS-25K configuration seems to be roughly comparable with the time needed for performing one experiment with the heuristic proposed by Cordeau and Laporte [29]. The ALNS-25K configuration spends on average 1.4 min to perform one experiment while the heuristic by Cordeau and Laporte spent around 12 min to perform the same task on a Sun Ultra 2 (300 MHz). It should be mentioned that the problem PR02 caused the ALNS heuristic some difficulties, as it was only able to find a feasible solution in 1 out of 10 experiments for the ALNS-25K configuration and 3 out of 10 experiments for the ALNS-50K configuration.

Table 9
Multi depot vehicle routing problems

	Best known					ALNS 25K				ALNS 50K			
	n	*t*	Type	Cost	Ref	Avg. sol.	Best sol.	Avg. gap (%)	Avg. time (s)	Avg. sol.	Best sol.	Avg. gap (%)	Avg. time (s)
P01	50	4	C	**576.87**	CGW	576.87	**576.87**	0.00	14	576.87	**576.87**	0.00	29
P02	50	4	C	**473.53**	RLB	473.53	**473.53**	0.00	14	473.53	**473.53**	0.00	28
P03	75	2	C	**641.19**	CGW	641.19	**641.19**	0.00	32	641.19	**641.19**	0.00	64
P04	100	2	C	1001.59	CGL	1008.49	1001.59	0.74	42	1006.09	**1001.04**	0.50	88
P05	100	2	C	**750.03**	CGL	753.04	751.86	0.40	58	752.34	751.26	0.31	120
P06	100	3	C	**876.5**	RLB	884.36	880.42	0.90	47	883.01	876.70	0.74	93
P07	100	4	C	885.8	CGL	889.14	**881.97**	0.81	43	889.36	**881.97**	0.84	88
P08	249	2	CD	4437.68	CGL	4426.86	**4387.38**	0.90	166	4421.03	4390.80	0.77	333
P09	249	3	CD	3900.22	CGL	3902.18	3874.75	0.74	182	3892.50	**3873.64**	0.49	361
P10	249	4	CD	3663.02	CGL	3676.93	3655.18	0.74	180	3666.85	**3650.04**	0.46	363
P11	249	5	CD	3554.18	CGL	3592.82	3552.27	1.32	174	3573.23	**3546.06**	0.77	357
P12	80	2	C	**1318.95**	RLB	1319.70	**1318.95**	0.06	38	1319.13	**1318.95**	0.01	75
P13	80	2	CD	**1318.95**	RLB	1321.10	**1318.95**	0.16	30	1318.95	**1318.95**	0.00	60
P14	80	2	CD	**1360.12**	CGL	1360.12	**1360.12**	0.00	29	1360.12	**1360.12**	0.00	58
P15	160	4	C	**2505.42**	CGL	2517.96	**2505.42**	0.50	125	2519.64	**2505.42**	0.57	253
P16	160	4	CD	**2572.23**	RLB	2577.28	**2572.23**	0.20	92	2573.95	**2572.23**	0.07	188
P17	160	4	CD	**2709.09**	CGL	2709.65	**2709.09**	0.02	90	2709.09	**2709.09**	0.00	179
P18	240	6	C	**3702.85**	CGL	3751.85	3727.58	1.32	209	3736.53	**3702.85**	0.91	419
P19	240	6	CD	**3827.06**	RLB	3846.35	3839.36	0.50	158	3838.76	**3827.06**	0.31	315
P20	240	6	CD	**4058.07**	CGL	4065.32	**4058.07**	0.18	151	4064.76	**4058.07**	0.16	300
P21	360	9	C	**5474.84**	CGL	5576.82	5519.47	1.86	293	5501.58	**5474.84**	0.49	582
P22	360	9	CD	**5702.16**	CGL	5731.10	5714.46	0.51	228	5722.19	**5702.16**	0.35	462
P23	360	9	CD	6095.46	CGL	6107.84	**6078.75**	0.48	223	6092.66	**6078.75**	0.23	443
PR01	48	4	CD	**861.32**	CGL	861.32	**861.32**	0.00	16	861.32	**861.32**	0.00	30
PR02	96	4	CD	1307.61	CGL	1311.54	**1307.34**	0.32	52	1308.17	**1307.34**	0.06	103
PR03	144	4	CD	1806.6	CGL	1810.90	**1806.53**	0.24	106	1810.66	1806.60	0.23	214
PR04	192	4	CD	2072.52	CGL	2080.55	2066.64	0.95	146	2073.16	**2060.93**	0.59	296
PR05	240	4	CD	2385.77	CGL	2352.59	2341.65	0.63	188	2350.31	**2337.84**	0.53	372
PR06	288	4	CD	2723.27	CGL	2695.15	**2685.35**	0.36	232	2695.74	2687.60	0.39	465
PR07	72	6	CD	**1089.56**	CGL	1089.56	**1089.56**	0.00	29	1089.56	**1089.56**	0.00	58
PR08	144	6	CD	1666.6	CGL	1677.31	1665.80	0.75	105	1675.74	**1664.85**	0.65	207
PR09	216	6	CD	2153.1	CGL	2148.85	**2136.42**	0.58	173	2144.84	**2136.42**	0.39	350
PR10	288	6	CD	2921.85	CGL	2913.34	**2889.49**	0.83	228	2905.43	2889.82	0.55	455
Tot.				80 394		80 651.59	80 249.57		3894	80 448.26	80 133.89		7809
Avg.								0.52	118			0.34	237
< PB							14				14		
#B				18			20				27		

The leftmost column shows the problem name, while the rest of the table is divided into three major columns that display the previously best known results and the results obtained by the ALNS-25K and ALNS-50K configurations. The sub-columns should be interpreted like this: *n*—number of customers, *t*—number of depots, type—the type of the instance (C indicates that the instance is capacity constrained, while D indicates that route duration constraints are present), cost—the cost of the previously best known solution (the cost is calculated as the total distance traveled), ref—where the solution was first reported. The following abbreviations are used: CGW—Chao et al. [26], RLB—Renaud et al. [25], CGL—Cordeau et al. [1]. The last 10 instances were introduced by Cordeau et al. [1] and the two other heuristics have not been applied to these instances. The columns best sol. and avg. sol. show the cost of the best solution and the average cost of the solutions obtained during 10 experiments. Avg. gap shows how far the average solution cost is from the best known solution. Avg. time shows how much time the heuristic spends in one experiment. The rows Tot. and Avg. sums and averages key columns. "< PB" shows how many times the best solution found by the ALNS configuration was better than the previous best known solution and #B shows the number of best known solutions obtained. Entries written in bold indicate best known solutions.

6.3.4. Capacitated vehicle routing problem

For the CVRP we have chosen to test the ALNS heuristic on three datasets. The first dataset was proposed by Christofides et al. [53] and contains instances with between 50 and 200 customers. The second dataset was proposed

Table 10
Site-dependent vehicle routing problems

	Best known				ALNS 25K				ALNS 50K			
	n	*t*	Cost	Ref	Avg. sol.	Best sol.	Avg. gap (%)	Avg. time (s)	Avg. sol.	Best sol.	Avg. gap (%)	Avg. time (s)
P01	50	3	642.66	CL	645.04	**640.32**	0.74	10	642.93	**640.32**	0.41	20
P02	50	2	**598.1**	CL	599.40	**598.10**	0.22	10	598.82	**598.10**	0.12	19
P03	75	3	959.36	CL	962.36	958.14	0.56	20	963.14	**957.04**	0.64	40
P04	75	2	**854.43**	CL	858.05	**854.43**	0.42	18	856.22	**854.43**	0.21	36
P05	100	3	1020.22	CL	1012.46	1007.51	0.89	34	1009.08	**1003.57**	0.55	68
P06	100	2	1036.02	CL	1034.09	1028.70	0.54	35	1032.67	**1028.52**	0.40	69
P07	27	3	**391.3**	CGW	391.30	**391.30**	0.00	4	391.30	**391.30**	0.00	8
P08	54	3	**664.46**	CGW	664.46	**664.46**	0.00	12	664.46	**664.46**	0.00	24
P09	81	3	**948.23**	CGW	958.69	**948.23**	1.10	24	961.36	**948.23**	1.38	47
P10	108	3	1223.88	CL	1229.42	**1218.75**	0.88	38	1225.28	**1218.75**	0.54	76
P11	135	3	1464.98	CL	1488.28	1468.38	1.70	58	1475.85	**1463.33**	0.86	116
P12	162	3	1695.67	CL	1697.98	1690.56	1.17	78	1689.62	**1678.40**	0.67	157
P13	54	3	1196.73	CL	1194.40	**1194.18**	0.02	12	1194.91	**1194.18**	0.06	24
P14	108	3	1962.66	CL	1961.11	**1960.62**	0.02	36	1960.83	**1960.62**	0.01	72
P15	162	3	2751.45	CL	2712.10	2695.22	1.01	77	2701.61	**2685.09**	0.61	152
P16	216	3	3491.18	CL	3421.74	3402.94	0.75	109	3411.50	**3396.36**	0.45	213
P17	270	3	4230.96	CL	4109.62	**4084.92**	0.60	146	4114.26	4085.61	0.72	291
P18	324	3	4929.71	CL	4821.55	4775.35	1.39	177	4795.31	**4755.50**	0.84	346
P19	100	3	850.39	CL	852.09	846.35	0.71	43	848.54	**846.07**	0.29	85
P20	150	3	1046.14	CL	1048.75	1042.21	1.74	83	1042.10	**1030.78**	1.10	168
P21	199	3	1337.83	CL	1281.58	1272.41	0.77	110	1283.03	**1271.75**	0.89	217
P22	120	3	1012.17	CL	1010.30	1008.78	0.16	65	1008.81	**1008.71**	0.01	130
P23	100	3	818.75	CL	807.67	**803.29**	0.55	37	807.00	**803.29**	0.46	73
PR01	48	4	1384.15	CL	1387.37	**1380.77**	0.48	10	1393.85	**1380.77**	0.95	19
PR02	96	4	2320.97	CL	2311.54	**2311.54**	0.00	32	2330.60	**2311.54**	0.82	63
PR03	144	4	2623.31	CL	2608.31	**2590.01**	0.71	71	2607.66	2602.13	0.68	140
PR04	192	4	3500.79	CL	3510.26	3481.44	1.04	98	3489.51	**3474.01**	0.45	191
PR05	240	4	4479.34	CL	4430.28	**4382.65**	1.09	123	4431.16	4416.38	1.11	251
PR06	288	4	4546.79	CL	4475.52	4452.93	0.70	159	4465.18	**4444.52**	0.47	314
PR07	72	6	1955.11	CL	1926.52	**1889.82**	1.94	19	1916.50	**1889.82**	1.41	39
PR08	144	6	3082.32	CL	3001.88	**2976.76**	0.84	66	3007.99	2977.50	1.05	135
PR09	216	6	3664.22	CL	3581.58	3548.22	1.28	113	3567.15	**3536.20**	0.88	226
PR10	288	6	4739.43	CL	4675.65	**4646.96**	0.62	162	4673.67	4648.76	0.57	322
PR11	1008	4	13 227.96	CL	12 987.58	12 888.47	2.11	433	12 810.71	**12 719.65**	0.72	847
PR12	720	6	9621.99	CL	9510.37	9437.14	1.30	332	9437.56	**9388.07**	0.53	658
Tot.			90 274		89 169.30	88 541.88		2853	88 810.17	88 273.77		5658
Avg.							0.80	81			0.60	162
< PB						29				30		
#B			5			18				30		

The table should be interpreted like Table 9. Column *t* shows the number of vehicle types. CL refers to the heuristic by Cordeau and Laporte [29] and CGW refers to the heuristic by Chao et al. [28]. The ALNS heuristic was applied 10 times for each problem.

by Golden et al. [18] and contains instances with up to 483 customers. The last dataset was proposed by Li et al. [19] and contains instances with up to 1200 customers. These are the so-far largest instances that the ALNS heuristic has been applied to. Table 11 summarizes these experiments. Note that we only compare the ALNS heuristic to a subset of all the CVRP heuristics that have been proposed in the literature. The heuristics used for benchmarking are the most recent heuristics that were surveyed by Cordeau et al. [16].

The table shows that the ALNS heuristic cannot compete with the well-performing heuristic by Mester and Bräysy [8], but its performance is comparable to the rest of the heuristics. For the last dataset, the heuristic proposed by Li et al. must be considered to be the best as it is very fast compared to the ALNS heuristic although the ALNS heuristic

Table 11
Capacitated vehicle routing problems

Heuristic	CPU	Christofides		Golden et al.		Li et al.	
		%	min	%	min	%	min
TV	P-200 MHz	0.64	3.84	2.88	17.55	–	–
LGV	Athlon 1 GHz	–	–	1.05	–	1.20	3.16
CGLM	P4-2 GHz	0.56	24.62	1.46	56.11	–	–
EOS	P3-733 MHz	0.24	30.95	3.77	137.95	–	–
P	P3-1 GHz	0.24	5.19	0.92	66.90	–	–
TK	P2-400 MHz	0.23	5.22	–	–	–	–
MB Best	P4-2 GHz	0.03	7.72	0.01	72.94	–	–
MB Fast	P4-2 GHz	0.07	0.27	0.94	0.63	–	–
BB	P-400 MHz	0.49	21.25	–	–	–	–
RDH	P-900 MHz	–	–	0.67	49.33	–	–
ALNS 25K Best of 10	P4-3 GHz	0.15	9.33	0.67	53.00	0.88	243.17
ALNS 25K Avg.	P4-3 GHz	0.39	0.93	1.25	5.30	2.40	24.32
ALNS 50K Best of 10	P4-3 GHz	0.11	17.50	0.49	107.67	0.50	497.90
ALNS 50K Avg.	P4-3 GHz	0.31	1.75	1.02	10.77	1.90	49.79
		14 instances 50–200 customers		20 instances 240–483 customers		12 instances 560–1200 customers	

The table compares the ALNS heuristic to nine heuristics proposed in the literature recently. The first column indicates the heuristic considered. TV—granular tabu search by Toth and Vigo [54], LGV—variable-length neighbor list record-to-record travel heuristic by Li et al. [19], CGLM—unified tabu search by Cordeau et al. [1,2], EOS—very large scale neighborhood search by Ergun et al. [55], P—evolutionary algorithm by Prins [56], TK—bone route heuristic by Tarantilis and Kiranoudis [57], MB—AGES heuristic by Mester and Bräysy [8] (two configurations of this heuristic is included in the table), BB—hybrid genetic algorithm by Berger and Barkaoui [58], RDH—ants system algorithm by Reimann et al. [59]. The table contains four rows for the ALNS heuristic. For each of the configurations ALNS-25K and ALNS-50K we report the best solution quality in 10 experiments and the average solution quality (averaged over the same 10 experiments). The CPU column lists the CPU used, *P* is used as an abbreviation for Pentium. The rest of the table contains three major columns, one for each dataset. For each of the datasets we report the gap between the solution obtained by the heuristic and the best known solution and we report the time spend on average by the heuristic to solve one instance. When reporting solution times for finding the best solution of 10 runs, the time of all runs has been included. The ALNS heuristic is the only heuristic that has been applied to all datasets, which explains the missing entries. It should be noted that some of the numbers reported in the table were obtained from the survey by Cordeau et al. [16].

overall is able to reach better solutions. We discovered one new best solution for the Golden et al. dataset and three new best solutions for the Li et al. dataset.

6.3.5. Open vehicle routing problem

The results on the OVRP are summarized in Table 12. The heuristic was tested on the same 16 instances that were used by Brandão [24] and Fu et al. [23]. The primary objective considered was to minimize the number of vehicles used, while the secondary objective was to minimize the traveled distance. The solutions obtained by the ALNS heuristic are promising as the best known solution to 11 out of the 16 instances has been improved. The running time of the ALNS heuristic is comparable to the two other heuristics: the configuration of Brandão's heuristic that obtains the best results spends on average 9.6 min to solve an instance on a 500 MHz Pentium III. In the paper by Fu et al., two configurations of their heuristic are tested. These configurations spend on average 6.6 and 13.9 min, respectively, to solve an instance on a 600 MHz Pentium II. The ALNS-25K and ALNS-50K configurations use 1.4 and 2.3 min, respectively, to solve an instance on a 3 GHz Pentium IV.

6.3.6. Computational results conclusion

The computational results presented in this section are very encouraging. The results show that the general ALNS heuristic is on par with the best specialized heuristics for the VRPTW and that the heuristic currently is the best when it comes to minimizing the number of vehicles in large VRPTW instances. One should keep in mind that numerous specialized heuristics have been proposed for the VRPTW making it difficult for a general heuristic to compete on these instances.

Table 12
Open vehicle routing problems

	Best known				ALNS 25K						ALNS 50K					
	n	#veh.	Cost	References	Avg. sol.	Avg. #veh.	Best sol.	Best #veh.	Avg. gap (%)	Avg. time (s)	Avg. sol.	Avg. #veh.	Best sol.	Best #veh.	Avg. gap (%)	Avg. time (s)
P01	50	**5**	**408.5**	FEL	416.67	5.0	416.06	5	2.00	12	416.45	5.0	416.06	5	1.95	23
P02	75	10	570.6	FEL	570.81	10.0	**567.14**	**10**	0.65	36	568.86	10.0	**567.14**	**10**	0.30	53
P03	100	**8**	**617**	FEL	642.93	8.0	641.76	8	4.20	85	642.32	8.0	641.76	8	4.10	128
P04	150	12	734.5	FEL	734.34	12.0	**733.13**	**12**	0.17	179	733.49	12.0	**733.13**	**12**	0.05	279
P05	199	16	953.4	B	912.54	16.0	897.93	16	1.84	124	907.03	16.0	**896.08**	**16**	1.22	237
P06	50	**6**	**400.6**	FEL	412.96	6.0	412.96	6	3.08	20	412.96	6.0	412.96	6	3.08	31
P07	75	10	634.5	B	592.16	10.0	584.15	10	1.54	18	588.72	10.0	**583.19**	**10**	0.95	33
P08	100	**9**	**638.2**	FEL	646.23	9.0	645.31	9	1.26	73	646.28	9.0	645.16	9	1.27	114
P09	150	13	785.2	B	766.42	13.1	759.35	13	1.13	108	764.32	13.1	**757.84**	**13**	0.85	185
P10	199	17	884.6	B	882.33	17.0	**875.67**	**17**	0.76	120	878.42	17.0	**875.67**	**17**	0.31	224
P11	120	7	683.4	B	682.68	7.0	**682.12**	**7**	0.08	73	682.39	7.0	**682.12**	**7**	0.04	141
P12	100	10	534.8	FEL	534.81	10.0	**534.24**	**10**	0.11	80	534.44	10.0	**534.24**	**10**	0.04	118
P13	120	11	943.7	B	911.98	11.0	**909.80**	**11**	0.24	61	911.12	11.0	**909.80**	**11**	0.15	116
P14	100	11	597.3	B	591.87	11.0	**591.87**	**11**	0.00	40	591.89	11.0	**591.87**	**11**	0.00	75
F11	71	**4**	**175**	FEL	177.00	4.0	177.00	4	1.14	69	177.00	4.0	177.00	4	1.14	104
F12	134	7	778.5	FEL	770.59	7.0	**770.17**	**7**	0.06	237	770.31	7.0	**770.17**	**7**	0.02	359
Tot.		156	10 340		10 246.32	156.10	10 198.67	156		1336	10 225.99	156.10	10 194.19	156		2222
Avg.									1.14	83					0.97	139
< PB							11						11			
#B			5				8						11			

The table should be interpreted like Table 9. The abbreviations used in the References column are: B—Brandao's heuristic [24], FEL—the heuristic by Fu et al. [23]. The column #veh. indicates the number of vehicles used in the previous best solution, avg. #veh. indicates the number of vehicles used on average by the particular ALNS configuration (averaged over 10 experiments). The column best #veh. indicates the number of vehicles used in the best found solution (out of 10 experiments).

For the MDVRP, SDVRP and OVRP the ALNS heuristic has been able to find many new best solutions and the results on the SDVRP are especially promising. For the CVRP the proposed heuristic is able to compete with many of the most recent heuristics, but it is outperformed by a more specialized heuristic for this problem. Nevertheless, a couple of new best solutions were found for this problem type also. One should also keep in mind that the heuristic was not tuned for each problem type, but a general parameter setting was used for all experiments.

The comparison between the fast and the slow version of the ALNS heuristic showed that it did not pay off to use the ALNS-50K variant for the smaller instances, while for instances with around 400–600 or more customers it seemed worthwhile to use the ALNS-50K configuration. Consequently, it might be useful to use a variable number of iterations I which depends on the number n of requests. E.g. $I := 20000 + 50n$.

7. Conclusion

A new general heuristic framework, denoted ALNS has been presented. The framework has been used to solve several variants of vehicle routing problems in the present paper as well as in [3,4]. This includes the vehicle routing problem with time windows (VRPTW), the capacitated vehicle routing problem (CVRP), the multi-depot vehicle routing problem (MDVRP), the site-dependent vehicle routing problem (SDVRP), the open vehicle routing problem (OVRP), the pickup and delivery problem with time windows (PDPTW), the vehicle routing problem with backhauls (VRPB), the mixed vehicle routing problem with backhauls (MVRPB), the multi-depot mixed vehicle routing problem with backhauls (MDMVRPB), the vehicle routing problem with backhauls and time windows (VRPBTW), the mixed vehicle routing problem with backhauls and time windows (MVRPBTW) and the vehicle routing problem with simultaneous deliveries and pickups (VRPSDP).

Due to the generality of the ALNS framework and the encouraging results demonstrated for a wide spectrum of VRP problems, we believe that ALNS should be considered as one of the standard frameworks for solving large-sized optimization problems.

Supply chain management is a research area getting increasing attention [60]. By co-ordinating activities in the supply chain, companies can rationalize the process resulting in mutual gains. If the involved companies co-ordinate their transportation activities we will see a need for solving mixed transportation problems, where the instances, for example, consist of a mixture of PDPTW, MDVRP and SDVRP problems. In order to handle future changes in the distribution structure, these algorithms need to be stable for various input types, and should not need to be tuned for particular problem characteristics. It should be clear that the ALNS framework is very promising for such types.

In conclusion, we may add an interesting observation: we have seen that a mixture of good and less good heuristics lead to better solutions than using good heuristics solely. It is however necessary to hierarchically control the search, such that well-performing heuristics are given most influence, but such that all heuristics participate in the solution process. Using this principle one gets a robust and well-performing solution approach.

Acknowledgements

The authors wish to thank the anonymous referee for valuable comments and corrections.

Appendix

New best solution to the Solomon R207 instance

Route	Length	Visit sequence
1	437.339	42 92 45 46 36 64 11 62 88 30 20 65 71 9 81 34 78 79 3 76 28 53 40 2 87 57 41 22 73 21 72 74 75 56 4 25 55 54 80 68 77 12 26 58 13 97 37 100 98 93 59 95 94
2	453.269	27 1 69 50 33 29 24 39 67 23 15 43 14 44 38 86 16 61 91 85 99 96 6 84 8 82 7 48 47 49 19 10 63 90 32 66 35 51 70 31 52 18 83 17 5 60 89

Total length: 890.61.

References

[1] Cordeau J-F, Gendreau M, Laporte G. A tabu search heuristic for periodic and multi-depot vehicle routing problems. Networks 1997;30: 105–19.
[2] Cordeau J-F, Laporte G, Mercier A. A unified tabu search heuristic for vehicle routing problems with time windows. Journal of the Operational Research Society 2000;52:928–36.
[3] Ropke S, Pisinger D. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation Science, to appear.
[4] Ropke S, Pisinger D. A unified heuristic for a large class of vehicle routing problems with backhauls. European Journal of Operational Research, 2004, to appear.
[5] Sigurd M, Pisinger D, Sig M. The pickup and delivery problem with time windows and precedences. Transportation Science 2004;38: 197–209.
[6] Cordeau J-F, Desaulniers G, Desrosiers J, Solomon MM, Soumis F. Vrp with time windows. In: Toth P., Vigo D, editors. The vehicle routing problem, SIAM monographs on discrete mathematics and applications, vol. 9. Philadelphia: SIAM; 2002. p. 157–93 [chapter 7].
[7] Bräysy O, Gendreau M. Vehicle routing problem with time windows, part II: metaheuristics. Transportation Science 2005;39:119–39.
[8] Mester D, Bräysy O. Active guided evolution strategies for large-scale vehicle routing problems with time windows. Computers and Operations Research 2005;32:1593–614.
[9] Bent R, Hentenryck PV. A two-stage hybrid local search for the vehicle routing problem with time windows. Transportation Science 2004;38: 515–30.
[10] Homberger J, Gehring H.A two-phase hybrid metaheuristic for the vehicle routing problem with time windows. European Journal of Operational Research 2005;162:220–38.

[11] Kallehauge B, Larsen J, Madsen OBG. Lagrangean duality applied on vehicle routing with time windows—experimental results. Technical Report IMM-REP-2000-8, Informatics and Mathematical Modelling, Technical University of Denmark, DTU Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby;2001.
[12] Irnich S, Villeneuve D. The shortest path problem with resource constraints and k-cycle elimination for $k \geqslant 3$. Technical Report G-2003-55, GERAD, Montreal, Canada; September 2003.
[13] Chabrier A. Vehicle routing problem with elementary shortest path based column generation. Working Paper, ILOG, Madrid, 2003.
[14] Laporte G, Semet F. Classical heuristics for the capacitated vrp. In: Toth P., Vigo D, editors. The vehicle routing problem, SIAM monographs on discrete mathematics and applications, vol. 9. Philadelphia: SIAM; 2002. p. 109–28 [chapter 5].
[15] Gendreau M, Laporte G, Potvin J-Y. Metaheuristics for the capacitated vrp. In: Toth P., Vigo D, editors. The vehicle routing problem, SIAM monographs on discrete mathematics and applications, vol. 9. Philadelphia: SIAM; 2002. p. 129–54 [chapter 6].
[16] Cordeau J-F, Gendreau M, Hertz A, Laporte G, Sormany J-S. New heuristics for the vehicle routing problem. Technical Report G-2004-33, GERAD, Montreal, Canada; 2004.
[17] Taillard E. Parallel iterative search methods for vehicle routing problems. Networks 1993;23:661–73.
[18] Golden BL, Wasil EA, Kelly JP, Chao I-M. Metaheuristics in vehicle routing. In: Crainic T, Laporte G, editors. Fleet management and logistics. Boston, MA: Kluwer; 1998. p. 33–56.
[19] Li F, Golden B, Wasil E. Very large-scale vehicle routing: new test problems, algorithms, and results. Computers and Operations Research 2005;32:1165–79.
[20] Lysgaard J, Letchford AN, Eglese RW. A new branch-and-cut algorithm for the capacitated vehicle routing problem. Mathematical Programming, Series A 2004;100:423–45.
[21] Fukasawa R, Lysgaard J, de Aragão MP, Reis M, Uchoa E, Werneck RF. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. In: Proceedings of IPCO X. New York: Columbia University; 2004.
[22] Sariklis D, Powell S. A heuristic method for the open vehicle routing problem. Journal of the Operational Research Society 2000;51:564–73.
[23] Fu Z, Eglese RW, Li L. A tabu search heuristic for the open vehicle routing problem. Technical Report 2003/042, Lancaster University Management School, Lancaster, United Kingdom; 2003.
[24] Brandão J. A tabu search algorithm for the open vehicle routing problem. European Journal of Operational Research 2004;157(3):552–64.
[25] Renaud J, Laporte G, Boctor FF. A tabu search heuristic for the multi-depot vehicle routing problem. Computers and Operations Research 1996;23(3):229–35.
[26] Chao I, Golden BL, Wasil E. A new heuristic for the multi-depot vehicle routing problem that improves upon best-known solutions. American Journal of Mathematics and Management Science 1993;13:371–406.
[27] Nag B, Golden BL, Assad A. Vehicle routing with site dependencies. In: Golden B, Assad A, editors. Vehicle routing: methods and studies. Amsterdam: Elsevier; 1988. p. 149–59.
[28] Chao I-M, Golden B, Wasil E. A computational study of a new heuristic for the site-dependent vehicle routing problem. INFOR 1999;37(3): 319–36.
[29] Cordeau J-F, Laporte G. A tabu search algorithm for the site dependent vehicle routing problem with time windows. INFOR 2001;39:292–8.
[30] Gehring H, Homberger J. A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. In: Miettinen K, Mäkelä M, Toivanen J, editors. Proceedings of EUROGEN99—Short course on evolutionary algorithms in engineering and computer science, University of Jyväskylä, 1999. p. 57–64.
[31] Desaulniers G, Desrosiers J, Erdmann A, Solomon MM, Soumis F. Vrp with pickup and a delivery. In: Toth P., Vigo D, editors. The vehicle routing problem, SIAM monographs on discrete mathematics and applications, vol. 9. Philadelphia: SIAM; 2002. p. 225–42 [chapter 9].
[32] Shaw P. Using constraint programming and local search methods to solve vehicle routing problems. In: CP-98, Fourth international conference on principles and practice of constraint programming, Lecture notes in computer science, vol. 1520, 1998. p. 417–31.
[33] Schrimpf G, Schneider J, Stamm-Wilbrandt H, Dueck G. Record breaking optimization results using the ruin and recreate principle. Journal of Computational Physics 2000;159(2):139–71.
[34] Dees Jr WA, Karger PG. Automated rip-up and reroute techniques. In: Proceedings of the 19th conference on design automation. New York: IEEE Press; 1982. p. 432–9.
[35] Ahuja RK, Ergun Ö, Orlin JB, Punnen AP. A survey of very large-scale neighborhood search techniques. Discrete Applied Mathematics 2002;123:72–102.
[36] Hansen P, Mladenovic N. An introduction to variable neighborhood search. In: Vess S et al., editor. Metaheuristics, advances and trends in local search paradigms for optimization. Dordrecht: Kluwer Academic Publishers; 1999. p. 433–58.
[37] Pisinger D. Upper bounds and exact algorithms for p-dispersion problems. Computers and Operations Research 2006;33(5):1380–98.
[38] Potvin J-Y, Rousseau J-M. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. European Journal of Operational Research 1993;66:331–40.
[39] Trick MA. A linear relaxation heuristic for the generalized assignment problem. Naval Research Logistics 1992;39:137–52.
[40] Pisinger D, Ropke S. A general heuristic for vehicle routing problems. Technical Report 05/01, DIKU, University of Copenhagen; 2005.
[41] Solomon MM. Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations Research 1987;35(2): 254–65.
[42] Berger J, Barkaoui M, Bräysy O. A route hybrid genetic approach for the vehicle routing problem with time windows. INFOR 2003;41(2).
[43] Bräysy O. A reactive variable neighborhood search for the vehicle-routing problem with time windows. INFORMS Journal on Computing 2003;15(4):347–68.
[44] Ibaraki T, Imahori S, Kubo M, Masuda T, Uno T, Yagiura M. Effective local search algorithms for routing and scheduling problems with general time window constraints. Transportation Science 2005;39:206–32.
[45] Larsen J. Parallelization of the vehicle routing problem with time windows. PhD thesis, Department of Mathematical Modelling, Technical University of Denmark; 1999. IMM-PHD-1999-62.

[46] Cook W, Rich JL. A parallel cutting-plane algorithm for the vehicle routing problem with time windows. Technical Report TR99-04, Department of Computational and Applied Mathematics, Rice University, 1999.
[47] Danna E, Pape CL. Accelerating branch-and-price with local search: a case study on the vehicle routing problem with time windows. Technical Report 03-006, ILOG, ILOG S.A, 9, rue de Verdun, F-94253 Gentilly Cédex; September 2003.
[48] Feillet D, Dejax P, Gendreau M, Gueguen C. An exact algorithm for the elementary shortest path problem with resource constraints: application to some vehicle routing problems. Networks 2004;44(3):216–29.
[49] Kohl N, Desrosiers J, Madsen OBG, Solomon MM, Soumis F. 2-path cuts for the vehicle routing problem with time windows. Transportation Science 1999;33(1):101–16.
[50] Gehring H, Homberger J. A parallel two-phase metaheuristic for routing problems with time windows. Asia-Pacific Journal of Operational Research 2001;18:35–47.
[51] Bouthillier AL, Crainic TG. A cooperative parallel meta-heuristic for the vehicle routing problem with time windows. Computers and Operations Research 2005;32:1685–708.
[52] Bräysy O, Hasle G, Dullaert W. A multi-start local search algorithm for the vehicle routing problem with time windows. European Journal of Operational Research 2004;159:586–605.
[53] Christofides N, Mingozzi A, Toth P. The vehicle routing problem. In: Christofides N et al., Mingozzi A, Toth P, Sandi C, editors. Combinatorial optimization. New York: Wiley; 1979. p. 315–38 [chapter 11].
[54] Toth P, Vigo D. The granular tabu search and its application to the vehicle-routing problem. INFORMS Journal on Computing 2003;15(4): 333–46.
[55] Ergun O, Orlin JB, Steele-Feldman A. Creating very large scale neighborhoods out of smaller ones by compounding moves: a study on the vehicle routing problem. Working Paper, Massachusetts Institute of Technology; 2003.
[56] Prins C. A simple and effective evolutionary algorithm for the vehicle routing problem. Computers and Operations Research 2004;31: 1985–2002.
[57] Tarantilis C, Kiranoudis C. Boneroute: an adaptive memory-based method for effective fleet management. Annals of Operations Research 2002;115:227–41.
[58] Berger J, Barkaoui M. A new hybrid genetic algorithm for the capacitated vehicle routing problem. Journal of the Operational Research Society 2003;54:1254–62.
[59] Reimann M, Doerner K, Hartl R. D-ants: savings based ants divide and conquer the vrp. Computers and Operations Research 2004;31:563–91.
[60] Lambert D, Cooper M. Issues in supply chain management. Marketing Management 2000;29:65–83.