

A Variable Neighborhood Search for the Capacitated Arc Routing Problem with Intermediate Facilities

Michael Polacek⁽¹⁾, Karl F. Doerner⁽¹⁾,
Richard F. Hartl⁽¹⁾, Vittorio Maniezzo⁽²⁾

(1) Department of Business Administration, University of Vienna, Bruenner
Strasse 72, 1210 Vienna, Austria

`{Michael.Polacek, Karl.Doerner, Richard.Hartl}@univie.ac.at`

(2) Department of Computer Science, University of Bologna, Mura Anteo
Zamboni, 7, 40100 Bologna, Italy
`vittorio.maniezzo@unibo.it`

Abstract

The capacitated arc routing problem (CARP) focuses on servicing edges of an undirected network graph. A wide spectrum of applications like mail delivery, waste collection or street maintenance outlines the relevance of this problem. A realistic variant of the CARP arises from the need of intermediate facilities (IFs) to load up or unload the service vehicle and from tour length restrictions. The proposed Variable Neighborhood Search (VNS) is a simple and robust solution technique which tackles the basic problem as well as its extensions. Particularly, it outperforms all known heuristics on four sets of benchmark instances.

Keywords: Capacitated Arc Routing Problem, Variable Neighborhood Search, Intermediate Facilities, Tour Length Restriction

1 Introduction

Routing problems are foremost problems of computational logistics, and of combinatorial optimization as a whole. Being of both practical and theoretical relevance, they have provided a primary arena for validating new metaheuristics and determining comparative efficiency. This was true also for Variable Neighborhood Search (VNS) introduced by Mladenovics and Hansen in 1997 [30]. VNS systems have proven their effectiveness on a number of variants of vehicle routing problems (VRP), e.g., the vehicle routing problem with time windows [9], the vehicle routing problem with multiple depots and time windows [31], and in real world routing problems [32]. The capacitated arc routing problem (CARP)

is one of the prototypical routing problems, asking a fleet of vehicles to service a set of clients which are distributed on the arcs of a road network. Every CARP instance can be transformed into an equivalent VRP instance with a number of nodes equal to the number of required arcs of the original CARP graph in case of directed instances [28], to twice that number in case of fully undirected instances [3], or to a combination thereof [29]. However, like most other efficient CARP algorithms we do not use this transformation.

Several applications of real-world relevance can be modelled as CARP, foremost among them are urban waste collection, mail collection or delivery, snow removal, street sweeping. The CARP was originally proposed as such by Golden and Wong [18], and given its actual interest many researches have studied it. Dror [11] collected a significant number of applications of CARP variants and of corresponding solution methodologies. For a survey the reader is also referred to [2].

A realistic variant of the CARP with respect to urban waste collection or snow removal and winter gritting is the Capacitated Arc Routing Problem with Intermediate Facilities (CARPIF). This problem was introduced by Ghiani et al. [16]. Ghiani et al. have also introduced the capacitated arc routing problem with intermediate facilities and tour length restrictions (CLARPIF) a tour length restricted version of the CARPIF [17]. In most cities the garbage collecting vehicles are assigned to a depot whereas the garbage has to be dumped at waste incinerating plants or special dump sites. Such a system was described by Wirasinghe and Waters [36] for the city of Calgary in Canada. This is also the case in snow plowing - the snow is dumped very often in some rivers at special dump sites. In this problem types we have demand collections. There exists also problems where we have demand deliveries. This is especially in the situation of winter gritting or street cleaning. Here we have the situation that intermediate facilities are located to pickup gritting material, salt or sand in the case of winter gritting or water and cleaning chemicals for the street cleaning situation.

In the literature, several exact and heuristic approaches have been proposed for the CARP. Among the exact techniques Hirabayashi et al. [24] proposed a Branch and Bound algorithm and Belenguer and Benavent [4] proposed valid inequalities for this problem. Recently, Wøhlk developed new lower bounds for the classical CARP [37]. Lower and upper bounds on the mixed CARP are presented by Belenguer et al. [7]. In the mixed CARP the connections between nodes can be arcs or edges. Arcs are oriented edges, whereas edges have no orientation.

However, exact techniques can seldom cope with the dimension of real-world instances. This calls for heuristic and metaheuristic approaches: among the most recent proposals we mention the tabu search of Belenguer et al. [5], the tabu search of Hertz et al. [22], the variable neighborhood descend of the same authors [23], the guided local search of Beullens et al. [8], the scatter search of Greistorfer [19], and the genetic algorithm of Lacomme et al. [27]. A memetic algorithm developed by Lacomme et al. provides the best solution quality for the standard benchmark CARP instances [25] so far.

As for more real-world oriented researches, we refer to the work of Amberg et al. [1], who studied the M-CARP, that is a multi depot CARP with heterogeneous vehicle fleet.

Xin [38] implemented a tabu search algorithm within a decision support system for waste collection in rural areas based on a simple augment-merge heuristic of Snizek et al. [33]. Maniezzo [28] developed a VNS for urban solid waste collection for an Italian town.

The waste collecting bins have to be emptied in regular periods, therefore the problem of waste collection can also be considered as a periodic arc routing problem. First works on periodic CARPs were published by Lacomme et al. [26] and by Chu et al. [10]. Fleury et al. [14] considered a variant of the CARP where the amount of waste which has to be collected is stochastic.

The paper is organized as follows. Section 2 describes the CARP and the adaption of intermediate facilities, Section 3 illustrates the proposed solution procedure, Section 4 reports about the obtained computational results, whereas Section 5 summarizes the results and the relevant aspects of the applied algorithm.

2 Problem Description

The CARP can be defined as follows. A connected and undirected graph $G = (V, E)$ representing a road network and a vehicle fleet are given. Each edge $(ij) \in E$ has a traversal cost c_{ij} . Furthermore, the subset R of E contains all required edges which can be associated with a customer with demand q_{ij} . All vehicles have identical capacity Q and are stationed at a depot, being one of the nodes of the graph G . The CARP asks for designing the vehicle routes, so that each vehicle starts and ends at the depot, each customer is serviced by one and only one vehicle, the sum of the requests of the customers serviced by a vehicle does not exceed the vehicle capacity, and the total travel cost is minimized.

In the CARPIF, the set V contains a subset I of intermediate facilities (IFs), possibly including the depot. The aim is to determine a least-cost tour of all edges of R such that the total vehicle load at any time, does not exceed Q . In other words, starting from the depot, the vehicle traverses edges of E , collects demands on required edges, and visits IFs to unload. The total demand accumulated between the depot and the first IF or between two successive IFs may never exceed Q . If the last IF on the tour is not the depot, then the final chain between that facility and the depot does not have any positive demand and is therefore unconstrained (see [16]). A variant of the CARPIF is the CLARPIF, where the length of any route may not exceed a preset bound L .

The CARP is closely akin to the VRP, and it is NP-hard [15], even in the single-vehicle case called Rural Postman Problem (RPP).

3 Solution Techniques

To solve the problem described in the previous section a metaheuristic called Variable Neighborhood Search (VNS) is applied. The basic scheme of VNS was proposed by Mladenović and Hansen in [30]. Further principles for solving combinatorial optimization problems and applications were introduced in [20] and [21].

The basic idea is a systematic change of neighborhood within a local search. Here, several neighborhood structures are used instead of a single one, as it is generally the case in many local search implementations. Furthermore, the systematic change of neighborhood is applied during both a descent phase and an exploration phase, allowing to get out of local optima.

More precisely, VNS follows the concept of exploring increasingly distant neighborhoods of the current solution. The search jumps from its current point in the solution space to a new one if and only if an improvement has been made. "In this way often favorable characteristics of the incumbent solution, e. g., that many variables are already at their optimal value, will be kept and used to obtain promising neighboring solutions. Moreover, a local search routine is applied repeatedly to get from these neighboring solutions to local optima."¹

The steps of the basic VNS are shown in Figure 1. Here, $N_\kappa (\kappa = 1, \dots, \kappa_{max})$ is a finite set of pre-selected neighborhood structures. The stopping condition may be, e. g., maximum CPU time allowed, maximum number of iterations or maximum number of iterations between two improvements.

Initialization. Select the set of neighborhood structures $N_\kappa (\kappa = 1, \dots, \kappa_{max})$, that will be used in the search; find an initial solution x ; choose a stopping condition;

Repeat the following until the stopping condition is met:

1. Set $\kappa \leftarrow 1$;
2. Repeat the following steps until $\kappa = \kappa_{max}$:
 - (a) *Shaking.* Generate a point x' at random from κ^{th} neighborhood of x ($x' \in N_\kappa(x)$);
 - (b) *Local search.* Apply some local search method with x' as initial solution; denote with x'' the so obtained local optimum;
 - (c) *Move or not.* If this local optimum x'' is better than the incumbent x , move there ($x \leftarrow x''$), and continue the search with N_1 ($\kappa \leftarrow 1$); otherwise, set $\kappa \leftarrow \kappa + 1$;

Figure 1: Steps of the basic VNS (c. f. [21])

The basic VNS consists of both a stochastic component, i. e., the randomized selection of a neighbor in the shaking phase, and a deterministic component,

¹see [21], p. 450

that is the application of a local search in each iteration. Finally, the solution obtained is compared to the incumbent one and will be accepted as new starting point if an improvement was made, otherwise it will be rejected. Thus, the procedure is a descent, first improvement method with randomization. However, as pointed out in [21], it could be transformed into a descent-ascent method without much additional effort. Thereby x is also set to x'' with some probability, even if the solution is worse than the incumbent.

Below, the implementation of the main parts of the VNS for solving the CARP and its variants is described. The description includes the building of an initial solution, the shaking phase with the proper exchange operator, the local search method, and the acceptance decision in the *Move or not* phase.

3.1 Initial Solution

The sole criterion for the initial solution used as a starting point generator for the VNS is to come up with a feasible solution. The fastest and simplest way to do so is to sequentially read in the required edges from the instance file and add them to the end of the current tour. However, if the insertion will exceed the capacity constraint or the depot node lies on the shortest path between the last node of the previous edge and the first node of the new edge, the latter one becomes the initial edge of a new tour.

Alternatively, we implemented the construction heuristic of Ulusoy [35]. Here one or more giant tours are generated which cover all mandatory edges. To obtain such a giant tour the graph has to be potentially extended by least cost edges so that the degree of every vertex is even. Afterwards the tour has to be split into feasible trips with respect to the capacity constraints. Therefore a directed auxiliary graph is built of all feasible subtours within the giant tour. The splitting can be done optimally by finding the shortest path of the auxiliary graph which starts and ends at the depot.

3.2 Shaking

The set of neighborhood structures used for shaking is the core of the VNS. To define a neighborhood of the current solution an appropriate function or operator must be specified. The main issue is that the neighborhood operator should sufficiently perturb the incumbent solution while still making sure that the new solution keeps important parts of the incumbent.

An operator which enables sufficient changes of a solution while preserving well composite sequences is called CROSS-exchange and was proposed in [34] for the Vehicle Routing Problem with Time Windows. Its effectiveness was also demonstrated in [31]. The main idea of this exchange is to take two segments of different routes and exchange them as illustrated in Figure 2. Hereby the orientation of the sequences keeps preserved.

In every shaking phase two routes are randomly chosen. By treating it as a special case it is allowed to select the same route twice. One of the routes can

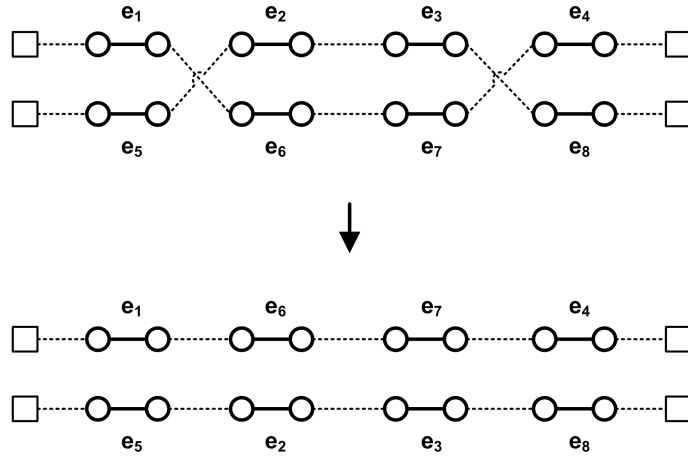


Figure 2: The CROSS-exchange operator applied to edges

also be a new empty route as well. After the selection of the routes the CROSS-exchange operator is applied to swap the sequences of successive serviced edges.

Let us denote the number of serviced edges of route i with T_i and the minimum number of selected edges with α_i . The length of the subsequence which gets swapped is uniformly distributed on the interval $[\alpha_i, \min\{T_i, \kappa\}]$. In order to permit more dramatic changes without having to choose a high κ_{max} , the upper bound of the interval is simply replaced by T_i if $\kappa = \kappa_{max}$. The latter case allows an exchange of all mandatory edges from one tour to the other. Furthermore, as it is done by default, setting $\alpha_1 = 1$ and $\alpha_2 = 0$ enables a simple move of edges.

The described application of the CROSS-exchange operator results in a strong bias towards smaller sequence lengths which focuses the search rather close to the incumbent solution whereas the possibility of large changes retains.

3.3 Local Search

A solution obtained through shaking is afterwards submitted to a local search procedure to come up with a local optimal solution. While the neighborhood operator was designed to preserve the orientation of the selected sequences, the local search has to compensate the missing functionality of systematically inverting the edges.

Starting with the first mandatory edge of a tour the algorithm inverts each serviced edge and calculates the cost reduction of the new shortest paths needed to reconnect the edge into the tour. In an analogous manner the inverting is done for all combinations of successive serviced edges of a tour. Optionally, the number of involved edges can be restricted to the value of parameter λ .

Another important issue related to algorithm effectiveness is that the local

search is applied only on a route basis. Thus, after each shaking only the two routes that have changed need to be re-optimized. Finally, as it is commonly done, the local search restarts instantly after an improving move was found.

3.4 Acceptance decision

After the shaking and the local search procedures have been performed, the solution thus obtained has to be compared to the incumbent solution to be able to decide whether or not to accept it.

The restricted use of the CROSS-exchange operator in the shaking phase makes it more difficult to leave a local optima. Therefore we use a modified acceptance decision which is based on Threshold Accepting ideas (c.f. [12]) to allow non-improving (ascending) moves. A move yielding an improvement is always accepted, while ascending moves are accepted as long as their objective value does not exceed a fixed threshold. This threshold is given by $\theta\%$ of the so far best found solution value. As proposed in Polacek et al. [31] ascending moves are only performed after a minimum number of σ iterations counted from the last accepted move.

4 Computational Analysis

In this section the VNS implementation proposed above will be analyzed and its performance on basic CARP will be compared to the results of the Memetic Algorithm (MA) described in [25]. For the CARPIF and CLARPIF its performance will be compared to the results of the CARP-based heuristic and the Tabu Search (TS) published in [16] and [17] respectively.

The VNS was implemented in C++ using elements of the Standard Template Library (STL). Experiments were performed on a Pentium IV processor with 3.6 GHz. In order to compare the runtimes with the MA of Lacomme et al., the VNS was additionally executed on a Pentium III processor with 933 MHz.

The first set of problem instances (val files) used for the analysis originates from Belenguer, Benavent and Cognata [5]. It consists of 34 instances with 24-50 nodes and 34-97 edges which all have to be serviced. The second set (egl files) contains more complex instances which originate from a winter gritting problem for the road network of the county of Lancashire (UK) proposed by Eglese [13]. Based on this data set Belenguer and Benavent [6] generated 24 problem instances with 77-140 nodes and 98-190 required and non-required edges.

For the CARPIF benchmark the first set of problem instances (val files) is reused as it is done in [16] by adding two IFs which are located at vertices $\lfloor |V|/2 \rfloor$ and $2\lfloor |V|/2 \rfloor$. The first 28 instances of this third set are also used by Ghiani et al. [17] to generate test examples for the CLARPIF. According to this the fourth set contains a tour length restriction L which was chosen in such a way that the number of routes is uniformly distributed between 5 and 20.

Because of the fact that the last two sets both originate from the first one all edges are again required.

To analyze the influence of the initial solution to the final solution quality we compared our simple sequential appending method with the more sophisticated process of Ulusoy described in [35] (see 3.1). In Table 1 and Table 2 results are presented according to the number of generated Ulusoy solutions for the val files and egl files respectively. More precisely the first row represents the results of our simple algorithm whereas the following rows contain the results of the Ulusoy method. The second column shows the time it took to generate the given number of solutions and the third column reports the runtimes for the VNS to obtain the final solution. Hereby the total time in the next column and the values of the previous ones are all measured in seconds. The last three columns focus on the achieved solution quality. So the relative percentage of deviation (RPD) compares the values of the initial solutions with the values of the final solutions obtained by the VNS. With respect to the Ulusoy values the best result of the given number of generated solutions was chosen as starting point. The results of the comparison outline two important issues. Firstly, an expedient use of the Ulusoy method can significantly reduce the total runtime. Secondly, the robustness of the VNS was demonstrated by achieving results of consistently high quality independently of the quality of the initial solution.

| Ulusoy Solutions | Ulusoy Time | VNS Time | Total Time | Initial Values | VNS Values | RPD |
|------------------|-------------|----------|------------|----------------|------------|---------|
| 0 | 0.00 | 1494.19 | 1494.19 | 16116 | 11717 | -27.30% |
| 1 | 0.97 | 1185.38 | 1186.35 | 13227 | 11716 | -11.43% |
| 10 | 1.02 | 1015.05 | 1016.02 | 12733 | 11715 | -7.99% |
| 100 | 1.52 | 1185.30 | 1186.81 | 12491 | 11716 | -6.21% |
| 1000 | 7.27 | 1016.14 | 1023.41 | 12307 | 11718 | -4.79% |
| 10000 | 64.59 | 1257.58 | 1322.17 | 12219 | 11718 | -4.10% |
| 100000 | 647.00 | 1092.30 | 1739.30 | 12107 | 11713 | -3.25% |

Table 1: Effect of initial solutions for the val files

| Ulusoy Solutions | Ulusoy Time | VNS Time | Total Time | Initial Values | VNS Values | RPD |
|------------------|-------------|----------|------------|----------------|------------|---------|
| 0 | 0.00 | 16362.49 | 16362.49 | 288581 | 235108 | -18.53% |
| 1 | 8.40 | 14001.55 | 14009.95 | 266699 | 235154 | -11.83% |
| 10 | 8.59 | 14964.90 | 14973.49 | 261392 | 235137 | -10.04% |
| 100 | 10.41 | 14260.28 | 14270.68 | 257789 | 235095 | -8.80% |
| 1000 | 28.30 | 13470.37 | 13498.67 | 255335 | 235069 | -7.94% |
| 10000 | 206.41 | 14365.35 | 14571.75 | 252646 | 235123 | -6.94% |
| 100000 | 1962.08 | 13234.66 | 15196.74 | 251147 | 235126 | -6.38% |

Table 2: Effect of initial solutions for the egl files

| File | E | V | LB | MA (1 GHz) | | VNS (3.6 GHz) | | | VNS (933 MHz) | | | Best | | | | |
|--------|----|----|-----|------------|-------|---------------|---------|--------|---------------|--------|--------|---------|--------|-------|-------|--------|
| | | | | Time | Value | Time | Avg. | RPD | Min. | RPD | Time | Avg. | RPD | MA | VNS | RPD |
| val1a | 24 | 39 | 173 | 0.00 | 173 | 0.06 | 173.0 | 0.00% | 173 | 0.00% | 0.42 | 173 | 0.00% | 173 | 173 | 0.00% |
| val1b | 24 | 39 | 173 | 8.02 | 173 | 1.37 | 173.0 | 0.00% | 173 | 0.00% | 5.74 | 173 | 0.00% | 173 | 173 | 0.00% |
| val1c | 24 | 39 | 235 | 28.67 | 245 | 0.64 | 245.0 | 0.00% | 245 | 0.00% | 1.64 | 245 | 0.00% | 245 | 245 | 0.00% |
| val2a | 24 | 34 | 227 | 0.05 | 227 | 0.21 | 227.0 | 0.00% | 227 | 0.00% | 0.9 | 227 | 0.00% | 227 | 227 | 0.00% |
| val2b | 24 | 34 | 259 | 0.22 | 259 | 6.65 | 259.0 | 0.00% | 259 | 0.00% | 8.11 | 259 | 0.00% | 259 | 259 | 0.00% |
| val2c | 24 | 34 | 455 | 21.76 | 457 | 16.66 | 457.0 | 0.00% | 457 | 0.00% | 13.51 | 457 | 0.00% | 457 | 457 | 0.00% |
| val3a | 24 | 35 | 81 | 0.05 | 81 | 0.06 | 81.0 | 0.00% | 81 | 0.00% | 0.24 | 81 | 0.00% | 81 | 81 | 0.00% |
| val3b | 24 | 35 | 87 | 0.00 | 87 | 2.43 | 87.0 | 0.00% | 87 | 0.00% | 2.71 | 87 | 0.00% | 87 | 87 | 0.00% |
| val3c | 24 | 35 | 137 | 28.23 | 138 | 25.33 | 138.0 | 0.00% | 138 | 0.00% | 4.49 | 138 | 0.00% | 138 | 138 | 0.00% |
| val4a | 41 | 69 | 400 | 0.72 | 400 | 3.35 | 400.0 | 0.00% | 400 | 0.00% | 5.13 | 400 | 0.00% | 400 | 400 | 0.00% |
| val4b | 41 | 69 | 412 | 1.21 | 412 | 3.15 | 412.0 | 0.00% | 412 | 0.00% | 6.68 | 412 | 0.00% | 412 | 412 | 0.00% |
| val4c | 41 | 69 | 428 | 19.11 | 428 | 39.55 | 428.0 | 0.00% | 428 | 0.00% | 99.15 | 430.4 | 0.56% | 428 | 428 | 0.00% |
| val4d | 41 | 69 | 520 | 103.26 | 541 | 82.15 | 531.2 | -1.81% | 530 | -2.03% | 88.39 | 530.5 | -1.94% | 530 | 530 | 0.00% |
| val5a | 34 | 65 | 423 | 1.86 | 423 | 2.50 | 423.0 | 0.00% | 423 | 0.00% | 4.22 | 423 | 0.00% | 423 | 423 | 0.00% |
| val5b | 34 | 65 | 446 | 1.04 | 446 | 5.00 | 446.0 | 0.00% | 446 | 0.00% | 3.13 | 446 | 0.00% | 446 | 446 | 0.00% |
| val5c | 34 | 65 | 469 | 101.01 | 474 | 8.49 | 474.0 | 0.00% | 474 | 0.00% | 10.83 | 474 | 0.00% | 474 | 474 | 0.00% |
| val5d | 34 | 65 | 571 | 90.74 | 581 | 278.93 | 579.8 | -0.21% | 575 | -1.03% | 130.3 | 579 | -0.34% | 581 | 575 | -1.03% |
| val6a | 31 | 50 | 223 | 0.17 | 223 | 0.49 | 223.0 | 0.00% | 223 | 0.00% | 0.79 | 223 | 0.00% | 223 | 223 | 0.00% |
| val6b | 31 | 50 | 231 | 67.34 | 233 | 1.79 | 233.0 | 0.00% | 233 | 0.00% | 11.41 | 233 | 0.00% | 233 | 233 | 0.00% |
| val6c | 31 | 50 | 311 | 52.23 | 317 | 25.61 | 317.0 | 0.00% | 317 | 0.00% | 9.7 | 317 | 0.00% | 317 | 317 | 0.00% |
| val7a | 40 | 66 | 279 | 1.97 | 279 | 0.93 | 279.0 | 0.00% | 279 | 0.00% | 2.1 | 279 | 0.00% | 279 | 279 | 0.00% |
| val7b | 40 | 66 | 283 | 0.44 | 283 | 0.21 | 283.0 | 0.00% | 283 | 0.00% | 0.67 | 283 | 0.00% | 283 | 283 | 0.00% |
| val7c | 40 | 66 | 333 | 101.17 | 334 | 14.20 | 334.0 | 0.00% | 334 | 0.00% | 37.54 | 334 | 0.00% | 334 | 334 | 0.00% |
| val8a | 30 | 63 | 386 | 0.66 | 386 | 1.13 | 386.0 | 0.00% | 386 | 0.00% | 1.89 | 386 | 0.00% | 386 | 386 | 0.00% |
| val8b | 30 | 63 | 395 | 9.95 | 395 | 1.56 | 395.0 | 0.00% | 395 | 0.00% | 4.62 | 395 | 0.00% | 395 | 395 | 0.00% |
| val8c | 30 | 63 | 517 | 71.46 | 527 | 269.18 | 522.0 | -0.95% | 521 | -1.14% | 140.8 | 525.8 | -0.23% | 527 | 521 | -1.14% |
| val9a | 50 | 92 | 323 | 18.29 | 323 | 13.23 | 323.0 | 0.00% | 323 | 0.00% | 43.23 | 323 | 0.00% | 323 | 323 | 0.00% |
| val9b | 50 | 92 | 326 | 29.39 | 326 | 10.17 | 326.0 | 0.00% | 326 | 0.00% | 23.87 | 326 | 0.00% | 326 | 326 | 0.00% |
| val9c | 50 | 92 | 332 | 71.19 | 332 | 51.63 | 332.0 | 0.00% | 332 | 0.00% | 42.84 | 332.4 | 0.12% | 332 | 332 | 0.00% |
| val9d | 50 | 92 | 382 | 211.13 | 391 | 88.84 | 390.8 | -0.05% | 389 | -0.51% | 49.46 | 391 | 0.00% | 391 | 389 | -0.51% |
| val10a | 50 | 97 | 428 | 25.48 | 428 | 63.70 | 428.4 | 0.09% | 428 | 0.00% | 41.95 | 428.2 | 0.05% | 428 | 428 | 0.00% |
| val10b | 50 | 97 | 436 | 4.67 | 436 | 7.23 | 436.6 | 0.14% | 436 | 0.00% | 23.61 | 437.2 | 0.28% | 436 | 436 | 0.00% |
| val10c | 50 | 97 | 446 | 17.30 | 446 | 120.60 | 447.0 | 0.22% | 446 | 0.00% | 60.38 | 447.1 | 0.25% | 446 | 446 | 0.00% |
| val10d | 50 | 97 | 524 | 215.04 | 530 | 347.16 | 526.9 | -0.58% | 526 | -0.75% | 112.2 | 529.5 | -0.09% | 528 | 525 | -0.57% |
| | | | | 1303.84 | 11734 | 1494.19 | 11716.7 | -0.09% | 11705 | -0.16% | 992.60 | 11725.1 | -0.04% | 11721 | 11704 | -0.10% |

Table 3: Computational results for the CARP val instances

| File | E | V | LB | MA (1 GHz) | | VNS (3.6 GHz) | | | VNS (933 MHz) | | | Best | | | | |
|----------|-----|-----|---------|------------|---------|---------------|---------|--------|---------------|--------|----------|---------|--------|--------|--------|--------|
| | | | | Time | Value | Time | Avg. | RPD | Min. | RPD | Time | Avg. | RPD | MA | VNS | RPD |
| egl-e1-A | 77 | 98 | 3515 | 74.3 | 3548 | 5.1 | 3548.0 | 0.00% | 3548 | 0.00% | 43.63 | 3556.8 | 0.25% | 3548 | 3548 | 0.00% |
| egl-e1-B | 77 | 98 | 4436 | 69.5 | 4498 | 56.1 | 4522.2 | 0.54% | 4498 | 0.00% | 121.93 | 4525.7 | 0.62% | 4498 | 4498 | 0.00% |
| egl-e1-C | 77 | 98 | 5453 | 71.2 | 5595 | 648.4 | 5608.0 | 0.23% | 5595 | 0.00% | 393.14 | 5615.4 | 0.36% | 5595 | 5595 | 0.00% |
| egl-e2-A | 77 | 98 | 4994 | 152.6 | 5018 | 360.3 | 5023.8 | 0.12% | 5018 | 0.00% | 372.3 | 5033 | 0.30% | 5018 | 5018 | 0.00% |
| egl-e2-B | 77 | 98 | 6249 | 153.4 | 6340 | 691.8 | 6335.4 | -0.07% | 6321 | -0.30% | 516.11 | 6341.5 | 0.02% | 6340 | 6317 | -0.36% |
| egl-e2-C | 77 | 98 | 8114 | 129.6 | 8415 | 431.7 | 8355.9 | -0.70% | 8335 | -0.95% | 469.72 | 8399.4 | -0.19% | 8395 | 8335 | -0.71% |
| egl-e3-A | 77 | 98 | 5869 | 242.0 | 5898 | 196.0 | 5898.0 | 0.00% | 5898 | 0.00% | 225.05 | 5906.1 | 0.14% | 5898 | 5898 | 0.00% |
| egl-e3-B | 77 | 98 | 7646 | 255.4 | 7822 | 474.0 | 7806.4 | -0.20% | 7775 | -0.60% | 450.01 | 7808.5 | -0.17% | 7816 | 7775 | -0.52% |
| egl-e3-C | 77 | 98 | 10019 | 206.4 | 10433 | 552.0 | 10322.3 | -1.06% | 10292 | -1.35% | 556.88 | 10384.8 | -0.46% | 10369 | 10292 | -0.74% |
| egl-e4-A | 77 | 98 | 6372 | 291.9 | 6461 | 489.0 | 6459.4 | -0.02% | 6446 | -0.23% | 588.55 | 6477.2 | 0.25% | 6461 | 6446 | -0.23% |
| egl-e4-B | 77 | 98 | 8809 | 312.9 | 9021 | 493.6 | 9016.3 | -0.05% | 9004 | -0.19% | 430.59 | 9026.2 | 0.06% | 9021 | 8996 | -0.28% |
| egl-e4-C | 77 | 98 | 11276 | 252.4 | 11779 | 503.7 | 11750.1 | -0.25% | 11652 | -1.08% | 187.16 | 11792.5 | 0.11% | 11779 | 11618 | -1.37% |
| egl-s1-A | 140 | 190 | 4992 | 208.6 | 5018 | 182.3 | 5018.0 | 0.00% | 5018 | 0.00% | 388.63 | 5028.6 | 0.21% | 5018 | 5018 | 0.00% |
| egl-s1-B | 140 | 190 | 6201 | 208.8 | 6435 | 363.6 | 6388.0 | -0.73% | 6388 | -0.73% | 369.24 | 6409.2 | -0.40% | 6435 | 6388 | -0.73% |
| egl-s1-C | 140 | 190 | 8310 | 165.6 | 8518 | 361.8 | 8518.2 | 0.00% | 8518 | 0.00% | 391.78 | 8520.7 | 0.03% | 8518 | 8518 | 0.00% |
| egl-s2-A | 140 | 190 | 9780 | 874.4 | 9995 | 800.5 | 9997.9 | 0.03% | 9944 | -0.51% | 498.87 | 10034.3 | 0.39% | 9995 | 9895 | -1.00% |
| egl-s2-B | 140 | 190 | 12886 | 760.5 | 13174 | 809.3 | 13176.0 | 0.02% | 13167 | -0.05% | 517.55 | 13217 | 0.33% | 13174 | 13100 | -0.56% |
| egl-s2-C | 140 | 190 | 16221 | 746.9 | 16795 | 673.5 | 16551.6 | -1.45% | 16491 | -1.81% | 411.97 | 16628.3 | -0.99% | 16715 | 16425 | -1.73% |
| egl-s3-A | 140 | 190 | 10025 | 1070.5 | 10296 | 749.7 | 10291.2 | -0.05% | 10259 | -0.36% | 416.7 | 10325 | 0.28% | 10296 | 10221 | -0.73% |
| egl-s3-B | 140 | 190 | 13554 | 1064.0 | 14053 | 572.5 | 13829.2 | -1.59% | 13751 | -2.15% | 414.2 | 13884.4 | -1.20% | 14028 | 13682 | -2.47% |
| egl-s3-C | 140 | 190 | 16969 | 874.3 | 17297 | 721.0 | 17327.9 | 0.18% | 17299 | 0.01% | 540.29 | 17355.4 | 0.34% | 17297 | 17259 | -0.22% |
| egl-s4-A | 140 | 190 | 12027 | 1537.6 | 12442 | 681.0 | 12440.4 | -0.01% | 12375 | -0.54% | 478.01 | 12469.6 | 0.22% | 12442 | 12292 | -1.21% |
| egl-s4-B | 140 | 190 | 15933 | 1430.3 | 16531 | 424.8 | 16410.3 | -0.73% | 16353 | -1.08% | 514.32 | 16476.8 | -0.33% | 16531 | 16321 | -1.27% |
| egl-s4-C | 140 | 190 | 20179 | 1495.0 | 20832 | 835.8 | 20731.5 | -0.48% | 20640 | -0.92% | 513.68 | 20791.2 | -0.20% | 20832 | 20582 | -1.20% |
| | | | 12647.7 | 236214 | 12077.3 | 235326.0 | -0.26% | 234585 | -0.53% | 9810.3 | 236007.6 | 0.00% | 236019 | 234037 | -0.64% | |

Table 4: Computational results for the CARP egl instances

| File | E | V | Ghiani et al. | | | VNS | | | VNS ($\lambda = 2$) | | | Best | | |
|--------|----|----|---------------|--------|---------|---------|--------|---------|-----------------------|--------|-------|--------|-----|-----|
| | | | Time | UB_2 | Time | Avg. | RPD | Time | Avg. | RPD | VNS | RPD | VNS | RPD |
| val1a | 24 | 39 | 46 | 183 | 0.25 | 183.0 | 0.00% | 0.36 | 183.0 | 0.00% | 183 | 0.00% | | |
| val1b | 24 | 39 | 65 | 183 | 0.65 | 183.0 | 0.00% | 1.71 | 183.0 | 0.00% | 183 | 0.00% | | |
| val1c | 24 | 39 | 250 | 218 | 211.11 | 220.7 | 1.24% | 212.24 | 215.6 | -1.10% | 215 | -1.38% | | |
| val2a | 24 | 34 | 39 | 247 | 0.24 | 247.0 | 0.00% | 1.14 | 247.0 | 0.00% | 247 | 0.00% | | |
| val2b | 24 | 34 | 43 | 247 | 0.20 | 247.0 | 0.00% | 2.78 | 247.0 | 0.00% | 247 | 0.00% | | |
| val2c | 24 | 34 | 575 | 340 | 128.57 | 343.3 | 0.97% | 171.35 | 331.0 | -2.65% | 330 | -2.94% | | |
| val3a | 24 | 35 | 37 | 85 | 0.20 | 85.0 | 0.00% | 1.63 | 85.0 | 0.00% | 85 | 0.00% | | |
| val3b | 24 | 35 | 41 | 91 | 0.37 | 85.0 | -6.59% | 2.82 | 85.0 | -6.59% | 85 | -6.59% | | |
| val3c | 24 | 35 | 122 | 96 | 42.92 | 94.0 | -2.08% | 6.02 | 91.0 | -5.21% | 91 | -5.21% | | |
| val4a | 41 | 69 | 269 | 406 | 4.69 | 406.0 | 0.00% | 3.08 | 406.0 | 0.00% | 406 | 0.00% | | |
| val4b | 41 | 69 | 403 | 406 | 6.84 | 406.0 | 0.00% | 7.00 | 406.0 | 0.00% | 406 | 0.00% | | |
| val4c | 41 | 69 | 603 | 410 | 46.94 | 406.0 | -0.98% | 116.84 | 406.0 | -0.98% | 406 | -0.98% | | |
| val4d | 41 | 69 | 1537 | 460 | 214.75 | 444.8 | -3.30% | 291.23 | 436.9 | -5.02% | 434 | -5.65% | | |
| val5a | 34 | 65 | 243 | 456 | 9.12 | 432.0 | -5.26% | 2.67 | 432.0 | -5.26% | 432 | -5.26% | | |
| val5b | 34 | 65 | 467 | 455 | 15.42 | 432.0 | -5.05% | 11.85 | 432.0 | -5.05% | 432 | -5.05% | | |
| val5c | 34 | 65 | 857 | 434 | 59.06 | 437.6 | 0.83% | 136.73 | 434.0 | 0.00% | 434 | 0.00% | | |
| val5d | 34 | 65 | 1358 | 512 | 177.69 | 521.0 | 1.76% | 397.78 | 510.0 | -0.39% | 496 | -3.13% | | |
| val6a | 31 | 50 | 158 | 230 | 1.21 | 229.0 | -0.43% | 2.68 | 229.0 | -0.43% | 229 | -0.43% | | |
| val6b | 31 | 50 | 339 | 247 | 15.40 | 244.0 | -1.21% | 208.93 | 242.4 | -1.86% | 241 | -2.43% | | |
| val6c | 31 | 50 | 791 | 369 | 24.31 | 346.0 | -6.23% | 11.55 | 346.0 | -6.23% | 346 | -6.23% | | |
| val7a | 40 | 66 | 286 | 287 | 3.57 | 287.0 | 0.00% | 1.59 | 287.0 | 0.00% | 287 | 0.00% | | |
| val7b | 40 | 66 | 354 | 295 | 2.42 | 295.0 | 0.00% | 3.73 | 295.0 | 0.00% | 295 | 0.00% | | |
| val7c | 40 | 66 | 1049 | 371 | 93.65 | 363.4 | -2.05% | 159.07 | 361.2 | -2.64% | 361 | -2.70% | | |
| val8a | 30 | 63 | 210 | 403 | 2.36 | 399.0 | -0.99% | 0.98 | 399.0 | -0.99% | 399 | -0.99% | | |
| val8b | 30 | 63 | 319 | 399 | 5.87 | 399.0 | 0.00% | 1.62 | 399.0 | 0.00% | 399 | 0.00% | | |
| val8c | 30 | 63 | 1273 | 442 | 178.74 | 432.7 | -2.10% | 77.68 | 424.0 | -4.07% | 424 | -4.07% | | |
| val9a | 50 | 92 | 1126 | 342 | 9.12 | 336.0 | -1.75% | 16.14 | 336.0 | -1.75% | 336 | -1.75% | | |
| val9b | 50 | 92 | 658 | 337 | 26.89 | 336.0 | -0.30% | 41.29 | 336.0 | -0.30% | 336 | -0.30% | | |
| val9c | 50 | 92 | 648 | 352 | 86.50 | 336.0 | -4.55% | 71.05 | 336.0 | -4.55% | 336 | -4.55% | | |
| val9d | 50 | 92 | 2873 | 413 | 363.52 | 402.2 | -2.62% | 339.43 | 397.3 | -3.80% | 393 | -4.84% | | |
| val10a | 50 | 97 | 686 | 450 | 21.13 | 433.0 | -3.78% | 125.43 | 433.0 | -3.78% | 433 | -3.78% | | |
| val10b | 50 | 97 | 483 | 450 | 38.82 | 433.0 | -3.78% | 100.27 | 433.0 | -3.78% | 433 | -3.78% | | |
| val10c | 50 | 97 | 700 | 456 | 163.80 | 438.6 | -3.82% | 211.13 | 438.0 | -3.95% | 435 | -4.61% | | |
| val10d | 50 | 97 | 4739 | 522 | 201.07 | 507.5 | -2.78% | 351.27 | 501.8 | -3.87% | 496 | -4.98% | | |
| | | | 23647 | 11594 | 2157.40 | 11390.8 | -1.61% | 3091.07 | 11324.2 | -2.18% | 11291 | -2.40% | | |

Table 5: Computational results for the CARPIF instances

| File | E | V | L | TS | | | VNS | | | Best | | |
|-------|----|----|----|---------|--------|-------|---------|---------|-------|---------|-------|---------|
| | | | | Time | UB_3 | Time | Avg. | RPD | Min. | RPD | VNS | RPD |
| val1a | 24 | 39 | 42 | 199 | 275 | 0.4 | 229.0 | -16.73% | 229 | -16.73% | 229 | -16.73% |
| val1b | 24 | 39 | 42 | 295 | 275 | 0.5 | 229.0 | -16.73% | 229 | -16.73% | 229 | -16.73% |
| val1c | 24 | 39 | 42 | 397 | 285 | 39.1 | 259.4 | -8.98% | 259 | -9.12% | 259 | -9.12% |
| val2a | 24 | 34 | 74 | 253 | 504 | 0.0 | 434.0 | -13.89% | 434 | -13.89% | 434 | -13.89% |
| val2b | 24 | 34 | 74 | 231 | 504 | 0.0 | 434.0 | -13.89% | 434 | -13.89% | 434 | -13.89% |
| val2c | 24 | 34 | 74 | 303 | 528 | 0.6 | 488.0 | -7.58% | 488 | -7.58% | 488 | -7.58% |
| val3a | 24 | 35 | 27 | 136 | 126 | 5.5 | 120.0 | -4.76% | 120 | -4.76% | 120 | -4.76% |
| val3b | 24 | 35 | 27 | 118 | 126 | 8.7 | 120.0 | -4.76% | 120 | -4.76% | 120 | -4.76% |
| val3c | 24 | 35 | 27 | 219 | 127 | 11.6 | 126.0 | -0.79% | 126 | -0.79% | 126 | -0.79% |
| val4a | 41 | 69 | 80 | 1534 | 627 | 54.0 | 536.4 | -14.45% | 533 | -14.99% | 533 | -14.99% |
| val4b | 41 | 69 | 80 | 1162 | 627 | 12.0 | 535.0 | -14.67% | 533 | -14.99% | 533 | -14.99% |
| val4c | 41 | 69 | 80 | 929 | 627 | 83.4 | 533.5 | -14.91% | 533 | -14.99% | 533 | -14.99% |
| val4d | 41 | 69 | 80 | 874 | 701 | 64.3 | 538.0 | -23.25% | 538 | -23.25% | 538 | -23.25% |
| val5a | 34 | 65 | 75 | 4797 | 1039 | 27.9 | 881.4 | -15.17% | 879 | -15.40% | 879 | -15.40% |
| val5b | 34 | 65 | 75 | 3976 | 1039 | 49.4 | 881.4 | -15.17% | 879 | -15.40% | 879 | -15.40% |
| val5c | 34 | 65 | 75 | 3501 | 1039 | 36.3 | 880.6 | -15.25% | 879 | -15.40% | 879 | -15.40% |
| val5d | 34 | 65 | 75 | 1629 | 1153 | 37.6 | 881.0 | -23.59% | 879 | -23.76% | 879 | -23.76% |
| val6a | 31 | 50 | 53 | 816 | 388 | 2.6 | 343.0 | -11.60% | 343 | -11.60% | 343 | -11.60% |
| val6b | 31 | 50 | 53 | 781 | 363 | 5.4 | 343.0 | -5.51% | 343 | -5.51% | 343 | -5.51% |
| val6c | 31 | 50 | 53 | 239 | 438 | 3.1 | 367.0 | -16.21% | 367 | -16.21% | 367 | -16.21% |
| val7a | 40 | 66 | 43 | 1080 | 461 | 0.3 | 444.0 | -3.69% | 444 | -3.69% | 444 | -3.69% |
| val7b | 40 | 66 | 43 | 850 | 461 | 0.3 | 444.0 | -3.69% | 444 | -3.69% | 444 | -3.69% |
| val7c | 40 | 66 | 43 | 856 | 461 | 0.3 | 444.0 | -3.69% | 444 | -3.69% | 444 | -3.69% |
| val8a | 30 | 63 | 71 | 1470 | 664 | 34.7 | 547.4 | -17.56% | 545 | -17.92% | 545 | -17.92% |
| val8b | 30 | 63 | 71 | 1204 | 664 | 28.7 | 548.4 | -17.41% | 545 | -17.92% | 545 | -17.92% |
| val8c | 30 | 63 | 71 | 574 | 716 | 43.9 | 548.1 | -23.45% | 547 | -23.60% | 547 | -23.60% |
| val9b | 50 | 92 | 46 | 4303 | 774 | 172.0 | 566.3 | -26.83% | 564 | -27.13% | 546 | -29.46% |
| val9d | 50 | 92 | 46 | 2026 | 774 | 128.9 | 567.8 | -26.64% | 564 | -27.13% | 546 | -29.46% |
| | | | | 34752.0 | 15766 | 851.6 | 13269.7 | -13.60% | 13242 | -13.73% | 13206 | -13.90% |

Table 6: Computational results for the CLARPIF instances

To further underline the robustness of the applied VNS we used a standard parameter setting for all four classes of benchmark instances. Here we set the neighborhood parameter $\kappa_{max} = 6$, the ascending move threshold $\theta = 120\%$ and the idle iteration parameter $\sigma = 10^6$. The optional restriction parameter λ in the local search was not used by default. The usage of only three parameters in the standard setting also indicates the simplicity of the algorithm.

The first two benchmark sets cover the basic CARP and results are presented in Table 3 and Table 4 for the val files and the egl files respectively. Hereby the results are compared to the so far best known solutions obtained by the MA of Lacomme, Prins and Ramdane-Cherif in [25].

The first three columns describe each instance by stating a name, the number of edges and the number of vertices. Additionally, a lower bound is given for every problem instance. The next two columns report the results of the MA whereas the subsequent section provides results of 10 separate runs of the VNS. Each run was performed on a Pentium IV processor with 3.6 GHz and the maximal runtime was set to 10 minutes. The section starts with the average times needed to find the best solutions within the predefined runtime. This is followed by the so obtained average values and the RPD between them and the results of the MA. As an additional information we provided the minimal objective value of the best found solutions of these 10 runs. In order to make the runtimes comparable to the MA results the next section contains data of VNS runs performed on a Pentium III processor with 933 MHz. This type of processor can be considered to be equivalent to the Pentium III processor with 1 GHz in [25]. For this comparable runs the maximal executing time was set to 5 minutes. We complete the tables with a comparison of the best found solutions of both methods. Finally, the last row shows the sum of runtimes and objective values and the averages of the RPDs.

For both benchmark sets it could be shown that the VNS outperforms even the excellent results of the so far best known heuristic method. For the instances of Belenguer et al. [6] 4 new best solutions were found and all other 30 best known solutions were reproduced. Hereby on average better results could be found by being about 20% faster. The same was true for the instances originated by Eglese. However 17 new best solutions were found while for the remaining 7 instances the best known solutions were reproduced. An important aspect is that even for the more complex instances of the egl files the VNS shows a better scaling according to the runtimes. In both cases the Ulusoy procedure used to generate better initial solutions and thereby reducing the runtime of the VNS was not used.

Table 5 contains the results for the CARPIF instances. Two IFs were added to each problem file where in the case of collecting goods the vehicles have to unload all items. Hence, only unloaded vehicles can return to the depot. The first three columns provide again information about the problem instances. Then the table provides the best results obtained by Ghiani et al. [16]. This includes the runtime performed on a Sun Ultra-10 station with 300 MHz as well as the upper bound results achieved with a CARP-based heuristic denoted as UB_2 . The following two sections of the table present the results obtained

after 10 runs of the VNS with the standard setting and with the usage of the restriction parameter λ in the local search. Here the strong restriction of the inverted sequences with $\lambda = 2$ has two important advantages. Because of the IFs a valid tour can consist of much more serviced edges as it was the case in the basic CARP. So the first advantage is the obvious reduction of the runtime per iteration. The other benefit lies in the stronger preservation of the subtours between the IFs. For both settings a maximal runtime of 10 minutes was preset. The last two columns show the best found solutions of VNS and the RPD between these values and those of Ghiani et al.

For the CARPIF instances a restricted version of the local search with $\lambda = 2$ clearly achieves better results. While the average improvement compared to the results of Ghiani et al. is 1.67% in the unrestricted case, it can be raised to 2.18% by using the optional sequence length parameter. Further the VNS found 23 new best solutions. For the remaining 11 instances the best known solutions were also achieved.

Table 5 reuses the structure of the previous tables. However, the fourth column provides a tour length restriction which is the essential constraint for the CLARPIF. Here the results obtained by 10 runs of VNS with standard setting are compared to the results of the upper bound UB_3 achieved with the TS reported by Ghiani et al. in [17]. The TS was performed on Pentium processor with 1 GHz. For the VNS which was executed on a Pentium processor with 3.6 GHz the maximal runtime was set to 10 minutes per instance.

Here the VNS outperforms the TS on all 28 instances with an average improvement of 13.80%. The average runtime for the VNS performed on an equivalent processor is 1999.92 seconds which is only 5.75% of the runtime for the TS used to solve the CLARPIF.

5 Conclusion

The proposed VNS outperforms all known heuristics on four benchmark sets. Two of them include the extension of IFs. Even for the well studied CARP instances of Belenguer et al. 4 new best solutions were found while improving all 28 solutions for the CLARPIF of Ghiani et al. with an average improvement of 13.80%. For all 120 instances the best known solution could be found and in 72 cases a new best solution was achieved.

It could be shown that the VNS is a simple and robust method. More precisely, the algorithm uses only one exchange operator, i.e., the CROSS-exchange, in the shaking phase and a simple inverting operator for the local search. Obviously there is a similarity to the 3-opt and the 2-opt operator respectively. Moreover the method gets by with only three parameters by default. A fourth parameter can be used to restrict the functionality of the local search. The robustness was demonstrated by solving different problem classes with the same algorithm and using a standard parameter setting for all problems. The high quality of the final solutions could be achieved independently of the quality of the initial solution.

Finally, even for the more complex instances originated by Eglese the VNS shows an excellent scaling according to the runtimes such that it is particularly suited for large real-world instances.

Acknowledgements

Michael Polacek, Karl F. Doerner and Richard F. Hartl gratefully acknowledge financial support of this research from the Oesterreichische Nationalbank (OeNB) under grant #11187, from the Fonds zur Förderung der wissenschaftlichen Forschung (FWF) under grant #L286-N04. Vittorio Maniezzo acknowledges Ser.In.Ar. fund "Optimization of technological networks". The authors thank Christoph Fichtenbauer for the implementation of the Ulusoy heuristic.

References

- [1] Amberg, A., Domschke, W., Voß, S. (2000): "Multiple Center Capacitated Arc Routing Problems: A tabu search algorithm using capacitated trees", *European Journal of Operational Research*, **124**, pp. 360–376.
- [2] Assad, A., Golden, B. (1995): "Arc routing methods and applications", M. Ball et al. (eds.), *Network Routing, Handbooks in Operations Research and Management Science*, **8**, pp. 375–483.
- [3] Baldacci, R. and Maniezzo, V. (2006): "Exact methods based on node routing formulations for arc routing problems", *Networks*, **47**, 52-60.
- [4] Belenguer, J. M., Benavent, E. (1998): "The capacitated arc routing problem: Valid inequalities and facets", *Computational Optimization & Applications*, **10**(2), pp. 165–187.
- [5] Belenguer, J. M., Benavent, E., Cognata, F. (1997): "A metaheuristic for the capacitated arc routing problem", Unpublished manuscript. University of Valencia, Spain 11, pp. 305-315.
- [6] Belenguer, J.M., Benavent, E., (2003): "A Cutting Plane Algorithm for the Capacitated Arc Routing Problem", *Computers & Operations Research*, **30**(5), pp. 705-728.
- [7] Belenguer, J. M., Benavent, E., Lacomme, P., Prins, C. (2006): "Lower and upper bounds for the mixed capacitated arc routing problem", to appear in *Computers & Operations Research*.
- [8] Beullens, P., Muyldermans, L., Cattrysse, D., Van Oudheusden, D. (2003): "A Guided Local Search Heuristic for the Capacitated Arc Routing Problem", *European Journal of Operational Research*, **147**, pp. 629–643.

- [9] O. Bräysy (2003): "A reactive variable neighborhood search for the vehicle-routing problem with time windows", *INFORMS Journal on Computing*, **15**(4). pp. 347–368.
- [10] Chu, F., Labadi, N., Prins, C. (2006): "A Scatter Search for the periodic capacitated arc routing problem", *European Journal of Operational Research*, **169**(2), pp. 586–605.
- [11] Dror, M. (2000): "*ARC ROUTING: Theory, Solutions and Applications*", Kluwer Academic Publishers.
- [12] Dueck, G. and Scheuer, T. (1990): "Threshold Accepting: A general purpose optimization algorithm appearing superior to Simulated Annealing", *Journal of Computational Physics* **90**, pp. 161–175.
- [13] Eglese, R.W. (1994): "Routing Winter Gritting Vehicles", *Discrete Applied Mathematics* **48**(3), pp. 231–244.
- [14] Fleury, G., Lacomme, P., Prins, C. (2004): "Evolutionary Algorithms for Stochastic Arc Routing Problems", *Proceedings of the EvoWorkshops 2004*, pp. 501–512.
- [15] Garey, M. R., Johnson, D. S. (1979): "Computers and Intractability: A Guide to the Theory of NP Completeness", W. H. Freeman & Co., New York.
- [16] Ghiani, G., Improta, G., Laporte, G. (2001): "The Capacitated Arc Routing Problem with Intermediate Facilities", *Networks*, **37**, pp. 134–143.
- [17] Ghiani, G., Guerriero, F., Laporte, G., Musmanno, R. (2004): "Tabu Search Heuristics for the Arc Routing Problem with Intermediate Facilities under Capacity and Length Restrictions", *Journal of Mathematical Modelling and Algorithms*, **3**, pp. 209–223.
- [18] Golden, B. L., Wong, R. T. (1981): "Capacitated arc routing problems", *Networks*, **11**, pp. 305–315.
- [19] Greistorfer, P. (2003): "A tabu scatter search metaheuristic for the arc routing problem", *Computers and Industrial Engineering*, **44**(2), pp. 249–266.
- [20] Hansen, P., Mladenović, N. (1999): "An Introduction to Variable Neighborhood Search", In: Voss, S. et al. (eds.): *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*. Kluwer Academic Publishers, Boston.
- [21] Hansen, P., Mladenović, N. (2001): "Variable Neighborhood Search: Principles and Applications", *European Journal of Operational Research* **130**, pp. 449–467.

- [22] Hertz, A., Laporte, G., Mittaz, M. (2000): "A Tabu search heuristic for the capacitated arc routing problem", *Operations Research*, **48**, pp. 129–135.
- [23] Hertz, A., Laporte, G., Mittaz, M. (2001): "A variable neighborhood descent algorithm for the undirected capacitated arc routing problem", *Transportation Science*, **35**, pp. 425–434.
- [24] Hirabayashi, R., Saruwatari, Y., Nishida, N. (1992): "Tour construction algorithm for the capacitated arc routing problems", *Asia Pacific Journal of Operations Research*, **9**(2), pp. 155–175.
- [25] Lacomme, P., Prins, C., Ramdane-Cherif, W. (2004): "Competitive Memetic Algorithms for Arc Routing Problems", *Annals of Operations Research*, **131**, pp. 159–185.
- [26] Lacomme, P., Prins, C., Ramdane-Cherif, W. (2005): "Evolutionary algorithms for periodic arc routing problems", *European Journal of Operational Research*, **165**(2), pp. 535–553.
- [27] Lacomme, P., Prins, C., Ramdane-Cherif, W. (2001): "A Genetic Algorithm for the Capacitated Routing Problem and its Extensions", Proceedings of the EvoWorkshops 2001, E.J.W. Boers et al. (eds.), pp. 473–483.
- [28] Maniezzo, V. (2004): "Metaheuristics for large directed CARP instances: urban solid waste collection operational support", Technical Report UBLCS-2004-16, Dept. Computer Science, University of Bologna.
- [29] Maniezzo, V. and Baldacci, R. (2006): "Exact solution of mixed CARP instances", *Proceedings of the Odysseus 2006*, Altea, Spain.
- [30] Mladenović, N., Hansen, P. (1997): "Variable Neighborhood Search", *Computers and Operations Research*, **24**, pp. 1097–1100.
- [31] Polacek, M., Hartl, R. F., Doerner, K. F., Reimann, M. (2004): "A variable neighborhood search for the multi depot vehicle routing problem with time windows", *Journal of Heuristics*, **10**(6), pp. 613–627.
- [32] Polacek, M., Doerner, K. F., Hartl, R. F., Kiechle, G., Reimann, M. (2006): "Scheduling periodic customer visits for a traveling salesperson", to appear in *European Journal of Operational Research*.
- [33] Snizek, J., Bodin, L., Levy, L., Ball, M. (2002): "Capacitated Arc Routing Problem with Vehicle-Site Dependencies: the Philadelphia Experience", Toth P. and Vigo, D. (eds.), *The Vehicle Routing Problem*, SIAM, pp. 287–308.
- [34] Taillard, E. D., Badeau, P., Gendreau, M., Guertin, F., Potvin, J. Y. (1997): "A tabu search heuristic for the vehicle routing problem with soft time windows", *Transportation Science*, **31** (1997) 170–186.

- [35] Ulusoy, G. (1985): "The Fleet Size and Mix Problem for Capacitated Arc Routing", *European Journal of Operational Research* **22**, pp. 329–337.
- [36] Wirasinghe, C., Waters, N.M. (1983): "An approximate procedure for determining the number, capacities and locations of solid waste transfer-stations in an urban region", *European Journal of Operational Research* **12**, pp. 105–111.
- [37] Wøhlk, S. (2006): "New lower bound for the Capacitated Arc Routing Problem", to appear in *Computers & Operations Research*.
- [38] Xin, X. (2000): "Hanblen SWRoute: A GIS Based Spatial Decision Support System for Designing Solid Waste Collection Routes in Rural Countries", The University of Tennessee, Knoxville, U.S.A.