An ILP Improvement Procedure for the Open Vehicle Routing Problem*

Majid Salari, Paolo Toth, and Andrea Tramontani

DEIS, University of Bologna,
Viale Risorgimento 2, 40136 Bologna, Italy
{majid.salari2,paolo.toth,andrea.tramontani}@unibo.it

Abstract. We address the Open Vehicle Routing Problem (OVRP), a variant of the "classical" Vehicle Routing Problem (VRP) in which the vehicles are not required to return to the depot after completing their service. We present a heuristic improvement procedure for OVRP based on Integer Linear Programming (ILP) techniques. Given an initial solution to be possibly improved, the method follows a destruct-and-repair paradigm, where the given solution is randomly destroyed (i.e., customers are removed in a random way) and repaired by solving an ILP model, in the attempt of finding a new improved solution. The overall procedure can be considered as a general framework which could be extended to cover other variants of Vehicle Routing Problems. We report computational results on benchmark instances from the literature. In several cases, the proposed algorithm is able to find the new best-known solution for the considered problem.

Key words: integer linear programming, local search, heuristics, open vehicle routing problem.

1 Introduction

We address the Open Vehicle Routing Problem (OVRP), a variant of the "classical" Vehicle Routing Problem (VRP) in which the vehicles are not required to return to the depot after completing their service. OVRP can be formally stated as follows. We are given a central *depot* and a set of n *customers*, which are associated with the nodes of a complete undirected graph $G = (V, E)$ (where $V = \{0, 1, \ldots, n\}$, node 0 represents the depot and $V \setminus \{0\}$ is the set of customers). Each edge $e \in E$ has an associated finite *cost* $c_e \geq 0$ and each customer $v \in V \setminus \{0\}$ has a *demand* $q_v > 0$ (with $q_0 = 0$). A fleet of m identical *vehicles* is located at the depot, each one with a *fixed cost* F, a *capacity* Q and a *total distance-travelled (duration)* limit D. The customers must be served by at most m Hamiltonian paths (*open routes*), each path associated with one vehicle, starting at the depot and ending at one of the customers. Each route must have

* This work has been partially supported by MUR (Ministero Università e Ricerca), Italy.

a total duration (computed as the sum of the edge costs in the route) not exceeding the given limit D of the vehicles, and can visit a subset S of customers whose total demand $\sum_{v \in S} q_v$ does not exceed the given capacity Q. The problem consists of finding a feasible solution covering (i.e., visiting) exactly once all the customers and having a minimum overall cost, computed as the sum of the traveling cost plus the fixed cost associated with the vehicles used to serve the customers.

In this paper we present a heuristic improvement procedure for OVRP based on Integer Linear Programming (ILP) techniques. Given an initial solution to be possibly improved, the procedure iteratively performs the following steps: (a) select several customers from the current solution, and build the restricted solution obtained from the current one by extracting (i.e., short-cutting) the selected customers; (b) reallocate the extracted customers to the restricted solution by solving an ILP problem, in the attempt of finding a new improved solution. This method has been proposed by De Franceschi et al. [4] and deeply investigated by Toth and Tramontani [22] in the context of the classical VRP. Here, we consider a simpler version of this procedure, which does not exploit any particular feature of the addressed problem. The method follows a destruct-and-repair paradigm, where the current solution is randomly destroyed (i.e., customers are removed in a random way) and repaired by following ILP techniques. Hence, the overall procedure can be considered as a general framework which could be extended to cover other variants of Vehicle Routing Problems.

The paper is organized as follows. Section 2 recalls the main works proposed in the literature for OVRP. In Section 3 we describe a neighborhood for OVRP and the ILP model which allows to implicitly define and explore the presented neighborhood. The implementation of the heuristic improvement procedure is given in Section 4, while Section 5 reports the computational experiments on benchmark capacitated OVRP instances from the literature (with/without distance constraints), comparing the presented method with the most effective metaheuristic techniques proposed for OVRP. Some conclusions are finally drawn in Section 6.

2 Literature review

The classical VRP is a fundamental combinatorial optimization problem which has been widely studied in the literature (see, e.g., Toth and Vigo [23] and Cordeau et al. [3]). At first glance, having open routes instead of closed ones looks like a minor change, and in fact OVRP can be also formulated as a VRP on a directed graph, by fixing to 0 the costs of the arcs entering the depot. However, if the undirected case is considered, the open version turns out to be more general than the closed one. Indeed, as shown by Letchford et al. [12], any closed VRP on n customers can be transformed into an OVRP on n customers, but there is no transformation in the reverse direction. Further, there are many practical applications in which OVRP naturally arises. This happens, of course, when a company does not own a vehicle fleet, and hence customers are served

by hired vehicles which are not required to come back to the depot (see, e.g., Tarantilis et al. [21]). But the *open model* also arises in pick-up and delivery applications, where each vehicle starts at the depot, delivers to a set of customers and then it is required to visit the same customers in reverse order, picking up items that have to be backhauled to the depot. An application of this type is described in Schrage [18]. Further areas of application, involving the planning of train services and the planning of a set of school bus routes, are reported by Fu et al. [9].

OVRP is $\mathcal{NP}$-hard in the strong sense, and has recently received an increasing attention in the literature. Exact branch-and-cut and branch-cut-and-price approaches have been proposed, respectively, by Letchford et al. [12] and Pessoa et al. [14], addressing the capacitated problem with no distance constraints and no empty routes allowed (i.e., $D = \infty$ and customers must be served by exactly m routes). Heuristic and metaheuristic algorithms usually take into account both capacity and distance constraints, and consider the number of routes as a decision variable. In particular, an unlimited number of vehicles is supposed to be available (i.e., $m = \infty$) and the objective function is generally to minimize the number of used vehicles first and the traveling cost second, assuming that the fixed cost of an additional vehicle always exceeds any traveling cost that could be saved by its use (i.e., considering $F = \infty$). However, several authors address also the variant in which there are no fixed costs associated with the vehicles (i.e., $F = 0$) and hence the objective function is to minimize the total traveling cost with no attention on the number of used vehicles (see, e.g., Tarantilis et al. [21]). Considering capacity constraints only (i.e., taking $D = \infty$), Sariklis and Powell [17] propose a two-phase heuristic which first assigns customers to clusters and then builds a Hamiltonian path for each cluster, Tarantilis et al. [19] describe a population-based heuristic, while Tarantilis et al. [20, 21] present threshold accepting metaheuristics. Taking into account both capacity and distance constraints, Brandão [1], Fu et al. [9, 10] and Derigs and Reuter [5] propose tabu search heuristics, Li et al. [13] describe a record-to-record travel heuristic, Pisinger and Ropke [15] present an adaptive large neighborhood search heuristic which follows a destruct-and-repair paradigm, while Fleszar et al. [8] propose a variable neighborhood search heuristic.

3 Reallocation Model

Let z be a feasible solution of the OVRP defined on G. For any given node subset $\mathcal{F} \subset V \setminus \{0\}$, we define $z(\mathcal{F})$ as the *restricted solution* obtained from z by *extracting* (i.e., by short-cutting) all the nodes $v \in \mathcal{F}$. Let $\mathcal{R}$ be the set of routes in the restricted solution, $\mathcal{I} = \mathcal{I}(z, \mathcal{F})$ the set of all the edges in $z(\mathcal{F})$, and $\mathcal{S} = \mathcal{S}(\mathcal{F})$ the set of all the *sequences* which can be obtained through the recombination of nodes in $\mathcal{F}$ (i.e., the set of all the elementary paths in $\mathcal{F}$). Each edge $i \in \mathcal{I}$ is viewed as a potential *insertion point* which can allocate one or more nodes in $\mathcal{F}$ through at most one sequence $s \in \mathcal{S}$. We say that the insertion point $i = (a, b) \in \mathcal{I}$ allocates the nodes $\{v_j \in \mathcal{F} : j = 1, \ldots, h\}$ through the sequence

$s = (v_1, v_2, \ldots, v_h) \in \mathcal{S}$, if the edge (a, b) in the restricted solution is replaced by the edges $(a, v_1), (v_1, v_2), \ldots, (v_h, b)$ in the new feasible solution. Since the restricted routes, as well as the final ones, are open paths starting at the depot, in addition to the edges of the restricted solution we also consider the insertion points $i = (p_r, 0)$, where p_r denotes the last customer visited by the route $r \in \mathcal{R}$, which allow to append any sequence to the last customer of any restricted route. Further, empty routes in the restricted solution are associated with insertion points $(0, 0)$.

For each sequence $s \in \mathcal{S}$, let $c(s)$ and $q(s)$ denote, respectively, the sum of the costs of the edges in s and the sum of the demands of the nodes in s. For each insertion point $i = (a, b) \in \mathcal{I}$ and for each sequence $s = (v_1, v_2, \ldots, v_h) \in \mathcal{S}$, we define γ_{si} as the extra-cost (i.e., the extra-distance) for assigning sequence s to insertion point i in its best possible orientation (i.e., $\gamma_{si} := c(s) - c_{ab} + \min\{c_{av_1} + c_{v_h b}, c_{av_h} + c_{v_1 b}\}$). Note that the insertion points involving the depot must be treated as directed arcs (the orientation being given by the corresponding restricted routes); i.e., if $i = (p_r, 0)$, then γ_{si} is computed as $c(s) + \min\{c_{p_r v_1}, c_{p_r v_h}\}$. The extra-cost for assigning the sequence s to the insertion point $i = (0, 0)$ associated with an empty route is simply $c(s) + \min\{c_{0v_1}, c_{0v_h}\}$. For each route $r \in \mathcal{R}$, let $\mathcal{I}(r)$ denote the set of insertion points associated with r, while letting $\tilde{q}(r)$ and $\tilde{c}(r)$ denote, respectively, the total demand and distance computed for route r, still in the restricted solution.

For each $i \in \mathcal{I}$, suppose $\mathcal{S}_i \subseteq \mathcal{S}$ is a sequence subset, containing all the sequences which can be allocated to the specific insertion point i. The definition of $\mathcal{S}_i$ will be discussed later in this section. Then, a neighborhood of the given solution z can be formulated (and explored) by solving an ILP problem (denoted as the *Reallocation Model*) based on the decision variables

$$x_{si} = \begin{cases} 1 & \text{if sequence } s \in \mathcal{S}_i \text{ is allocated to insertion point } i \in \mathcal{I} \\ 0 & \text{otherwise} \end{cases}, \tag{1}$$

which reads as follows:

$$\sum_{r \in \mathcal{R}} \tilde{c}(r) + \min \sum_{i \in \mathcal{I}} \sum_{s \in \mathcal{S}_i} \gamma_{si} x_{si} \tag{2}$$

subject to:

$$\sum_{i \in \mathcal{I}} \sum_{s \in \mathcal{S}_i(v)} x_{si} = 1 \quad v \in \mathcal{F} \ , \tag{3}$$

$$\sum_{s \in \mathcal{S}_i} x_{si} \leq 1 \quad i \in \mathcal{I} \ , \tag{4}$$

$$\sum_{i \in \mathcal{I}(r)} \sum_{s \in \mathcal{S}_i} q(s) x_{si} \leq Q - \tilde{q}(r) \quad r \in \mathcal{R} \ , \tag{5}$$

$$\sum_{i \in \mathcal{I}(r)} \sum_{s \in \mathcal{S}_i} \gamma_{si} x_{si} \leq D - \tilde{c}(r) \quad r \in \mathcal{R} \ , \tag{6}$$

$$x_{si} \in \{0, 1\} \quad i \in \mathcal{I}, \ \ s \in \mathcal{S}_i \ , \tag{7}$$

where, for any $i \in \mathcal{I}$ and $v \in \mathcal{F}$, $\mathcal{S}_i(v)$ denotes the set of sequences covering the customer v which can be allocated to the insertion point i. The objective function (2), to be minimized, gives the traveling cost of the final OVRP solution. Constraints (3) impose that each extracted node belongs to exactly one of the selected sequences, i.e., that it is covered exactly once in the final solution. Constraints (4) avoid to allocate two or more sequences to the same insertion point. Finally, constraints (5) and (6) impose that each route in the final solution fulfills the capacity and distance restrictions, respectively. Note that, if there is a non-null fixed cost F associated with the vehicles, it can be taken into account by simply adding F to the cost of the edges incident at the depot node.

The Reallocation Model (2)–(7) defines a neighborhood of a given solution which depends on the extracted nodes $\mathcal{F}$ and on the subsets $\mathcal{S}_i$. In particular, for any given $\mathcal{F}$, the choice of $\mathcal{S}_i$ is a key factor in order to allow an effective exploration of the solution space in the neighborhood of the given solution. The subsets $\mathcal{S}_i$ are built by following a column generation approach: we initialize the Linear Programming relaxation of the Reallocation Model (LRM) with a subsets of variables with small insertion cost and afterwards we iteratively solve the column generation problem associated with LRM, adding other variables with *small* reduced cost. The overall procedure for building the subsets $\mathcal{S}_i$ can be described as follows.

1. (Initialization) For each insertion point $i \in \mathcal{I}$, initialize each subset $\mathcal{S}_i$ with the *basic* sequence extracted from i plus the feasible singleton sequence with the minimum insertion cost (i.e., the sequence (v), with $v \in \mathcal{F}$, with the minimum extra-cost among all the singleton sequences which can be allocated to i without violating the capacity and distance restrictions for the restricted route containing i). Initialize LRM with the initial set of variables corresponding to the current subsets $\mathcal{S}_i$, and solve LRM.
2. (Column generation) For each insertion point $i \in \mathcal{I}$, solve the column generation problem associated with i, adding to $\mathcal{S}_i$ all the sequences s corresponding to elementary paths in $\mathcal{F}$, whose associated variables x_{si} have a reduced cost rc_{si} under a given threshold RC_{max} (i.e., variables x_{si} such that $rc_{si} \leq RC_{max}$). If at least one sequence/variable has been added, solve the new LRM and repeat from step 2. Otherwise terminate.

For any fixed insertion point $i \in \mathcal{I}$, the column generation problem associated with i in LRM is in practice a Resource Constrained Elementary Shortest Path Problem (RCESPP), which usually arises in the Set Partitioning formulation of the classical VRP (see, e.g., Feillet et al. [6] and Righini and Salani [16]). Here, for each insertion point $i \in \mathcal{I}$, we solve the corresponding RCESPP through a simple greedy heuristic, with the aim of finding as many variables with small reduced cost as possible. Hashing techniques are used to avoid the generation of duplicated variables.

Note that each subset $\mathcal{S}_i$ contains the *basic* sequence extracted from the insertion point i, and hence the current solution can always be obtained as a new feasible solution of the Reallocation Model.

4 Heuristic Improvement Procedure

The Reallocation Model described in the previous section allows for exploring a neighborhood of a given feasible solution, depending on the choice of the extracted customers in $\mathcal{F}$. We propose a heuristic improvement procedure for OVRP, based on the model (2)–(7), which iteratively explores different neighborhoods of the current solution. Given an initial feasible solution z_0 for OVRP (taken from the literature or found by any heuristic method), the procedure works as follows.

1. (Initialization) Set $kt := 0$ and $kp := 0$. Take z_0 as the incumbent solution and initialize the current solution z_c as $z_c := z_0$.
2. (Node selection) Build the set $\mathcal{F}$ by selecting each customer with a probability p.
3. (Node extraction) Extract the nodes selected in the previous step from the current solution z_c and construct the corresponding restricted OVRP solution $z_c(\mathcal{F})$, obtained by short-cutting the extracted nodes.
4. (Reallocation) Define the subsets $\mathcal{S}_i$ ($i \in \mathcal{I}(z_c, \mathcal{F})$) as described in Section 3. Build the corresponding Reallocation Model (2)–(7) and solve the model by using a general-purpose ILP solver. Once an optimal ILP solution has been found, construct the corresponding new OVRP solution and possibly update z_c and z_0.
5. (Termination) Set $kt := kt + 1$. If $kt = KT_{max}$, terminate.
6. (Perturbation) If z_c has been improved in the last iteration, set $kp := 0$; otherwise set $kp := kp + 1$. If $kp = KP_{max}$, "perturb" the current solution z_c and set $kp := 0$. In any case, repeat from step 2.

The procedure performs KT_{max} iterations and at each iteration explores a randomly generated neighborhood of the current solution z_c. However, if z_c is not improved for KP_{max} consecutive iterations, we introduce a random perturbation in order to move to a different area of the solution space, so as to enforce the diversification of the search. In particular, when creating a perturbation, we randomly extract np customers from z_c (with np randomly chosen in $[np_{min}, np_{max}]$), and we reinsert each extracted customer, in turn, in its best feasible position. If a customer cannot be inserted in any currently non-empty route (due to the capacity and/or distance restrictions), a new route is created to allocate the customer. In general, when creating the perturbation, several customers cannot be inserted in the non-empty routes of the current solution, and hence the new perturbed solution uses more vehicles than the minimum required.

5 Computational Results

The performance of the Heuristic Improvement Procedure (HIP) described in the previous sections was evaluated on the 16 benchmark instances usually addressed in the literature, taken from Christofides et al. [2] (instances `C1`–`C14`) and from

Fisher [7] (instances F11–F12). The number of customers ranges from 50 to 199. C1–C5, C11–C12 and F11–F12 have only capacity constraints, while C6–C10 and C13–C14 are the same instances as C1–C5 and C11–C12, respectively, but with both capacity and distance constraints. As usual, for the problems with distance constraints, D is taken as the original value for the classical VRP multiplied by 0.9.

HIP needs an initial solution to be given, which in principle could be computed through any available constructive heuristic algorithm. We decided to run HIP starting from an extremely-good feasible solution (in several cases, the best-known solution reported in the literature), with the aim of attempting to further improve it (this is of course possible only if the initial solution is not optimal, as it is the case for some of them). In particular, we considered as initial solutions the ones obtained by Fu et al. [9, 10], by Fleszar et al. [8] and by Pisinger and Ropke [15]. Since in several cases the same authors report different solutions for the same instance, obtained by using slightly different versions of their algorithm, among the solutions provided for the same instance, we considered as initial solutions for HIP the best ones provided by [10], [8] and [15], respectively.

HIP has been tested on a Pentium IV 3.4 GHz with 1 GByte RAM, running under Microsoft Windows XP Operative System, and has been coded in C++ with Microsoft Visual C++ 6.0 compiler. The ILP solver used in the experiments is ILOG Cplex 10.0 [11]. HIP setting depends on the parameters RC_{max}, p, np_{min}, np_{max}, and on the number of iterations KP_{max} and KT_{max}. Although these parameters could be tuned considering the edge costs and the particular characteristics of the tested instances, we preferred to run all the experiments with a fixed tuning: $RC_{max} = 1$, $p = 0.5$ (i.e., the 50% of the customers are selected on average), $np_{min} = 15$, $np_{max} = 25$, $KP_{max} = 50$ and $KT_{max} = 5,000$ (i.e., we perform globally 5,000 iterations, and the current solution is perturbed if it cannot be improved for 50 consecutive iterations). Finally, since most of the authors address the problem considering as objective function to minimize the number of vehicles first and the traveling cost second, we decided to run HIP without allowing to increase the number of vehicles used by the initial solution (i.e., we keep as maximum number of available vehicles the one used by the initial solution). However, as stated in Section 4, the Perturbation Step often requires an additional route to be created (to preserve the feasibility of the solution). In such cases, we add a small penalty θ to the cost of the edges incident at the depot, in order to force HIP to "recover" the solution in the following iterations. After some preliminary tests, we decided to fix $\theta = 12$ for the considered instances.

Computational results are reported in Tables 1–3. All the tables have the same structure, and the computing times are given in seconds. The first column gives the instance name, while columns 2–4 report the number of used vehicles and the traveling cost of the best known solution, considering both the OVRP variants in which $F = \infty$ (i.e., the objective is to minimize the number of used vehicles first and the traveling cost second) and $F = 0$ (i.e., the objective is to minimize the traveling cost). The best known solution cost for the case $F = 0$ is reported only if smaller than the corresponding one for the case $F = \infty$. Note

that, for all the solutions but those corresponding to $F = \infty$, we report the number of used vehicles and the cost as $(m)/cost$, where m is given only if greater than the one obtained in the best known solution for the case $F = \infty$. Columns 5–6 report the cost of the initial solution given to HIP and the CPU time of the corresponding algorithm from the literature (the solution cost has been recomputed, while the CPU time is taken from [8]). The last three columns report the computational results provided by HIP. For each instance, we report the final solution cost, the CPU time required to reach the final solution (*b.time*) and the overall computing time required to perform all the 5,000 iterations (*t.time*). When HIP was not able to improve on the initial solution, we mark with a "—" the final solution cost and the corresponding *b.time*. Final solution costs equal to the previously best known ones are underlined, new best solutions are in bold face, while provably optimal solutions, taken from Letchford et al. [12], are marked with an * (HIP was not run when the initial solution is provably optimal). As also reported in the last row of the tables, the CPU times related to the algorithms by Fu et al. [9, 10] and by Pisinger and Ropke [15] were obtained on a Pentium IV 3 GHz, while the CPU times related to the algorithm by Fleszar et al. [8] were obtained on a Pentium M 2 GHz. We have to note that the cost of the initial solution reported in Table 1 for instance `C8` is different from all the ones reported in [10], and hence for this initial solution we did not report the corresponding computing time.

Table 1. Computational results on benchmark instances starting from the solutions by Fu et al. [10].

Instance	Prev. best sol. $F = \infty$		Prev. best sol. $F = 0$	Initial solution		HIP		
	m	cost	(m)/cost	(m)/cost	time	(m)/cost	b.time	t.time
C1	5	*416.06	(6)/412.96	*416.06	0.8			
C2	10	567.14	(11)/564.06	567.14	7.8	—	—	84.2
C3	8	*639.74	(9)/639.57	641.88	23.2	*<u>639.74</u>	106.0	119.9
C4	12	733.13		738.94	6.8	<u>733.13</u>	21.2	156.6
C5	16	879.37	(17)/869.25	(17)/878.95	61.9	(17)/**868.81**	10.3	220.3
C6	6	412.96		412.96	0.6	—	—	45.1
C7	10	583.19	(11)/568.49	(11)/568.49	6.0	—	—	83.1
C8	9	644.63		646.31		<u>644.63</u>	0.1	136.2
C9	13	757.84	(14)/756.14	(14)/761.28	46.6	(14)/<u>756.14</u>	102.7	255.5
C10	17	875.07		903.10	51.9	878.54	323.9	460.2
C11	7	682.12	(10)/678.54	717.15	23.1	683.64	165.8	198.8
C12	10	*534.24		534.71	4.2	*<u>534.24</u>	1.6	94.0
C13	11	904.04	(12)/896.50	(12)/917.90	82.1	(12)/**894.19**	475.0	1165.3
C14	11	591.87	(12)/581.81	600.66	2.5	<u>591.87</u>	293.8	354.7
F11	4	*177.00		*177.00	0.4			
F12	7	769.66		777.07	28.4	**769.55**	77.8	148.2
				Pentium IV 3 GHz		Pentium IV 3.4 GHz		

The tables show that HIP is able to improve even extremely-good quality solutions, obtained by some of the most effective metaheuristic techniques proposed for OVRP. Taking into account the different performance of the processors

Table 2. Computational results on benchmark instances starting from the solutions by Fleszar et al. [8].

Instance	Prev. best sol. $F=\infty$		Prev. best sol. $F=0$	Initial solution		HIP		
	m	cost	(m)/cost	(m)/cost	time	(m)/cost	b.time	t.time
C1	5	*416.06	(6)/412.96	*416.06	0.5			
C2	10	567.14	(11)/564.06	567.14	2.3			
C3	8	*639.74	(9)/639.57	*639.74	239.6			
C4	12	733.13		733.13	585.0	—	—	151.2
C5	16	879.37	(17)/869.25	905.96	13.2	900.01	349.3	436.6
C6	6	412.96		412.96	1.2			
C7	10	583.19	(11)/568.49	596.47	1.0	584.32	70.8	96.8
C8	9	644.63		644.63	587.6	—	—	136.8
C9	13	757.84	(14)/756.14	760.06	1094.1	758.24	41.45	121.2
C10	17	875.07		875.67	1252.4	**874.71**	30.6	396.2
C11	7	682.12	(10)/678.54	682.12	5.7	—	—	178.3
C12	10	*534.24		*534.24	163.7			
C13	11	904.04	(12)/896.50	904.04	1820.1	**899.16**	6.3	959.3
C14	11	591.87	(12)/581.81	591.87	389.0	—	—	276.3
F11	4	*177.00		178.09	140.2	*177.00	14.2	98.5
F12	7	769.66		769.66	75.4	**769.55**	56.6	142.2
				Pentium M 2 GHz		Pentium IV 3.4 GHz		

Table 3. Computational results on benchmark instances starting from the solutions by Pisinger and Ropke [15].

Instance	Prev. best sol. $F=\infty$		Prev. best sol. $F=0$	Initial solution		HIP		
	m	cost	(m)/cost	(m)/cost	time	(m)/cost	b.time	t.time
C1	5	*416.06	(6)/412.96	*416.06	120.0			
C2	10	567.14	(11)/564.06	567.14	360.0			
C3	8	*639.74	(9)/639.57	641.76	850.0	—	—	90.2
C4	12	733.13		733.13	1790.0	—	—	161.2
C5	16	879.37	(17)/869.25	896.08	2370.0	892.37	276.5	450.2
C6	6	412.96		412.96	200.0			
C7	10	583.19	(11)/568.49	583.19	330.0	—	—	80.6
C8	9	644.63		645.16	1140.0	—	—	130.9
C9	13	757.84	(14)/756.14	757.84	1850.0	**757.73**	33.9	412.5
C10	17	875.07		875.67	1200.0	**874.71**	201.3	459.8
C11	7	682.12	(10)/678.54	682.12	730.0			
C12	10	*534.24		*534.24	800.0			
C13	11	904.04	(12)/896.50	909.80	610.0	905.87	360.2	874.5
C14	11	591.87	(12)/581.81	591.87	400.0			
F11	4	*177.00		*177.00	690.0			
F12	7	769.66		770.17	2370.0	—	—	149.3
				Pentium IV 3 GHz		Pentium IV 3.4 GHz		

used for testing the different algorithms, the overall computing time required by HIP is comparable with the other ones reported in the tables and in several cases the final improved solution is found very quickly. For instance `C1`, with $F = \infty$, [10], [8] and [15] all yield a provably optimal solution, and in several instances they provide the same initial solution. Hence, our test-bed concerns in practice 15 instances (instead of 16) and 34 different non provably optimal

initial solutions which could be improved. (Note that, for instances C4 and C10, the solution provided by [8] and [15] are different but with the same cost.) Out of these 34 solutions, HIP improves on the initial solution in 22 cases, in 7 cases reaches the previously best known solution (provably optimal in 3 cases), while in 8 cases finds a new best solution. Concerning the 12 initial solutions which HIP does not improve, it is worth noting that 9 of them are best known solutions (for the case $F = \infty$ or $F = 0$).

In addition, we considered also the (previously) best known solution provided by Derigs and Reuter [5] for instances C5, C9, C10 and C14. HIP was not able to improve on C9 and C14, while for instance C5 found a solution of cost (17)/868.93 after 2960 iterations and 130.2 seconds, starting from (17)/869.24, and for instance C10 found a solution of cost 874.71 after 56 iterations and 2.6 seconds, starting from 875.07.

In order to look for possible better solutions, we performed some additional experiments. In particular, after the first 5,000 iterations, we ran HIP for 2,000 more iterations with a slightly different parameter setting. Starting from the solutions provided by Fu et al. [10], for instance C5 HIP found a solution of cost (17)/868.44 after 5220 iterations and 237.4 seconds, for instance C10 a solution of cost 878.52 after 5151 iterations and 483.1 seconds, and for instance C11 a solution of cost 683.15 after 6371 iterations and 380.1 seconds. (Note that the solution obtained for instance C5 corresponds to a further improvement on the previous best-known solution.)

Finally, still starting from the solutions by Fu et al. [10], we ran HIP with a different tuning of the parameter p, to understand how the neighborhood size affects the overall performance of the method, both in terms of quality of the solutions found and of CPU time. Let $z_{avg}(\bar{p})$ be the average final solution value obtained on the 14 instances C2–C14 and F12 with $p = \bar{p}$, and let $ttime_{avg}(\bar{p})$ be the corresponding average CPU time in seconds. With $p = 0.3$, $p = 0.5$ and $p = 0.7$ we obtained the following results: $z_{avg}(0.3) = 684.55$ and $ttime_{avg}(0.3) = 71.9$, $z_{avg}(0.5) = 681.65$ and $ttime_{avg}(0.5) = 251.6$, $z_{avg}(0.7) = 683.32$ and $ttime_{avg}(0.7) = 460.0$. As expected, the average CPU time consistently increases with the number of extracted customers, while the best solution values are obtained with the default setting of p (i.e., $p = 0.5$), thus indicating that extracting too many customers leads in general to worse solutions (i.e., $z_{avg}(0.7) > z_{avg}(0.5)$). This is not completely surprising, and it is essentially due to the column generation heuristic, which falls in troubles in finding good variables for the Reallocation Model when the current solution has been completely "destroyed" by the removal of too many customers.

The current best known solution costs are given in summary in Table 5, where we also report the number of customers n and the total distance-travelled limit D of the tested problems. For each solution which was not improved by HIP we report the algorithms providing the corresponding best known cost. Previously best known solution costs reached also by HIP are underlined, while new best solution costs found by HIP are in bold face. For the capacitated instances, in

the case $F = \infty$, we also report the best known lower bound LB taken from [12] and [14].

Table 4. Current best known solutions for the tested OVRP benchmark instances.

Inst.	n	D	Best known sol. $F = \infty$				$F = 0$	
			m	LB	cost	best heuristics	(m)/cost	best heursitiscs
C1	50		5	416.1	*416.06	[1], [5], [8], [9, 10], [13], [15]	(6)/412.96	[19], [20], [21]
C2	75		10	559.62	567.14	[5], [8], [9, 10], [13], [15]	(11)/564.06	[19], [20], [21]
C3	100		8	639.7	*639.74	[5], [8], [13]	(9)/639.57	[21]
C4	150		12	730.2	733.13	[5], [8], [13], [15]		
C5	199		16	848.5	879.37	[20]	(17)/**868.44**	
C6	50	180	6		412.96	[1], [5], [8], [9, 10], [13], [15]		
C7	75	144	10		583.19	[15]	(11)/568.49	[5], [9, 10], [13]
C8	100	207	9		644.63	[1], [5], [8], [13]		
C9	150	180	13		**757.73**		(14)/756.14	[5]
C10	199	180	17		**874.71**			
C11	120		7	657.1	682.12	[5], [8], [15]	(10)/678.54	[21]
C12	100		10	534.2	*534.24	[5], [8], [13], [15], [19], [20], [21]		
C13	120	648	11		**899.16**		(12)/**894.19**	
C14	100	936	11		591.87	[5], [8], [13], [15]	(12)/581.81	[5]
F11	71		4	177.0	*177.00	[5], [9, 10], [13], [15]		
F12	134		7	762.9	**769.55**			

6 Conclusions and Future Directions

We addressed the Open Vehicle Routing Problem (OVRP) and presented a heuristic improvement procedure for OVRP based on Integer Linear Programming (ILP) techniques. Given an initial solution to be possibly improved, the method follows a destruct-and-repair paradigm, where the given solution is randomly destroyed (i.e., customers are removed in a random way) and repaired by solving an ILP model, in the attempt of finding a new improved solution.

Computational results on benchmark instances from the literature showed that the proposed method can be used as a profitable tool for improving existing OVRP solutions, and that even extremely-good quality solutions found by the most effective metaheuristic techniques proposed for OVRP can be further improved. Out of the 21 best known solutions which are not provably optimal, in 6 cases the proposed method was able to improve the best-known solution reported in the literature.

Future directions of work could involve more sophisticated criteria for removing customers from the current solution, as well as more sophisticated algorithms for solving the column generation problem related to the ILP model. Further, it could be interesting to consider the larger benchmark instances proposed by Li et al. [13] and addressed also by Derigs and Reuter [5], and to test the method with random initial solutions, in order to understand if "near-optimal" solutions can be reached even when starting from "poor-quality" initial solutions. On the other side, the overall procedure can be considered as a general framework and it could be extended to cover other variants of Vehicle Routing Problems, as, for example, Vehicle Routing Problems with heterogenous vehicles and multi-depot Vehicle Routing Problems.

Acknowledgments. We wish to thank Zhuo Fu, Richard Eglese, Leon Li, Krzysztof Fleszar, Ibrahim H. Osman, Khalil S. Hindi, David Pisinger, Stefan Ropke, Ulrich Derigs and Katharina Reuter who provided the initial solutions used in the computational experiments.

References

1. Brandão, J.: A tabu search algorithm for the open vehicle routing problem. European Journal of Operational Research 157, 552-564 (2004)
2. Christofides, N., Mingozzi, A., Toth, P.: The vehicle routing problem. In: Christofides, N., Mingozzi, A., Toth, P., Sandi, C. (eds.), Combinatorial Optimization, pp. 313–338. Wiley, Chichester (1979)
3. Cordeau, J.-F., Laporte, G., Savelsbergh, M.W.P., Vigo, D.: Vehicle routing. In: Barnhart, C., Laporte, G. (eds.), Transportation. Handbooks in Operations Research and Management Science, vol. 14, pp. 367–428. Elsevier, Amsterdam (2007)
4. De Franceschi, R., Fischetti, M., Toth, P.: A new ILP-based refinement heuristic for vehicle routing problems. Mathematical Programming 105, 471–499 (2006)
5. Derigs, U., Reuter, K.: A simple and efficient tabu search heuristic for solving the open vehicle routing problem. Technical report, WINFORS–University of Cologne, Germany (2008)
6. 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 44, 216–229 (2004)
7. Fisher, M.: Optimal solutions of vehicle routing problems using minimum k-trees. Operations Research 42, 626–642 (1994)
8. Fleszar, K., Osman, I.H., Hindi, K.S.: A variable neighbourhood search for the open vehicle routing problem. European Journal of Operational Research (2008), doi:10.1016/j.ejor.2007.06.064
9. Fu, Z., Eglese, R., Li, L.Y.O.: A new tabu search heuristic for the open vehicle routing problem. Journal of the Operational Research Society 56, 267-274 (2005)
10. Fu, Z., Eglese, R., Li, L.Y.O.: Corrigendum: A new tabu search heuristic for the open vehicle routing problem. Journal of the Operational Research Society 57, 1018 (2006)
11. ILOG Cplex 10.0: User's Manual and Reference Manual, ILOG, S.A., `http://www.ilog.com`
12. Letchford, A.N., Lysgaard, J., Eglese, R.W.: A branch-and-cut algorithm for the capacitated open vehicle routing problem. Journal of the Operational Research Society 58, 1642–1651 (2007)
13. Li, F., Golden, B., Wasil, E.: The open vehicle routing problem: Algorithms, large-scale test problems, and computational results. Computers & Operations Research 34, 2918-2930 (2007)
14. Pessoa, A., Poggi de Aragão, M., Uchoa, E.: Robust branch-cut-and-price algorithms for vehicle routing problems. To appear in: Golden, B., Raghavan S., Wasil, E. (eds.), The Vehicle Routing Problem: Latest Advances and New Challenges. Springer Verlag, New York (2008)
15. Pisinger, D., Ropke, S.: A general heuristic for vehicle routing problems. Computers & Operations Research 34, 2403–2435 (2007)
16. Righini, G., Salani, M.: New dynamic programming algorithms for the resource constrained elementary shortest path problem. Networks (2008), doi:10.1002/net.20212

17. Sariklis, D., Powell, S.: A heuristic method for the open vehicle routing problem. Journal of the Operational Research Society 51, 564-573 (2000)
18. Schrage, L.: Formulation and structure of more complex/realistic routing and scheduling problems. Networks 11, 229-232 (1981)
19. Tarantilis, C.D., Diakoulaki, D., Kiranoudis, C.T.: Combination of geographical information system and efficient routing algorithms for real life distribution operations. European Journal of the Operational Research 152, 437-453 (2004)
20. Tarantilis, C.D., Ioannou, G., Kiranoudis, C.T., Prastacos, G.P.: A threshold accepting approach to the open vehicle routing problem. RAIRO Operations Research 38, 345-360 (2004)
21. Tarantilis, C.D., Ioannou, G., Kiranoudis, C.T., Prastacos, G.P.: Solving the open vehicle routing problem via a single parameter metaheuristic algorithm. Journal of the Operational Research Society 56, 588-596 (2005)
22. Toth, P., Tramontani, A.: An integer linear programming local search for capacitated vehicle routing problems. To appear in: Golden, B., Raghavan S., Wasil, E. (eds.), The Vehicle Routing Problem: Latest Advances and New Challenges. Springer Verlag, New York (2008)
23. Toth P., Vigo D.: The Vehicle Routing Problem. SIAM Monographs on Discrete Mathematics and Applications, Philadelphia (2002)