J Heuristics (2014) 20:1–38
DOI 10.1007/s10732-013-9230-1

A GRASP + ILP-based metaheuristic for the capacitated location-routing problem

Claudio Contardo · Jean-François Cordeau · Bernard Gendron

Received: 30 August 2011 / Revised: 18 June 2013 / Accepted: 1 July 2013 /
Published online: 24 July 2013
© Springer Science+Business Media New York 2013

Abstract In this paper we present a three-phase heuristic for the Capacitated Location-Routing Problem. In the first stage, we apply a GRASP followed by local search procedures to construct a bundle of solutions. In the second stage, an integer-linear program (ILP) is solved taking as input the different routes belonging to the solutions of the bundle, with the objective of constructing a new solution as a combination of these routes. In the third and final stage, the same ILP is iteratively solved by column generation to improve the solutions found during the first two stages. The last two

C. Contardo (✉)
Département de management et technologie, ESG UQÀM, 315 rue Ste-Catherine Est, Montréal, QC H2X 3X2, Canada
e-mail: claudio.contardo@gerad.ca

C. Contardo
Groupe d'études et de recherche en analyse des décisions (GERAD), 3000 chemin de la Côte-Sainte-Catherine, Montréal , QC H3T 2A7, Canada

J.-F. Cordeau
Canada Research Chair in Logistics and Transportation, HEC Montréal, 3000 chemin de la Côte-Sainte-Catherine, Montréal , QC H3T 2A7, Canada
e-mail: jean-francois.cordeau@cirrelt.ca

B. Gendron
Département d'informatique et de recherche opérationnelle, Université de Montréal, C.P. 6128, succ. Centre-ville, Montréal , QC H3C 3J7, Canada
e-mail: bernard.gendron@cirrelt.ca

J.-F. Cordeau · B. Gendron
Centre interuniversitaire de recherche sur les réseaux d'entreprise la logistique et le transport (CIRRELT), C.P. 6128, succ. Centre-ville, Montréal, QC H3C 3J7, Canada

stages are based on a new model, the location-reallocation model, which generalizes the capacitated facility location problem and the reallocation model by simultaneously locating facilities and reallocating customers to routes assigned to these facilities. Extensive computational experiments show that our method is competitive with the other heuristics found in the literature, yielding the tightest average gaps on several sets of instances and being able to improve the best known feasible solutions for some of them.

Keywords Location-routing · Column generation · Metaheuristic

1 Introduction

In the capacitated location-routing problem (CLRP) we are given a set of potential facilities I and a set of customers J. To each facility $i \in I$ we associate a fixed setup cost f_i and a capacity b_i. To each customer $j \in J$ we associate a demand d_j. An unlimited, homogeneous fleet must be routed from the open facilities to serve the demand of the customers in J. To each vehicle is associated a capacity Q, and to every pair of nodes i and j is associated a traveling cost c_{ij}. The goal is to select a subset of facilities and to design vehicle routes around these facilities in order to (1) visit each customer once, (2) respect both vehicle and facility capacities and (3) minimize the total cost.

The CLRP is an $\mathcal{NP}$-hard combinatorial optimization problem since it generalizes two well known $\mathcal{NP}$-hard problems: the capacitated facility location problem (CFLP) and the capacitated vehicle routing problem (CVRP). Exact methods for this problem include branch-and-cut (Belenguer et al. 2011; Contardo et al. 2011) and column generation (Baldacci et al. 2011; Contardo et al. 2013). These methods are able to solve instances with up to 200 customers. However, some instances with 100 customers remain unsolved. To handle large size instances, Prins et al. (2007), Prodhon and Prins (2008), Prodhon (2009), Prodhon (2011) and Duhamel et al. (2010) proposed several metaheuristics. Among these, the method based on Lagrangean relaxation with cooperative granular tabu search is the most effective for handling large instances of the CLRP. This method combines the solution of an integer-linear program (ILP) (a CFLP) solved by Lagrangean relaxation (for location decisions) followed by a granular tabu search (for routing decisions). Pirkwieser and Raidl (2010) have introduced a variable neighborhood search (VNS) algorithm for the periodic CLRP (PLRP) and the CLRP based on the combination of a pure VNS with the solution of several ILPs. The ILPs they consider include a location model (a two-index CFLP) and a reallocation model (a set partitioning model). Hemmelmayr et al. (2012) have developed an adaptive large neighborhood search (ALNS) heuristic for the CLRP. In an ALNS method, several different neighborhoods are applied and ranked on the run according to their success in improving solutions. In subsequent iterations, the highest ranked neighborhoods have a larger probability of being chosen. Their algorithm is capable of improving the best known solutions on several instances. Finally, Yu et al. (2010) proposed a simulated annealing heuristic for the problem in which diversification is controlled by means

of a temperature parameter to allow the deterioration of the solution in the hope of escaping from local optima.

The main contributions of this paper are:

i. to introduce a new greedy randomized adaptive search procedure (GRASP) for the CLRP that is competitive with the GRASP proposed by Prins et al. (2006) and Duhamel et al. (2010) and which provides better average gaps on several sets of instances.
ii. to introduce a novel location-reallocation model that takes into account the location and the routing decisions simultaneously. The proposed model is based on a set-partitioning formulation that generalizes both the CFLP and the reallocation model of Franceschi et al. (2006), the first by adding the possibility of inserting customers in the middle of the routes, and the second by adding the possibility of reallocating whole routes to different facilities.

The location-reallocation model introduced here can also be seen as a restricted CLRP in which some routing decisions are fixed, and thus also inherits all of the cuts valid for the CLRP (Belenguer et al. 2011; Contardo et al. 2011). The addition of these extra cuts plays an important role in the proposed heuristic. Indeed, the strength of the model relies on the quality of the root relaxation lower bound. As a pure branch-and-cut-and-price algorithm is computationally too demanding, column generation is applied only at the root node, and even there by relying on some simple pricing heuristics. The resulting ILP is then solved by means of a general-purpose solver. Therefore, the strength of the linear relaxation lower bound is crucial for the performance of the algorithm.

The rest of the paper is organized as follows. In Sect. 2 we give a general description of our solution approach. In Sect. 3 we present two of the metaheuristics that are used in our algorithm, namely a GRASP and a local search procedure used to improve solutions. In Sect. 4 we introduce the location-reallocation model (LRM). We strengthen it with valid inequalities and describe the pricing algorithm used to derive columns of negative reduced cost. In Sect. 5 we introduce the two hybrid metaheuristics, namely a solution blender heuristic and a local improvement heuristic, both of which are based on the solution of the LRM. This is followed by computational results in Sect. 6 and by conclusions in Sect. 7.

2 An overview of the complete algorithm

In this section we give a general description of the different parts of our algorithm, and describe it by means of a pseudo-code. Our algorithm consists of four main procedures: a GRASP, a Local Search heuristic (LS), a Solution Blender (SB) and a Local Improvement Heuristic (LIH).

The first part of our algorithm is a new **GRASP** (Feo and Resende 1989) based on the randomization of the so-called Extended Clarke and Wright Savings Algorithm (ECWSA) introduced by Prins et al. (2006), a greedy insertion heuristic originally introduced for multiple-depot vehicle routing problems. In the GRASP paradigm, the greedy insertion heuristic is applied several times. Diversification is applied by

allowing sub-optimal movements (randomly chosen) during the insertion algorithm. At the end of each run, the constructed feasible solution is inserted into a pool of solutions $\mathcal{P}$. The details of our procedure will be given in Sect. 3.1.

The second part of our algorithm is **Local Search (LS)**. A LS procedure is an iterative algorithm that takes as input a feasible solution of the problem. At any iteration, it inspects the feasible solutions lying inside a neighborhood of the current solution and, if it finds a solution of lower cost, it replaces the current solution by the new one before starting the next iteration. Otherwise, it stops and returns the best solution found. In this article, we consider several types of neighborhoods, including neighborhoods involving customers and facilities. This will be discussed in more detail in Sect. 3.2.

The third part of our algorithm is what we call the **Solution Blender (SB)**, a method based on the solution of an integer-linear program, the location-reallocation model. The LRM is a set-partitioning model in which three types of variables are considered: location variables, assignment variables and routing variables. The first two are polynomial in number while there is an exponential number of the latter. Normally, such models are solved by column generation. However, in the SB heuristic the set of routing variables is restricted to contain a fixed number of columns defined in advance, and therefore no column generation is applied. The key of the SB is to combine two procedures into one. On the one hand, it solves the problem of re-assigning routes to facilities. This type of neighborhood was introduced by Prins et al. (2007) for the CLRP and later used by Pirkwieser and Raidl (2010). On the other hand, it solves the problem of combining routes from different solutions, which was introduced by Pirkwieser and Raidl (2010) for the CLRP. In the SB heuristic, these two neighborhoods are inspected at once, thus solving the reassignment problem of routes to facilities and the combination problem of routes belonging to different solutions, all at once. The details of the procedure are given in Sect. 5.1.

The fourth component of our method is what we call the **Local Improvement Heuristic (LIH)**, a *destroy-and-repair* method inspired from the ALNS metaheuristic (Røpke and Pisinger 2006). In this method, a destroy operator is applied to remove customers from the current solution. The LRM is then solved by column generation, with the aim of constructing a new feasible solution of better quality. The LIH uses a parameter $\Gamma \leq |J|$ in the destroy operators to remove a target number Γ of customers from the solution, which we denote by $LIH(\Gamma)$. The details of the LIH will be discussed in Sect. 5.2.

We now describe by means of a pseudo-code the complete algorithm. For a given solution $\mathcal{T}$ of the CLRP, let $v(\mathcal{T})$ denote the cost of $\mathcal{T}$. Also, let Γ_0 be a parameter representing a (usually small) number of customers.

3 Pure heuristics

In this section we describe the two pure heuristic procedures used in our algorithm, namely the GRASP and the LS methods. We refer to these as pure heuristics to distinguish them from the ILP-based heuristics that will be introduced later.

Algorithm 1 GRASP + ILP

1: Use GRASP + LS and build solution pool $\mathcal{P}$.
2: Use SB and add the newly found solutions to $\mathcal{P}$.
3: $\mathcal{T} \leftarrow \arg\min\{v(\mathcal{S}) : \mathcal{S} \in \mathcal{P}\}$.
4: $\Gamma \leftarrow \Gamma_0$.
5: **repeat**
6: Apply $LIH(\Gamma)$ to $\mathcal{T}$ and let $\mathcal{T}'$ be the solution obtained.
7: **if** $\mathcal{T}' \notin \mathcal{P}$ **then**
8: $\mathcal{P} \leftarrow \mathcal{P} \cup \mathcal{T}'$.
9: **if** $v(\mathcal{T}') < v(\mathcal{T})$ **then**
10: $\mathcal{T} \leftarrow \mathcal{T}'$ and **go to** 1.
11: **end if**
12: **end if**
13: Use SB and add the newly found solutions to $\mathcal{P}$.
14: **if** a new solution $\mathcal{T}'$ was found with $v(\mathcal{T}') < v(\mathcal{T})$ **then**
15: $\mathcal{T} \leftarrow \mathcal{T}'$ and **go to** 1.
16: **end if**
17: Increase Γ by some positive value.
18: **until** some stopping criterion is met

3.1 GRASP

GRASP is a popular metaheuristic which, based on some simple greedy deterministic criterion, includes some randomization to diversify the search of the solution space. This randomized greedy algorithm is applied several times, thus increasing the likelihood of identifying a good quality solution. The randomization is usually subject to what is called a restricted candidate list (RCL), for which a given greedy criterion of the form "$pick\, x' = \arg\min_x\{f(x) : x \in X\}$" is replaced by "*Let* $\mathcal{L}$ *contain the* κ *elements* $x \in X$ *with smallest value of* $f(x)$. *Pick* x' *randomly in* $\mathcal{L}$". For the CLRP, Prins et al. (2006) proposed a GRASP that they complemented with path relinking. Their method is based on the so-called Extended Clarke and Wright Savings Algorithm (ECWSA). In this paper we propose a variant of that method, and explain how we apply randomization at three different levels of the algorithm. Our method differs from that of Prins et al. (2006) mainly in the way the initial assignments of customers to facilities are performed. Our method penalizes the opening of a new facility by considering its opening cost in the evaluation of the assignment, which is not taken into account in the original version of the ECWSA. Our computational results show that the merging phase of the algorithm can take advantage of this new evaluation rule to find solutions of higher quality than the original GRASP introduced by Prins et al. (2006). We now describe, by means of a pseudo-code (Algorithm 2), the deterministic algorithm on which is based the proposed GRASP.

First, let us introduce some notation. A route R is represented by a sequence of nodes $(u_0, u_1, \ldots, u_n, u_{n+1} = u_0)$, with $u_0 = u_{n+1} \in I$ and $u_1, \ldots, u_n \in J$. For any two routes R, S and for any facility $i \in I$, $s(R, S, i)$ represents the saving produced when routes R and S are merged to create a new route T which is assigned to facility i, and such that capacities are respected. Note that if R and S contain two or more customers, four different mergings are possible, and so the definition

of $s(R, S, i)$ implicitly assumes that the resulting route T is the one with the lowest cost. For details on the merging procedure, the reader is referred to Clarke and Wright (1964) and to Prins et al. (2006). Also, for a Boolean statement p, we define δ_p to be equal to 1 if p is $true$, and 0 otherwise. Finally, F denotes the set of currently open facilities, $\gamma(\cdot)$ represent the facilities to which customers are assigned (unassigned customers are such that $\gamma(j) = -1$), and $l(\cdot)$ represent the current loads of facilities.

Algorithm 2 ECWSA

$F \leftarrow \emptyset, \gamma(j) \leftarrow -1$ for all $j \in J, l(i) \leftarrow 0$ for all $i \in I$.
while $\exists j \in J$ such that $\gamma(j) = -1$ **do**
  $j' \leftarrow \arg\min\{\sum_{i \in F} c_{ij} + 0.1 \sum_{i \notin F} c_{ij} : j \in J, \gamma(j) = -1\}$.
  $i' \leftarrow \arg\min\{2c_{ij'} + f_i \delta_{i \notin F} : i \in I, l(i) + d_{j'} \leq b_i\}$.
  $F \leftarrow F \cup \{i'\}, \gamma(j') \leftarrow i', l(i') \leftarrow l(i') + d_{j'}$.
end while
$\mathcal{R} \leftarrow \{(\gamma(j), j, \gamma(j)) : j \in J\}$.
repeat
  $(R', S', i') \leftarrow \arg\max\{s(R, S, i) : R, S \in \mathcal{R}, i \in I$, and merging respects capacities$\}$.
  $s \leftarrow s(R', S', i')$.
  if $s > 0$ **then**
    Merge R', S' into a new route T' and assign it to facility i'.
    Update $\mathcal{R}$ by replacing R' and S' by the merged route T'.
    Update F, γ and l accordingly.
  end if
until $s \leq 0$

In our GRASP, we replace the three optimization problems appearing in the pseudocode with some randomized variants.

The deterministic statement $j' \leftarrow \arg\min\{\sum_{i \in F} c_{ij} + 0.1 \sum_{i \notin F} c_{ij} : \gamma(j) = -1\}$ is changed to randomly picking a customer j' with among the five customers satisfying $\gamma(j) = -1$ with minimum value of $\sum_{i \in F} c_{ij} + 0.1 \sum_{i \notin F} c_{ij}$. The second term of this sum is particularly useful at the beginning of the algorithm for picking a customer close to most facilities.

The statement $i' \leftarrow \arg\min\{2c_{ij'} + f_i \delta_{i \notin F} : i \in I, l(i) + d_{j'} \leq b_i\}$ is decomposed into two random stages. For the set of closed facilities F^c (if any), we compute the quantity $v(F^c) = (\sum_{i \notin F} 2c_{ij'} + f_i)/|F^c|$ and assign to this quantity a dummy node i_{F^c}, and for each facility $i \in F$ we compute separately the quantity $v(i) = 2c_{ij'}$ and assign to it the node i. Now, we put in a list the $|F| + 1$ quantities defined before (only $|F|$ in case $|F^c| = 0$) and randomly pick a node i' among the three which minimize it. If $i' \in F$, then we assign customer j' to this facility. Otherwise, if $i' = i_{F^c}$ we randomly pick a facility $i'' \notin F$ among the $k = \lceil |I|/3 \rceil$ that minimize $2c_{i''j'} + f_{i''}$. Facility i'' is then opened and customer j' is assigned to it.

Finally, the statement $(R', S', i') \leftarrow \arg\max\{s(R, S, i) : R, S \in \mathcal{R}, i \in I$, and merging respects capacities$\}$ is modified to randomly pick a merging among the five possible mergings with maximum saving.

We call this algorithm the Randomized ECWSA (RECWSA). The RECWSA is repeated for 300 times, and the solutions are stored in a solution pool $\mathcal{P}$. For

each of the solutions in the pool, we apply LS (detailed in the next section) to improve its quality. After that, we clean the pool by keeping the 100 best solutions. These solutions will be the input of the SB heuristic which will be described in Sect. 5.1.

3.2 Local Search

Local search procedures are simple greedy algorithms applied to a feasible solution to further improve its quality. They are usually based on simple greedy criteria, which are fast to compute. In our case, we have implemented seven different LS procedures:

FACILITY OPEN	Compute the cost of opening a previously closed facility i and of re-assigning routes to this newly open facility. We potentially close a facility if it is cheaper to move all of its routes to the newly open one. This procedure is performed using a first-improvement criterion.
FACILITY SWAP	Swap an open facility with a closed one, and reassign routes from the first facility to the next. This procedure is performed using a first improvement criterion.
GIANT TOUR SPLIT	Merge all the routes linked to the same facility into one giant TSP tour (Salhi et al. 1992). Split the tour using a shortest path algorithm so as to minimize the total routing cost. This procedure is performed using a first-improvement criterion.
ROUTE SWAP	Swap two routes linked to different facilities. This procedure is performed using a first-improvement criterion.
2-OPT + 2-EXCHANGE	Swap any two customers whether they belong to the same route or not (Croes 1958; Savelsbergh 1992). This procedure is performed using a best-improvement criterion.
2-OPT*	Two routes are split and re-merged (Potvin and Rousseau 1995). This procedure is performed using a best-improvement criterion.
3-OPT	Pick three customers belonging to the same route and evaluate all possible swaps between them (Lin and Kernighan 1973). This procedure is performed using a first-improvement criterion.

Each of these procedures is performed repeatedly until no further improvements are detected. Also, the order in which each of the procedures is performed is as described above, and they are cyclically performed until no further improvements are found. This order is motivated by the following two observations. The first four procedures are sorted according to the potential impact of a successful move. This impact is in general larger for the first four methods and that is why they are performed first. Next, the last three movements involving the swapping of customers are sorted according to their computational complexity (with the fastest ones first).

4 A location-reallocation model

In this section we introduce the Location-Reallocation Model (LRM), a new ILP model that generalizes the CFLP and the reallocation model of Franceschi et al. (2006), the first by adding the routing decisions to the problem, and the second the location decisions. This model is the core of the ILP-based heuristics introduced in this paper, namely the SB and the LIH. We present a mathematical formulation of the model, some valid inequalities, and the pricing algorithm used in the column generation.

4.1 Mathematical formulation

Let us consider a feasible solution $\mathcal{T}$ of the CLRP. For a given customer subset $T \subseteq J$ let $\mathcal{T}(T)$ be the truncated solution of the CLRP obtained from $\mathcal{T}$ after

i. removing the customers of set T,
ii. removing the customers that are isolated (because they belonged to a route where all other customers are in T) and inserting them into T,
iii. short-cutting the remaining consecutive nodes in the routes,
iv. deleting the edges linking facilities to customers,
v. and relinking the two remaining endpoints of every route.

As a result, what we obtain is a set of closed subtours, each of which consisting of at least two customers. F1 illustrates this procedure. On the left side, circular dots represent customer locations, whereas square nodes represent facility locations. The nodes surrounded by dotted circles are the nodes in set T. The right side represents the subtours resulting from the removal of the customers in set T. Let us denote by $\mathcal{R}$ the set of these subtours and for each $r \in \mathcal{R}$ and $i \in I$ let $h(i, r)$ and $t(i, r)$ be the two consecutive nodes in r which, after linking r to i using these two nodes as endpoints, produce the route with the least possible cost. To avoid symmetries, we arbitrarily take $h(i, r), t(i, r)$ satisfying $h(i, r) < t(i, r)$. Customers in T must be reinserted back into $\mathcal{T}(T)$ and subtours $r \in \mathcal{R}$ must be assigned to facilities to construct a (eventually new) feasible solution of the CLRP. For every subtour r we let $E(r), V(r)$ be the sets of edges and customers in that subtour. We also let $c(r)$ be the routing cost of the subtour, and $q(r)$ be its load. For every $i \in I$ and subtour $r \in \mathcal{R}$ we define $E(i, r) = E(r) \setminus \{\{h(i, r), t(i, r)\}\}$. Let us denote, for a given facility i and subtour r the set of insertion points associated, $\mathcal{I}(i, r) = E(i, r) \cup \{\{i, h(i, r)\}, \{i, t(i, r)\}\}$. For each facility $i \in I$ we also consider an additional insertion point $\{i, i\}$ for full routes that are not extended from any subtour. We let $\mathcal{I}$ be the set of all possible insertion points.

Every insertion point $p \in \mathcal{I}$ is uniquely assigned to a facility $i(p)$ and to an edge $e(p)$. Also, every insertion point is either assigned to an unique subtour $r \in \mathcal{R}$ (in which case we denote $r(p) = r$) or to none (if $p = \{i, i\}$, in which case we denote $r(p) = -1$). For every insertion point p, we denote by $\mathcal{S}_p$ the set of sequences or partial paths that can be inserted in p, and we denote by $\mathcal{S} = \cup\{\mathcal{S}_p : p \in \mathcal{I}\}$ the set of all possible sequences. Note that a sequence that results in a violation of the capacities can be safely removed from $\mathcal{S}$. For every $s \in \mathcal{S}_p$ we let $E(s)$

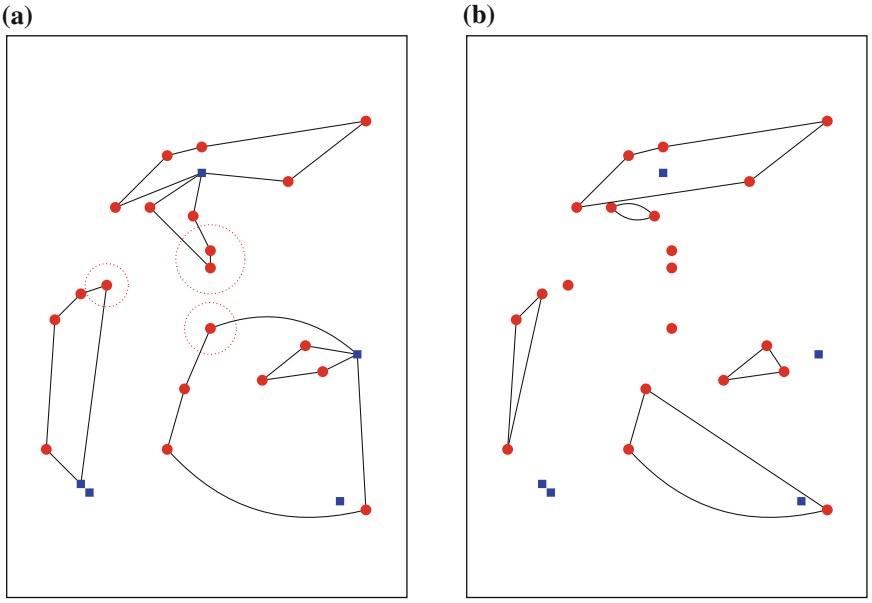

Fig. 1 (a) Complete solution. Set T surrounded by dotted circles (b) Incomplete solution after the removal of nodes in T Example of node removal from a CLRP solution

be the set of edges defining s, $q(s)$ be the load of s (without considering the two endpoints) and $c(s)$ be the cost associated to that partial route, computed as follows:

$$c(s) = \begin{cases} \sum_{e \in E(s)} c_e - c_{e(p)} & \text{if } p \in \cup_{i,r} E(i,r), s \in \mathcal{S}_p \\ \sum_{e \in E(s)} c_e & \text{otherwise.} \end{cases} \tag{1}$$

Let us define the following notation. Let z_i be a binary variable equal to 1 iff facility i is selected for opening. For every pair $\{i, j\}, i \in I, j \in T$ let y_{ij} be a binary variable equal to 1 iff customer j is served by a single-customer route from facility i. For every subtour $r \in \mathcal{R}$ and for every facility $i \in I$ let $u_{ir}^{\mathcal{R}}$ be a binary variable equal to 1 iff subtour r is assigned to facility i. For every facility $i \in I$ and customer $j \in T$ let u_{ij}^{T} be a binary variable equal to 1 iff customer j is served from facility $i \in I$. For every $s \in \mathcal{S}$ we let w_s be a binary variable equal to 1 iff sequence s (associated to a certain insertion point) is selected. The LRM is as follows:

$$\sum_{r \in \mathcal{R}} c(r) \quad + \quad \min \quad \sum_{i \in I} f_i z_i - \sum_{i \in I, r \in \mathcal{R}} c_{h(i,r)t(i,r)} u_{ir}^{\mathcal{R}}$$
$$+2 \sum_{i \in I, j \in J} c_{ij} y_{ij} + \sum_{s \in \mathcal{S}} c(s) w_s \tag{2}$$

subject to

$$\sum_{i \in I} u_{ij}^{T} = 1 \qquad j \in T \qquad (3)$$

$$\sum_{i \in I} u_{ir}^{\mathcal{R}} = 1 \qquad r \in \mathcal{R} \qquad (4)$$

$$y_{ij} + \sum_{p \in \mathcal{I}, i(p)=i} \sum_{s \in \mathcal{S}_p, j \in V(s)} w_s = u_{ij}^{T} \qquad i \in I, j \in T \qquad (5)$$

$$\sum_{s \in \mathcal{S}_{\{i,h(i,r)\}}} w_s = u_{ir}^{\mathcal{R}} \qquad i \in I, r \in \mathcal{R} \qquad (6)$$

$$\sum_{s \in \mathcal{S}_{\{i,h(i,r)\}}} w_s - \sum_{s \in \mathcal{S}_{\{i,t(i,r)\}}} w_s = 0 \qquad i \in I, r \in \mathcal{R} \qquad (7)$$

$$\sum_{s \in \mathcal{S}_p} w_s \leq u_{ir}^{\mathcal{R}} \qquad i \in I, r \in \mathcal{R}, p \in \mathcal{I}(i,r) \qquad (8)$$

$$\sum_{i \in I} \sum_{p \in \mathcal{I}(i,r)} \sum_{s \in \mathcal{S}_p} q(s) w_s \leq Q - q(r) \qquad r \in \mathcal{R} \qquad (9)$$

$$\sum_{j \in T} d_j u_{ij}^{T} + \sum_{r \in \mathcal{R}} q(r) u_{ir}^{\mathcal{R}} \leq b_i z_i \qquad i \in I \qquad (10)$$

$$z, y, u, w \text{ binary.} \qquad (11)$$

The objective function (2) contains two parts: a constant term given by the expression $\sum_{r \in \mathcal{R}} c(r)$, which takes into account the cost of the subtours remaining in the solution after the removal of the nodes in set T; and a linear term, combining setup costs with routing costs. Constraints (3)–(4) are the assignment constraints of customers to facilities. Constraints (5) are the degree constraints which ensure that customers in T will be reinserted. Constraints (6)–(7) ensure that partial routes $r \in \mathcal{R}$ will be linked to a facility. Constraints (8) ensure that for every insertion point $p \in \mathcal{I}(i,r)$ at most one column will be assigned. Moreover, if a route r is not assigned to a certain facility i, then all of the sequences $s \in \mathcal{S}_p$ with $i(p) = i$ and $r(p) = r$ are automatically set to 0. Constraints (9) are the vehicle capacity inequalities. They make sure that the final routes will not exceed vehicle capacities. Constraints (10) are the facility capacity inequalities. They make sure that the total demand assigned to every facility will not exceed its capacity, while at the same time no load will be assigned to closed facilities.

Note that the minimum sizes of the sequences s may vary. Indeed, a sequence s participates in the construction of multiple-customer routes, so every time we have to make sure that only routes containing two or more customers are generated. Thus, for $p \in \cup_{i,r} E(i,r)$, the minimum size of $s \in \mathcal{S}_p$ (defined as the number of nodes visited other than those of $e(p)$) is 1. If $p = \{i, i\}$ then the minimum size is 2. Finally, if $p \in \cup_{i,r} \{\{i, h(i,r)\}, \{i, t(i,r)\}\}$, then the minimum size is 0.

4.2 Valid inequalities

The location-reallocation problem described above includes a polynomial number of constraints and can be solved by means of branch-and-price. However, it is possible

to include all the valid inequalities from the three-index formulation (Contardo et al. 2011) after the inclusion of the following flow and assignment variables.

For every facility $i \in I$ and edge $e \in E$, let us define a flow variable x_e^i as follows:

$$x_e^i = \begin{cases} u_{ir}^{\mathcal{R}} - \sum_{s \in \mathcal{S}_e} w_s & \text{if } e \in E(i,r) \text{ for some } r \in \mathcal{R} \\ 1 - u_{ir}^{\mathcal{R}} & \text{if } e = \{h(i,r), t(i,r)\} \text{ for some } r \in \mathcal{R} \\ \sum_{p \in \mathcal{I}, i(p)=i} \sum_{s \in \mathcal{S}_p, e \in E(s)} w_s & \text{otherwise.} \end{cases} \quad (12)$$

Also, for every facility $i \in I$ and customer $j \in J$ let us define the following assignment variables:

$$u_{ij} = \begin{cases} u_{ir}^{\mathcal{R}} & \text{if } j \in V(r), r \in \mathcal{R} \\ u_{ij}^{T} & \text{if } j \in T. \end{cases} \quad (13)$$

Finally, in adition to variables y_{ij} for $i \in I, j \in T$, we define $y_{ij} = 0$ for all $i \in I, j \notin T$.

It suffices to use variables (x_e^i), (u_{ij}) and (y_{ij}) to include the valid inequalities from the three-index vehicle-flow formulation presented in Contardo et al. (2011). In particular, we found useful to include the following four families of inequalities: y-capacity cuts (y-CC), y-strengthened effective facility capacity inequalities (y-SEFCI), y-location-routing generalized large multistar inequalities (y-LRGLM), and disaggregated co-circuit constraints (DCoCC). For details on the inequalities, we refer to Belenguer et al. (2011) and Contardo et al. (2011). Moreover, it is possible to strengthen the y-CC and the y-ESFCI to hybrid forms of the y-strengthened capacity cuts (y-SCC) and y-set-partitioning strengthened effective facility capacity inequalities (y-SP-SEFCI), which have been developed by Contardo et al. (2013) for solving the CLRP by branch-and-cut-and-price.

4.3 Column generation

For each column w_s, its reduced cost will be computed differently depending on its insertion point p. Let $T(s) \subseteq T$ be the set of customers in T that are served by column s. Suppose that no additional inequalities have been added to the problem, and let $\alpha, \beta, \sigma, \gamma, \theta$ be the dual variables associated with constraints (5)–(9). The reduced cost associated to a column s with an insertion point $p \in \mathcal{I}$ is given by

$$\bar{c}(s) = \begin{cases} c(s) - \sum_{j \in T(s)} \alpha_j - \sum_{j \in T(s)} d_j \theta_{r(p)} - \gamma_p & \text{if } p \in \cup_{i,r} E(i,r) \\ c(s) - \sum_{j \in T(s)} \alpha_j - \beta_{ir(p)} - \sigma_{ir(p)} & \text{if } p \in \cup_{i,r} \{\{i, h(i,r)\}\} \\ c(s) - \sum_{j \in T(s)} \alpha_j + \sigma_{ir(p)} & \text{if } p \in \cup_{i,r} \{\{i, t(i,r)\}\} \\ c(s) - \sum_{j \in T(s)} \alpha_j & \text{if } p \in \cup_i \{i, i\}. \end{cases} \quad (14)$$

If valid inequalities have been added during the solution of the problem, the reduced costs are modified accordingly using the dual variables associated to these inequalities. Our pricing algorithms take into account the different expressions in (14) (modified

by the dual information associated to valid inequalities) but they work along the exact same principle. The expressions in (14) also imply the following: A different pricing algorithm must be performed for each possible insertion point p. In our implementation, the insertion points are sequentially considered without giving any particular preference to any of them. For each insertion point p, the pricing is performed in two stages.

First, we use a simple tabu search heuristic starting from a column containing a single customer. That customer is chosen in such a way that the reduced cost of the resulting column is as small as possible. We consider four neigborhoods to inspect the solution space around a given sequence. An ADD neighborhood picks a customer not in the sequence and inserts it into the sequence. A DROP neighborhood is used to perform the opposite move. A SWAP neighborhood picks a customer inside the current sequence and one outside, and swaps them. Finally, a SWITCH neighborhood takes two customers inside the sequence and swaps them. We combine neighborhoods ADD, DROP, SWAP and SWITCH using the customers in set T. The neighborhoods are sorted and applied in the following order: ADD-DROP-ADD-SWAP-ADD-SWITCH. Indeed, preliminary experiments showed that the ADD neighborhhod is often the most useful, and thus it is the one that is performed the most. The movements use a best-improvement criterion, and a tabu list forbids movements to positions previously visited during the last three iterations. The algorithm stops whenever a column of negative reduced cost has been detected or when a maximum number of iterations has been reached. The maximum number of iterations at the beginning is set to 100. In order to accelerate the pricing algorithms, after seven rounds of cut generation, we lower this threshold to 20.

When the tabu search procedure finishes with success (i.e., after having identified a column with negative reduced cost), starting from that column we apply a greedy insertion algorithm, similar to the one presented by Franceschi et al. (2006). We evaluate the insertion of every single customer in a list $\mathcal{L}$ initially containing the customers in T not yet inserted into the column at every possible position. If the resulting column has negative reduced cost, then it is added to a pool and the same algorithm is recursively applied to it. This dynamic programming algorithm is applied until it reaches a depth of 5 from the starting column (the one obtained by the tabu search procedure).

5 ILP-based metaheuristics

In this section we describe two hybrid metaheuristics based on the solution of the LRM described earlier. We first describe the SB heuristic, a method based on the existence of a pool of reasonably good solutions. We then describe the LIH based on the iterative solution of the LRM and solved by column and cut generation.

5.1 Solution blender

We use a heuristic procedure based on the solution of a particular case of the LRM to combine routes belonging to different solutions. Given a pool of solutions $\mathcal{P}$, we apply the following procedure to every solution $\mathcal{S} \in \mathcal{P}$. Let $\mathcal{R}(\mathcal{S})$ be the set of routes

describing solution $\mathcal{S}$. For every route $R \in \mathcal{R}(\mathcal{S})$ we first consider the subtour produced by disconnecting R from its facility and then reconnecting its two endpoints. This tour is then reconnected to every facility i using as endpoints the pair of consecutive nodes in the subtour that produces the route with minimum cost. This procedure creates, for every route $R \in \mathcal{R}(\mathcal{S})$, $|I|$ routes, each connected to a different facility. We refer to this procedure as the *replication step*.

At the end of the replication step, we will potentially have $\sum_{\mathcal{S}\in\mathcal{P}} |\mathcal{R}(\mathcal{S})| \times |I|$ routes (some repeated routes might be discarded). The LRM is then solved using $T = J$ and by restricting the set of columns to contain those constructed during the replication step, without applying any column generation. The optimal solution of this restricted problem is then likely to combine routes from different solutions. Indeed, in many cases in which the GRASP procedure was not able to find a near optimal solution, the blending phase performed substantially better. In our case, the input for the SB is the solution pool $\mathcal{P}$ containing the 100 best solutions found by the GRASP combined with LS. Every new solution found is also subject to LS. Note that the SB may fail to find a feasible solution to the problem (even though all solutions in $\mathcal{P}$ are feasible for the SB), in which case the best solution in $\mathcal{P}$ is returned.

Our SB heuristic is an implementation of the one proposed by Pirkwieser and Raidl (2010) for the Periodic LRP (PLRP), which is inspired from a similar procedure introduced by Rochat and Taillard (1995) and recently used in other heuristic methods for vehicle routing problems (Subramanian et al. 2013). The main difference of our procedure with respect to that of Pirkwieser and Raidl (2010) is the use of valid inequalities to 1) speed-up the solution of the problem, and 2) guide the local branching (to be described in the next paragraph) towards fixing the right subset of location variables. Moreover, the SB generalizes the reallocation heuristic used in Prins et al. (2007) and Pirkwieser and Raidl (2010) based on the solution of a CFLP, and aimed to decide the optimal assignments to facilities of the routes belonging to a single solution.

At the end of the root node relaxation, we perform a local branching heuristic to guide the search towards promising directions during the branch-and-bound search. We fix to 1 the location variables whose values are greater than or equal to 0.9. For the location variables taking values smaller than or equal to 0.1, we pick the two variables z_{i_1}, z_{i_2} with the smallest reduced costs. If two or more variables have the same reduced cost, priority is given to those with the largest values of z_i. For these two location variables we impose the following constraint:

$$z_{i_1} + z_{i_2} \leq 1. \tag{15}$$

The remaining location variables satisfying $z_i \leq 0.1$ are all fixed to zero.

5.2 Local improvement heuristic

Let $\mathcal{T}$ be the solution with minimum cost resulting from the previous heuristic procedures. Let $\rho = \lceil 0.1|J| \rceil$ be a parameter. For different values of $k > 0$, we let $\Gamma = k\rho$ be the target size of customer set T to be removed from and reinserted back in $\mathcal{T}(T)$. The local improvement phase starts with $\mathcal{T}$ and $k = 1$, and successively

solves the LRM using sets T of target size $k\rho$. Each time a better solution is found, the algorithm is restarted with the same value of k. When no more improvement can be detected, k is increased by one unit and the algorithm is restarted. The value of k is increased at most twice, and each update of this value corresponds to a *major iteration* of the LIH. Note that every newly found solution is subject to LS. Also, at the end of a *major iteration* the SB heuristic is run using a restricted set of solutions stored in the pool (the best 50, eventually including the new found solutions). Indeed, we have found that after the several heuristics applied beforehand and the new solutions found, the SB takes no significant advantage in considering the initial pool size of 100.

Note that because of the heuristic column generation and the local branching heuristic, the resulting integer problem may be infeasible or the solver may be unable to find a feasible solution. In this case, the current feasible solution is stored and used in the following iterations.

In what follows we describe the different parts of this procedure, namely the choice of the customer set T, the inclusion of an initial pool of columns as well as some local branching rules.

5.2.1 Choice of set T

The set T of customers to be erased from $\mathcal{T}$ is selected by following similar rules to those explained in Franceschi et al. (2006) and Pirkwieser and Raidl (2010). We first define the following notion of relatedness between two customers. Let $u, v \in J$ be two customer nodes. Let $c_{max} = \max\{c_{hj} : h, j \in J\}$ be the maximum distance between any two customers. We define the relatedness between u and v as $r(u, v) = 1 - c_{uv}/c_{max}$. If u and v belong to the same route then $r(u, v)$ is multiplied by 0.75, and if they belong to the same facility then $r(u, v)$ is multiplied by 0.85. The idea is to penalize the choice of customers belonging to the same route or being served by the same facility, as the LS makes it unlikely that these customers will switch places. The two rules that we have implemented can be summarized as follows:

NEIGHBORHOOD rule: Given a pivot customer $\overline{u}$, we make $T = \{\overline{u}\}$ and iteratively insert into T the customer $u \notin T$ such that $\sum_{v \in T} r(u, v)$ is maximal.

RANDOM rule: We randomly pick a subset of customers and insert it into T.

We first apply the NEIGHBORHOOD rule five times. The first time that it is performed, we choose as pivot the customer $\overline{u}$ that maximizes $\sum_{v \in J \setminus \{\overline{u}\}} r(\overline{u}, v)$. We save in a list N_T the customers that have participated in T in the previous iterations. For the next iteration, we use as pivot node the customer $u \notin N_T$ such that $\sum_{v \neq u, v \notin N_T} r(u, v)$ is maximal. When the NEIGHBORHOOD rule has been used five times without improving the current solution, we use the RANDOM rule five more times. If using any of the two rules the current solution is improved, the counters are reset to zero and the LIH is restarted.

5.2.2 Initial set of columns

We have found it is beneficial to start the column generation algorithm with a small, but likely useful set of initial columns. For every insertion point p, we let $V(p) \subseteq T$ be the subset of customers of size $\min\{5, |T|\}$ containing the closest nodes to $e(p)$, in terms of the sum of the distances to the two endpoints of $e(p)$. Then, we add to the master problem all the sequences obtained as combinations of the nodes in $V(p)$.

5.2.3 Local Branching

Let I^o, I^c the subsets of facilities that are open or closed in solution $\mathcal{T}$. From the beginning of the optimization we let

$$\sum_{i \in I^o} z_i - \sum_{i \in I^c} z_i \geq |I^o| - \eta.$$

Depending on the value of Γ, the parameter η is set either to 2 (if $\Gamma = \rho$) or 0 (if $\Gamma \geq 2\rho$). In the first case, we let at most two location variables change their values, while in the second case the location variables are actually fixed to their current values in $\mathcal{T}$. When the root node relaxation has been solved with success and no more columns with negative reduced cost or violated inequalities are detected, we also consider the same local branching constraint as for the SB (see constraints (15)).

6 Computational experiments

We have run our method on an Intel Xeon E5462, 3.0 Ghz processor with 16GB of memory. The code was compiled with the Intel C++ compiler v11.0 and executed on Linux, kernel 2.6. Linear and integer programs were solved with CPLEX 12.5. The algorithm has been tested on four sets of instances from the literature, containing a total of 89 instances. The first set of instances ($\mathcal{F}_1$) has been developed by Belenguer et al. (2011) and contains 30 instances with capacitated vehicles and facilities. The second set of instances ($\mathcal{F}_2$) has been introduced by Tuzun and Burke (1999) and contains 36 instances with capacitated vehicles and uncapacitated facilities. The third set of instances ($\mathcal{F}_3$) has been adapted from other vehicle routing problems by Barreto (2004) and contains 19 instances with capacitated vehicles, mixing some instances with capacitated and uncapacitated facilities. The fourth and last set of instances ($\mathcal{F}_4$) has been introduced by Baldacci et al. (2011) and contains four instances with limited vehicle capacities and uncapacitated facilities. The dimensions of the instances vary from very small instances with 12 customers and two facilities up to very large instances with 200 customers and 20 facilities.

For the parameter setting, several runs have been performed on the four sets of instances. At the end, however, we use the same parameters for all instances and the average values reported correspond to those obtained on a total of 10 runs for each instance. The calibrated parameters are reported in Table 1.

In Tables 2, 3, 4 and 5 we report the results obtained by our algorithm on all sets of instances. In these tables, z^*_{BKS} corresponds to the best known solution as reported

Table 1 Values for the calibrated parameters

GRASP	
# iterations	300
SB	
# solns (after GRASP)	100
# solns (after LIH 1, 2, 3)	50
max time (sec.)	300
LIH 1	
ρ	$\lceil 0.1\|J\| \rceil$
Γ	ρ
# RANDOM	5
# NEIGHBORHOOD	5
η	2
max time on each it. (sec.)	180
LIH 2	
ρ	$\lceil 0.1\|J\| \rceil$
Γ	2ρ
# RANDOM	5
# NEIGHBORHOOD	5
η	0
max time on each it. (sec.)	180
LIH 3	
ρ	$\lceil 0.1\|J\| \rceil$
Γ	3ρ
# RANDOM	5
# NEIGHBORHOOD	5
η	0
max time on each it. (sec.)	180
CPLEX	
Version	12.5
Internal cutting planes	No
Probing	No
Scaling	No
Node search strategy	Feasibility
Variable selection strategy	Strong branching

by previous authors, z^*_{avg} is the average cost obtained by our solution method, *stdev* is the standard deviation (in %) of the cost over the 10 runs, gap_{avg} is the average relative gap (in %), computed as $100 \times (z^*_{avg} - z^*_{BKS})/z^*_{BKS}$, T_{avg} is the average CPU time, in seconds, over the 10 runs, and z^*_{best} is the best solution found in these 10 runs. This value does not necessarily correspond to the best known solution found by our method during the parameter setting phase, which is reported later in Table 12. Finally, gap_{best} is the relative gap (in %) of the best solution found, computed as $100 \times (z^*_{best} - z^*_{BKS})/z^*_{BKS}$. When the best found solution is strictly better than the

Table 2 Results on instances of set $\mathcal{F}_1$

Instance	z^*_{BKS}	z^*_{avg}	*stdev*	gap_{avg}	T_{avg}	z^*_{best}	gap_{best}
ppw-20x5-1a	54,793	54,793	0.00	0.00	1.7	54,793	0.00
ppw-20x5-1b	39,104	39,104	0.00	0.00	2.6	39,104	0.00
ppw-20x5-2a	48,908	48,908	0.00	0.00	1.5	48,908	0.00
ppw-20x5-2b	37,542	37,542	0.00	0.00	2.8	37,542	0.00
ppw-50x5-1a	90,111	90,111	0.00	0.00	15.0	90,111	0.00
ppw-50x5-1b	63,242	63,281	0.19	0.06	18.4	63,242	0.00
ppw-50x5-2a	88,298	88,333	0.12	0.04	17.5	88,298	0.00
ppw-50x5-2b	67,308	67,436	0.13	0.19	22.0	67,373	0.10
ppw-50x5-2bis	84,055	84,055	0.00	0.00	21.0	84,055	0.00
ppw-50x5-2bbis	51,822	51,898	0.02	0.15	27.3	51,883	0.12
ppw-50x5-3a	86,203	86,203	0.00	0.00	16.6	86,203	0.00
ppw-50x5-3b	61,830	61,853	0.09	0.04	22.9	61,830	0.00
ppw-100x5-1a	274,814	275,628	0.05	0.30	220.4	275,457	0.23
ppw-100x5-1b	213,615	214,785	0.16	0.55	230.2	214,056	0.21
ppw-100x5-2a	193,671	194,054	0.12	0.20	121.9	193,708	0.02
ppw-100x5-2b	157,095	157,311	0.13	0.14	100.0	157,178	0.05
ppw-100x5-3a	200,079	200,394	0.03	0.16	97.3	200,339	0.13
ppw-100x5-3b	152,441	152,814	0.38	0.24	100.1	152,466	0.02
ppw-100x10-1a	287,983	292,657	3.02	1.62	2, 621.8	**287,892**	−0.03
ppw-100x10-1b	231,763	236,026	0.55	1.84	1, 067.2	234,080	1.00
ppw-100x10-2a	243,590	243,851	0.11	0.11	236.1	243,695	0.04
ppw-100x10-2b	203,988	204,253	0.15	0.13	258.5	203,988	0.00
ppw-100x10-3a	250,882	253,610	0.17	1.09	723.3	252,927	0.82
ppw-100x10-3b	204,317	205,110	0.18	0.39	584.4	204,664	0.17
ppw-200x10-1a	477,248	477,656	0.26	0.09	3, 960.4	**475,327**	−0.40
ppw-200x10-1b	378,065	378,656	0.21	0.16	4, 006.0	**377,327**	−0.20
ppw-200x10-2a	449,571	449,797	0.07	0.05	4, 943.0	**449,291**	−0.06
ppw-200x10-2b	374,330	374,996	0.08	0.18	3, 486.0	374,575	0.07
ppw-200x10-3a	469,433	471,272	0.19	0.39	4, 075.1	469,870	0.09
ppw-200x10-3b	362,817	363,581	0.12	0.21	7, 887.6	363,103	0.08
Average			0.22	0.28	1, 162.9		0.08

Bold values indicate that a new best solution has been found

best known solution, its value is marked in bold characters. As the results show, our solutions are 0.28% above the best known solutions on average for the instances of set $\mathcal{F}_1$, 0.45% for the instances of set $\mathcal{F}_2$, 0.62% for the instances of set $\mathcal{F}_3$ and 0.33% for the instances of set $\mathcal{F}_4$. Moreover, we are able to improve these values in 12 out of the 89 instances considered in our study. Regarding the CPU times, they lie around 45 min on average, and usually stay below 3 h.

In Tables 6, 7 , 8 and 9 we report the evolution of our algorithm during the different stages. In these tables, instances are grouped according to their size. The headers

Table 3 Results on instances of set $\mathcal{F}_2$

Instance	z^*_{BKS}	z^*_{avg}	*stdev*	gap_{avg}	T_{avg}	z^*_{best}	gap_{best}
P111112	1,467.7	1,475.5	0.34	0.53	198.5	1,469.4	0.12
P111122	1,449.2	1,452.0	0.15	0.20	579.8	1,449.2	0.00
P111212	1,394.8	1,405.8	0.38	0.79	219.6	1,394.9	0.01
P111222	1,432.3	1,440.6	0.48	0.58	754.7	1,432.3	0.00
P112112	1,167.2	1,176.2	0.33	0.77	278.0	1,169.1	0.16
P112122	1,102.2	1,103.6	0.11	0.13	633.6	1,102.4	0.01
P112212	791.7	795.8	0.49	0.53	226.5	791.7	0.01
P112222	728.3	728.5	0.02	0.02	550.4	728.3	0.00
P113112	1,238.5	1,239.6	0.04	0.09	285.7	1,238.5	0.00
P113122	1,245.3	1,246.3	0.08	0.08	645.6	1,245.3	0.00
P113212	902.3	902.8	0.13	0.06	230.6	902.3	0.00
P113222	1,018.3	1,018.3	0.00	0.00	748.9	1,018.3	0.00
P131112	1,866.8	1,924.1	0.83	3.07	1,639.9	1,899.9	1.78
P131122	1,823.5	1,831.0	0.24	0.41	3,611.7	1,825.3	0.10
P131212	1,965.1	1,969.3	0.16	0.21	1,274.5	**1,964.3**	−0.04
P131222	1,796.5	1,800.3	0.24	0.21	3,099.4	**1,792.8**	−0.20
P132112	1,443.3	1,450.4	0.27	0.49	871.3	1,446.8	0.24
P132122	1,434.6	1,447.2	0.25	0.88	2,738.3	1,443.9	0.65
P132212	1,204.4	1,205.9	0.05	0.12	2,081.6	1,204.8	0.03
P132222	931.0	931.9	0.05	0.10	3,734.0	931.3	0.03
P133112	1,694.2	1,703.8	0.52	0.57	937.8	1,695.9	0.10
P133122	1,392.0	1,401.5	0.16	0.68	2,751.2	1,398.0	0.43
P133212	1,198.3	1,199.6	0.06	0.11	1,009.8	1,198.6	0.03
P133222	1,151.8	1,158.7	0.08	0.60	3,559.6	1,157.3	0.48
P121112	2,251.9	2,251.3	0.27	−0.03	2,805.5	**2,243.4**	−0.38
P121122	2,159.9	2,154.9	0.52	−0.23	5,679.9	**2,138.4**	−0.99
P121212	2,220.0	2,226.1	0.37	0.27	3,004.6	**2,209.3**	−0.48
P121222	2,230.9	2,241.7	0.34	0.48	6,143.1	**2,225.1**	−0.26
P122112	2,073.7	2,093.8	0.55	0.97	3,462.4	2,077.8	0.20
P122122	1,692.2	1,704.4	0.36	0.72	8,546.8	1,694.8	0.16
P122212	1,453.2	1,467.8	0.14	1.01	3,470.9	1,465.4	0.84
P122222	1,082.7	1,086.0	0.15	0.30	5,292.0	1,082.9	0.01
P123112	1,960.3	1,968.7	0.37	0.43	3,865.3	**1,954.7**	−0.29
P123122	1,918.9	1,936.2	0.18	0.90	9,366.7	1,931.1	0.63
P123212	1,762.0	1,766.2	0.18	0.24	3,766.3	1,763.1	0.06
P123222	1,391.7	1,392.7	0.03	0.07	5,156.8	1,392.0	0.03
Average			0.25	0.45	2,589.5		0.10

Bold values indicate that a new best solution has been found

Table 4 Results on instances of set $\mathcal{F}_3$

Instance	z^*_{BKS}	z^*_{avg}	*stdev*	gap_{avg}	T_{avg}	z^*_{best}	gap_{best}
Perl-12x2	203.98	203.98	0.00	0.00	0.3	203.98	0.00
Gas-21x5[a]	424.90	424.90	0.00	0.00	1.7	424.90	0.00
Gas-22x5[a]	585.11	585.11	0.00	0.00	2.9	585.11	0.00
Min-27x5[a]	3, 062.02	3, 062.02	0.00	0.00	3.5	3, 062.02	0.00
Gas-29x5[a]	512.10	512.10	0.00	0.00	5.4	512.10	0.00
Gas-32x5[a]	562.22	562.26	0.00	0.01	6.2	562.22	0.00
Gas-32x5b[a]	504.33	504.33	0.00	0.00	7.9	504.33	0.00
Gas-36x5[a]	460.37	460.64	0.18	0.06	8.6	460.37	0.00
Chr-50x5ba	565.62	577.41	0.42	2.08	17.1	574.66	1.60
Chr-50x5be[a]	565.60	584.87	1.19	3.41	17.7	569.49	0.69
Perl-55x15	1, 112.06	1, 112.40	0.06	0.03	47.4	1, 112.06	0.00
Chr-75x10ba	844.40	847.06	0.33	0.31	87.9	844.58	0.02
Chr-75x10be	848.85	850.56	0.20	0.20	97.8	848.85	0.00
Chr-75x10bmw	802.08	809.55	0.55	0.93	100.0	802.08	0.00
Perl-85x7	1, 622.50	1, 627.31	0.08	0.30	81.8	1, 625.84	0.21
Das-88x8[a]	355.78	356.10	0.12	0.09	209.6	355.78	0.00
Chr-100x10[a]	833.43	849.99	0.85	1.99	492.0	840.67	0.87
Min-134x8[a]	5, 709.00	5, 801.92	0.68	1.63	750.2	5, 719.25	0.18
Das-150x10[a]	43, 963.60	44, 263.49	0.35	0.68	1, 842.1	**43, 952.30**	−0.03
Average			0.26	0.62	199.0		0.19

Bold values indicate that a new best solution has been found
[a] Aggregate results reported in Table 11 are based on these instances

Table 5 Results on instances of set $\mathcal{F}_4$

Instance	z^*_{BKS}	z^*_{avg}	*stdev*	gap_{avg}	T_{avg}	z^*_{best}	gap_{best}
M-n150x14a	1, 352.93	1, 354.45	0.09	0.11	1, 546.2	1, 353.46	0.04
M-n150x14b	1, 212.46	1, 218.45	0.17	0.49	1, 486.9	1, 213.78	0.11
M-n199x14a	1, 644.35	1, 646.49	0.12	0.13	3, 807.8	1, 644.35	0.00
M-n199x14b	1, 480.43	1, 489.02	0.24	0.58	3, 762.9	1, 483.22	0.19
Average			0.15	0.33	2, 650.9		0.08

GRASP, SB, LIH 1, 2, 3 stand for the different parts of our algorithm, including the three major iterations of the LIH. The sub-headers gap_{avg} and T_{avg} stand for the average relative gap (in %, computed as before) and the average CPU time spent in seconds. In general, the SB is a very effective method for reducing the gap with respect to the solutions found during the GRASP. However, the GRASP should not be underestimated, since the behaviour of the SB depends on the good quality of the routes found by the GRASP. For the LIH, it is worth observing that for instances of set $\mathcal{F}_1$ the first improvement alone is able to reduce the gap by one half. Subsequent iterations of the improvement stage are able to reduce the gap by smaller margins. Depending

Table 6 Algorithm evolution for instances of set $\mathcal{F}_1$

Instances	GRASP		SB		LIH 1		LIH 2		LIH 3	
	gap_{avg}	T_{avg}	gap_{avg}	T_{avg}	gap_{avg}	T_{avg}	gap_{avg}	T_{avg}	gap_{avg}	T_{avg}
ppw-20x5	0.00	0.4	0.00	0.6	0.00	0.8	0.00	1.2	0.00	2.1
ppw-50x5	0.64	7.5	0.12	8.5	0.10	10.6	0.07	14.1	0.06	20.1
ppw-100x5	1.49	64.0	0.41	86.7	0.35	96.8	0.30	111.2	0.26	145.0
ppw-100x10	3.16	76.0	2.59	132.2	1.48	290.8	1.03	517.7	0.86	915.2
ppw-200x10	1.46	968.2	0.71	1, 629.9	0.39	2, 315.1	0.25	3, 111.6	0.18	4, 726.3
Average	1.39	223.7	0.77	372.1	0.47	543.5	0.33	752.0	0.28	1, 162.9

Table 7 Algorithm evolution for instances of set $\mathcal{F}_2$

Instances	GRASP		SB		LIH 1		LIH 2		LIH 3	
	gap_{avg}	T_{avg}	gap_{avg}	T_{avg}	gap_{avg}	T_{avg}	gap_{avg}	T_{avg}	gap_{avg}	T_{avg}
100x10	2.21	115.5	0.67	123.1	0.55	141.7	0.48	174.0	0.46	239.8
100x20	1.76	179.9	0.37	201.5	0.28	267.5	0.24	392.1	0.17	652.2
150x10	3.10	653.2	1.17	715.0	1.00	780.7	0.83	913.1	0.76	1, 302.5
150x20	2.97	916.1	0.71	1, 038.0	0.60	1, 304.5	0.52	2, 031.8	0.48	3, 249.1
200x10	3.28	1, 856.2	1.17	2, 232.1	0.80	2, 441.4	0.65	2, 755.5	0.48	3, 395.8
200x20	3.48	2, 706.2	0.80	3, 050.6	0.61	3, 484.5	0.47	4, 459.6	0.37	6, 697.6
Average	2.80	1, 071.2	0.82	1, 226.7	0.64	1, 403.4	0.53	1, 787.7	0.45	2, 589.5

Table 8 Algorithm evolution for instances of set $\mathcal{F}_3$

Instances	GRASP		SB		LIH 1		LIH 2		LIH 3	
	gap_{avg}	T_{avg}	gap_{avg}	T_{avg}	gap_{avg}	T_{avg}	gap_{avg}	T_{avg}	gap_{avg}	T_{avg}
$\leq$50 custs	1.29	2.5	0.79	2.7	0.70	3.3	0.60	4.8	0.56	7.1
>50 custs	2.87	135.3	1.24	142.7	0.97	235.2	0.82	310.9	0.68	412.1
Average	2.04	65.4	1.00	69.0	0.83	113.2	0.71	149.8	0.62	199.0

Table 9 Algorithm evolution for instances of set $\mathcal{F}_4$

Instances	GRASP		SB		LIH 1		LIH 2		LIH 3	
	gap_{avg}	T_{avg}	gap_{avg}	T_{avg}	gap_{avg}	T_{avg}	gap_{avg}	T_{avg}	gap_{avg}	T_{avg}
150 custs	6.11	1, 058.5	0.47	1, 160.7	0.39	1, 220.1	0.34	1, 319.5	0.30	1, 516.6
199 custs	6.30	2, 723.7	0.56	3, 057.6	0.44	3, 192.8	0.38	3, 415.2	0.36	3, 785.3
Average	6.21	1, 891.1	0.51	2, 109.2	0.41	2, 206.5	0.36	2, 367.4	0.33	2, 650.9

on the needs of the decision maker, the improvement phase can be extended to more iterations or reduced to fewer, compensating the time saved or added with the quality of the solutions obtained.

Table 10 Sensitivity of method GRASP+SB to the number of iterations/solutions considered

SB SOLNS	GRASP ITS					
	150		300		450	
	gap	*T*	*gap*	*T*	*gap*	*T*
50	0.98	242.1	0.79	356.4	0.79	470.44
100	1.00	256.0	0.77	372.1	0.72	474.84
150	0.99	296.9	0.79	408.1	0.71	517.52

Table 11 Comparison against other heuristics

Method	$\mathcal{F}_1$		$\mathcal{F}_2$		$\mathcal{F}_3^a$	
	gap	*T*	*gap*	*T*	*gap*	*T*
GRASP+PR[b]	3.60	96.5	3.42	159.56	1.49	21.15
MA|PM[b]	1.38	76.7	1.78	203.13	2.01	37.8
LRGTS[b]	0.74	17.5	1.76	21.24	1.64	18.21
GRASP+ELS[c]	1.07	65.2	1.22	606.6	0.02	187.7
VNS+ILP[d]	0.86	6.7	–	–	–	–
SALRP[b]	0.41	422.4	1.41	826.4	0.27	140.46
ALNS	0.70	451.0	0.81	830.0	0.15	174.75
GRASP	1.39	223.7	2.80	1, 071.2	1.98	86.2
GRASP+SB	0.77	372.1	0.82	1, 226.7	1.13	89.9
GRASP+ILP	0.28	1, 162.9	0.45	2, 589.5	0.65	279.0

[a]Restricted to instances marked with Superscript 'a' in Table 4
[b]Results reported on a single run
[c]Best results after 5 runs
[d]Instances in $\mathcal{F}_2$ and $\mathcal{F}_3$ not tested

Table 12 New best known solutions

Instance	z^*_{BKS}	z^*_{NEW}	Instance	z^*_{BKS}	z^*_{NEW}
ppw-100x10-1a	287,983	**287,695**	P131122	1, 823.5	**1, 823.2**
ppw-100x10-1b	231,763	**230,989**	P131212	1, 965.1	**1, 964.3**
ppw-200x10-1a	477,248	**475,294**	P131222	1, 796.5	**1, 792.8**
ppw-200x10-1b	378,065	**377,043**	P133212	1, 198.3	**1, 198.2**
ppw-200x10-2a	449,571	**449,115**	P121112	2, 251.9	**2, 243.4**
ppw-200x10-2b	374,330	**374,280**	P121122	2, 159.9	**2, 138.4**
ppw-200x10-3b	362,817	**362,653**	P121212	2, 220.0	**2, 209.3**
Das-150x10	43,963.6	**43,952.3**	P121222	2, 230.9	**2, 222.9**
			P123112	1, 960.3	**1, 954.7**

Bold values indicate that a new best solution has been found

In Table 10 we report the performance of the sub-method GRASP+SB (i.e., without including LIH) for different settings of GRASP and SB. The objective of this experiment is to assess the sensitivity of GRASP+SB to the addition or limitation of time

and memory resources. In addition to the calibrated GRASP performed during 300 iterations, we also run it for 150 and 450 iterations. Also, method SB is not only run by taking the 100 best solutions of the GRASP, but also considering 50 and 150 solutions. We have run method GRASP+SB for each possible setting (nine in total), on the instances of family $\mathcal{F}_1$. Each instance is solved 10 times, and the average gaps (in %) and CPU times (in seconds) are computed and reported. As one can see, the method is sensitive to the number of iterations performed by GRASP and the number of solutions considered in method SB. The simplest setting (GRASP for 150 iterations and SB using 50 solutions) provides the worst gaps but is also the fastest, while the more complex setting (GRASP for 450 iterations and SB using 150 solutions) seems to take advantage of the additional time and memory resources to obtain better solutions on average.

In Table 11 we compare our algorithm against several of the most recent heuristics developed for the CLRP. The algorithms considered are: GRASP+PR (Prins et al. 2006), MA|PM (Prodhon and Prins 2008), LRGTS (Prins et al. 2007), GRASP+ELS (Duhamel et al. 2010), VNS+ILP (Pirkwieser and Raidl 2010), SALRP (Yu et al. 2010) and ALNS (Hemmelmayr et al. 2012). Note that average results are not available for all these methods, some of them reporting results on single runs or the best results after several runs. Therefore, direct comparisons may be in many cases biased. In the last three rows of this table we report average results obtained by our method. Row GRASP corresponds to our GRASP, GRASP+SB to the addition of the SB and GRASP+ILP to the whole method, including the three major iterations of the LIH. The set of instances $\mathcal{F}_4$ has not been considered by any of the previous heuristics and is therefore not included in this table. As shown in the table, our algorithm is able to obtain the tightest average gaps for sets $\mathcal{F}_1$ and $\mathcal{F}_2$, and competitive average gaps on instances of set $\mathcal{F}_3$, getting better average results than GRASP+PR, MA|PM and LRGTS but outperformed by SALRP and ALNS. On the other hand, algorithms LRGTS and VNS+ILP take much less CPU time, but they seem to be less robust than our method in terms of solution quality. Additionally, our GRASP is able to obtain better solutions than that developed by Prins et al. (2006) for instances of families $\mathcal{F}_1$ and $\mathcal{F}_2$. Finally, note that by only applying our GRASP algorithm and the SB, we already obtain very competitive gaps, usually better than the previous approaches except for SALRP on instances of set $\mathcal{F}_1$ and for SALRP and ALNS on instances of set $\mathcal{F}_3$. In this discussion we have omitted comparisons against GRASP+ELS (Duhamel et al. 2010) since they only report best results after 5 runs, therefore any comparison to their method would be biased.

Finally, in Table 12 we report (in bold characters) the new best known feasible solutions found by our algorithm. Note that these solutions were not necessarily found during the 10 runs of our method, but rather during the calibration of several parameters. In total, our algorithm was able to improve the solutions on 17 out of the 89 instances considered in this study.

7 Concluding remarks

In this paper we have introduced a new heuristic method for the CLRP based on a GRASP followed by the iterative solution of a new ILP model, the location-reallocation

model (LRM). The GRASP introduced in this paper provides better solutions than the previous approach of Prins et al. (2006) for most of the instances considered in this study. We have introduced the location-reallocation model that generalizes the CFLP and the RM of Franceschi et al. (2006) by simultaneously determining the locations of facilities as well as the reallocation of customers and routes to those facilities. We use a IP-based method, the solution blender (SB), that takes as input a set of solutions for the CLRP and solves the LRM to find near optimal solutions. Indeed, by only applying our GRASP followed by the SB we obtain gaps that are competitive with the methods found in the literature. We complement this by applying a local improvement heuristic (LIH) based on the iterative solution of the LRM solved by column and cut generation. This LIH was found to be very effective in tightening the optimality gap. Finally, we were able to improve the best known feasible solutions on 17 out of the 89 instances considered in this study. As an avenue of future research, we believe that this heuristic can be adapted to solve some generalizations of the CLRP, such as the two-echelon CLRP (2E-CLRP) or the two-echelon CVRP (2E-CVRP), important problems arising in the operation of city-logistics systems.

Acknowledgments We thank the three anonymous referees and the Associate Editor for their helpful comments and suggestions that contributed to improve the quality of the article. We also thank the Canadian Natural Sciences and Engineering Research Council (NSERC) for its financial support.

Appendix

Detailed results for different seetings of GRASP+SB

In this appendix we provide a detailed comparison of the performance of method GRASP+SB (i.e., without considering the LIH) for different settings of GRASP and SB on the instances of set $\mathcal{F}_1$. Table 10 is based on the average values reported in Table 13. Each gap and CPU time reported corresponds to the average on 10 runs of method GRASP+SB.

New best solutions found

In this appendix we provide the best solutions found by our algorithm (including the calibration phase) as reported in Table 12. In each table, the first lines report the loads (q) and fixed costs ($cost$) of the open facilities (f). The following lines report the loads (q), costs ($cost$), facilities (f) and customers ($customers$) associated with each route. The customers are listed in their order of visit (See Tables 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, and 30).

Table 13 Detailed results for different settings of GRASP+SB

Instance	G-150xSB-50		G-150xSB-100		G-150xSB-150		G-300xSB-50		G-300xSB-100		G-300xSB-150		G-450xSB-50		G-450xSB-100		G-450xSB-150	
	gap	t	gap	t	gap	t	gap	t	gap	t	gap	t	gap	t	gap	t	gap	t
ppw-20x5-1a	0.00	0.3	0.00	0.4	0.00	0.4	0.00	0.5	0.00	0.6	0.00	0.7	0.00	0.6	0.00	0.7	0.00	0.9
ppw-20x5-1b	0.00	0.3	0.00	0.4	0.00	0.4	0.00	0.6	0.00	0.7	0.00	0.7	0.00	0.9	0.00	0.9	0.00	1.0
ppw-20x5-2a	0.00	0.2	0.00	0.3	0.00	0.3	0.00	0.3	0.00	0.4	0.00	0.6	0.00	0.5	0.00	0.6	0.00	0.7
ppw-20x5-2b	0.00	0.3	0.00	0.3	0.00	0.3	0.00	0.6	0.00	0.6	0.00	0.7	0.00	0.8	0.00	0.9	0.00	1.0
ppw-50x5-1a	0.00	3.7	0.00	4.2	0.00	5.1	0.00	7.1	0.00	7.7	0.00	8.4	0.00	10.4	0.00	10.9	0.00	11.8
ppw-50x5-1b	0.47	3.9	0.42	4.2	0.16	4.9	0.14	7.4	0.17	7.9	0.10	8.3	0.12	11.1	0.02	11.4	0.04	12.1
ppw-50x5-2a	0.12	3.3	0.08	4.1	0.00	5.0	0.00	6.2	0.04	7.0	0.00	7.7	0.00	9.2	0.00	9.7	0.00	10.8
ppw-50x5-2b	0.86	3.5	0.72	4.0	0.47	4.7	0.58	6.7	0.45	7.2	0.39	7.8	0.64	9.8	0.42	10.4	0.36	11.1
ppw-50x5-2bis	0.00	4.9	0.00	5.6	0.00	6.5	0.00	9.6	0.00	10.2	0.00	10.9	0.00	14.0	0.00	14.7	0.00	15.4
ppw-50x5-2bbis	0.15	4.3	0.16	5.1	0.15	6.0	0.15	7.9	0.15	8.8	0.15	9.6	0.15	11.6	0.16	12.4	0.15	13.3
ppw-50x5-3a	0.13	4.7	0.00	5.8	0.00	6.8	0.01	8.6	0.00	9.8	0.00	11.1	0.01	12.6	0.00	13.5	0.00	15.0
ppw-50x5-3b	0.22	4.6	0.12	5.2	0.20	6.1	0.15	8.9	0.12	9.7	0.09	10.4	0.11	13.1	0.01	13.7	0.05	14.7
ppw-100x5-1a	0.38	53.8	0.32	75.8	0.29	102.0	0.41	104.4	0.32	140.9	0.29	149.3	0.34	139.1	0.29	162.9	0.24	203.2
ppw-100x5-1b	0.88	54.6	0.69	69.9	0.67	91.5	0.75	98.0	0.66	120.9	0.59	150.3	0.82	144.4	0.65	168.2	0.51	184.1
ppw-100x5-2a	0.72	35.9	0.44	82.9	0.39	164.2	0.50	61.6	0.31	84.9	0.34	233.8	0.48	89.3	0.36	122.7	0.42	232.8
ppw-100x5-2b	0.52	28.4	0.44	32.0	0.34	38.3	0.39	56.1	0.28	59.0	0.20	62.2	0.25	82.3	0.24	85.0	0.22	88.3
ppw-100x5-3a	0.37	31.4	0.22	35.6	0.15	38.5	0.35	53.5	0.23	59.5	0.14	67.3	0.40	77.6	0.22	80.6	0.19	86.8
ppw-100x5-3b	1.32	27.9	0.95	30.5	0.67	32.4	1.08	51.9	0.66	54.8	0.55	56.6	1.00	77.2	0.84	78.8	0.74	82.7
ppw-100x10-1a	6.67	46.9	7.09	52.9	7.32	75.8	4.91	85.4	5.98	92.2	5.89	117.6	5.94	125.5	4.66	131.7	4.34	220.4
ppw-100x10-1b	8.02	47.4	9.76	51.2	9.62	60.7	6.35	90.8	6.09	92.7	7.21	98.5	6.30	133.7	6.50	137.4	6.15	142.7
ppw-100x10-2a	0.40	67.0	0.20	86.3	0.15	112.3	0.43	102.5	0.14	102.1	0.14	147.1	0.39	132.7	0.23	158.0	0.08	165.9
ppw-100x10-2b	0.60	47.9	0.53	73.9	0.19	73.1	0.29	80.4	0.22	90.5	0.22	105.6	0.43	118.3	0.26	125.3	0.18	143.0

Table 13 continued

Instance	G-150xSB-50		G-150xSB-100		G-150xSB-150		G-300xSB-50		G-300xSB-100		G-300xSB-150		G-450xSB-50		G-450xSB-100		G-450xSB-150	
	gap	t	gap	t	gap	t	gap	t	gap	t	gap	t	gap	t	gap	t	gap	t
ppw-100x10-3a	2.15	175.7	2.48	46.7	2.54	66.1	2.02	339.0	2.20	142.0	2.33	100.5	1.57	435.7	2.34	116.6	2.22	136.3
ppw-100x10-3b	1.16	105.2	0.88	190.9	0.86	387.7	0.96	122.4	0.89	273.5	0.70	371.0	0.91	143.8	0.68	230.7	0.80	460.9
ppw-200x10-1a	0.34	988.0	0.35	1, 071.4	0.91	1, 303.9	0.20	1, 325.8	0.64	1, 430.5	0.07	1, 685.3	0.25	1, 717.9	0.37	1, 794.8	0.65	2, 029.5
ppw-200x10-1b	1.11	971.9	1.23	1, 043.6	1.23	1, 081.3	1.26	1, 418.7	0.99	1, 449.0	1.00	1, 487.2	0.98	1, 816.1	0.67	1, 836.5	0.94	1, 886.6
ppw-200x10-2a	0.45	1, 042.4	0.52	1, 108.5	0.83	1, 334.8	0.39	1, 454.1	0.39	1, 530.4	0.80	1, 759.9	0.47	1, 877.6	0.36	1, 940.3	0.57	2, 157.6
ppw-200x10-2b	0.61	1, 007.6	0.68	1, 072.6	0.88	1, 106.2	0.69	1, 446.6	0.67	1, 500.5	0.80	1, 538.1	0.75	1, 911.1	0.71	1, 935.4	0.73	1, 977.8
ppw-200x10-3a	0.72	1, 208.1	0.76	1, 215.4	0.70	1, 403.9	0.83	1, 791.2	0.58	1, 862.7	0.68	2, 024.8	0.69	2, 387.3	0.46	2, 399.0	0.56	2, 546.5
ppw-200x10-3b	1.14	1, 288.4	1.09	1, 301.5	1.09	1, 389.1	0.88	1, 946.8	1.00	2, 006.0	1.05	2, 010.3	0.76	2, 608.9	1.00	2, 641.6	1.15	2, 672.7
Average	0.98	242.08	1.00	256.04	0.99	296.95	0.79	356.45	0.77	372.09	0.79	408.08	0.79	470.44	0.72	474.84	0.71	517.52

Table 14 New solution for instance ppw-100x10-1a with cost 287695

q	*cost*	*f*	*customers*
490	47,865	104	
560	47,995	105	
560	59,082	110	
51	2,870	104	44 54 14
70	6,057	104	35 23 19 48
70	7,314	104	82 72 51 12
70	4,431	104	69 8 1 86 73
26	1,524	104	77 31
70	6,093	104	41 61 25 40
64	9,079	104	93 27 99 53 9
69	4,137	104	84 2 4 63
67	6,933	105	42 62 89 49
57	3,936	105	52 38 47
70	6,654	105	79 34 59 65 83
68	4,687	105	13 7 17 71
69	5,463	105	67 16 15 80
70	5,081	105	88 87 5 50
69	4,626	105	66 60 10 3
29	2,272	105	39 30
61	6,275	105	45 91 46 95
34	1,765	110	11 76
70	9,947	110	37 26 64 22
64	4,166	110	75 68 58 81
69	3,606	110	29 70 6 98
64	3,707	110	43 33 28 94
67	4,939	110	55 85 74 18
69	7,718	110	90 92 96 24 100
69	6,329	110	56 20 36 78 97
54	3,144	110	32 21 57

Table 15 New solution for instance ppw-100x10-1b with cost 230989

q	*cost*	*f*	*customers*
490	47,865	104	
560	47,995	105	
560	59,082	110	
149	9,457	104	41 40 25 93 27 26 53 99 9 37
148	5,492	104	39 4 80 3 10 60 66 2 84
44	1,958	104	63 31 77
149	7,971	104	73 23 19 12 51 72 82 87 88
111	5,171	105	47 52 5 50 38 13
150	7,571	105	67 16 49 89 62 42 7 17 71
150	7,620	105	79 15 45 91 95 46 34 59 65 83
149	4,820	105	30 44 14 69 8 48 1 35 54
139	5,791	110	32 57 21 55 85 18 33 43
140	8,915	110	74 100 24 61 64 22 96 92 90
131	4,102	110	11 94 29 70 58 68 75 28
150	7,179	110	76 98 97 78 86 36 20 81 6 56

Table 16 New solution for instance ppw-200x10-1a with cost 475294

q	*cost*	*f*	*customers*
1,190	106,139	201	
758	71,504	202	
1,150	76,197	206	
70	3,899	201	177 196 60 188 32
67	4,962	201	75 180 199 77 28
65	3,461	201	69 191 70 129
62	2,050	201	62 33 186 156
65	3,721	201	74 120 11 6
66	7,543	201	38 128 72 110
70	5,318	201	89 41 125 66
68	7,111	201	113 58 167 82
62	5,544	201	170 136 146 168
68	3,849	201	131 18 157 155
70	6,683	201	107 34 142 21 44
69	5,046	201	87 45 31 108
66	5,807	201	178 162 79 93
70	3,569	201	134 49 116 153 175
65	4,365	201	139 182 126 200
65	2,868	201	164 163 92 64
69	4,249	201	187 81 185 179 20
53	1,683	201	27 112 94
68	5,388	202	176 101 96 85 172
69	5,473	202	67 193 86 43 183
67	2,713	202	10 98 14 138
70	5,925	202	194 149 147 118 55
70	3,470	202	19 59 154 56
70	5,468	202	117 122 95 68
68	3,868	202	88 84 2 189
69	5,605	202	97 152 91 144
68	5,082	202	133 57 130 16 46
70	3,783	202	151 123 23 160 158
69	3,388	202	26 8 150 22 124
67	3,234	206	51 166 61 198
70	4,041	206	114 9 171 143
65	5,620	206	145 161 76 127
70	5,601	206	173 90 29 65
69	6,197	206	174 37 109 78 50
69	3,698	206	115 181 119 63 165
66	7,011	206	73 25 141 4
70	5,522	206	30 35 36 148
68	6,329	206	190 17 197 121

Table 16 continued

q	*cost*	*f*	*customers*
68	9,116	206	80 3 53 54 5
68	3,455	206	47 105 40 106 100
63	3,207	206	104 103 71 48
70	5,499	206	184 13 99 137 140
70	7,391	206	135 1 192 39
59	2,534	206	42 169 7 12
69	2,450	206	83 102 111 52
69	8,658	206	132 195 159 15 24

Table 17 New solution for instance ppw-200x10-1b with cost 377043

q	*cost*	*f*	*customers*
1,183	106,139	201	
872	71,504	202	
1,043	76,197	206	
147	3,269	201	27 62 164 64 92 163 33 186 156
149	4,880	201	129 108 31 11 120 6 74 155 94
144	4,764	201	134 157 131 49 116 153 32 175 177 112
147	5,731	201	139 196 182 200 126 81 187 185 179 20
150	7,925	201	44 38 128 72 110 167 58 15 34
149	8,920	201	113 21 142 82 53 54 5 192 159 66
147	6,187	201	89 170 107 41 125 136 146 168 18
150	5,369	201	45 75 180 199 77 87 28 70 191 69
147	4,747	202	26 138 124 84 88 16 130 151 123 160
139	6,468	202	56 154 194 149 178 172 85 96 176
146	6,099	202	19 59 101 188 60 43 183 86 193 67
142	4,009	202	46 2 189 22 150 8 14 98 10
149	6,322	202	144 91 152 65 29 97 90 133 57
149	6,381	202	158 23 68 117 95 122 76 147 118 55
149	4,054	206	51 12 169 105 47 40 173 106 114 100
146	6,259	206	71 140 50 78 4 190 137 99 13 184
150	7,274	206	42 145 161 127 37 162 93 79 109 174
150	4,332	206	111 198 115 119 148 61 102 83 166
149	7,724	206	141 25 73 195 132 80 17 197 121 63
150	7,801	206	36 35 39 3 1 135 24 30 181
149	4,688	206	52 165 48 103 104 143 171 9 7

Table 18 New solution for instance ppw-200x10-2a with cost 449115

q	*cost*	*f*	*customers*
895	113,643	201	
1,229	74,099	204	
977	92,628	208	
62	1,717	201	9 8 15 66
70	3,396	201	44 59 94 35
60	3,381	201	39 1 53 64
69	3,000	201	69 18 98 32 46
63	3,517	201	54 23 95 56
70	2,825	201	77 17 82 49 42
66	2,547	201	75 51 33 57
67	3,630	201	37 88 84 14 65
68	2,967	201	60 16 28 13
68	3,818	201	20 45 93 85
59	4,901	201	43 90 38 78
66	4,641	201	97 48 58 62 52
48	1,744	201	7 30 80
59	2,814	201	26 100 11 61
70	4,360	204	29 83 27 40
68	6,304	204	164 161 178 173
68	3,350	204	135 112 139 120
70	5,320	204	74 4 12 81
67	2,808	204	99 31 6 63 70
69	4,614	204	124 151 160 172 185
69	3,282	204	71 73 195 101
68	2,323	204	133 186 141 180
65	4,375	204	19 55 24 87
68	4,112	204	34 5 41 79
66	3,608	204	47 91 36 25
69	5,464	204	22 76 92 72 10
68	3,353	204	123 111 179 107
68	2,627	204	137 128 121 181
68	3,155	204	50 67 96 3
70	4,005	204	89 2 68 21 86
70	3,668	204	113 122 115 188 134
68	3,863	204	193 200 131 157
49	1,484	208	146 126 149
60	3,226	208	175 156 118 177
70	2,827	208	130 192 163 184 155
61	3,138	208	136 154 104 169
68	5,246	208	191 109 197 125 183
62	4,097	208	168 171 110 189

Table 18 continued

q	*cost*	*f*	*customers*
70	4,501	208	103 196 102 127
68	2,547	208	166 162 182 129
66	2,499	208	165 159 176 145
69	5,050	208	140 142 116 143 105
64	4,687	208	148 117 150 174 132
70	3,506	208	138 190 144 106 158
66	3,937	208	152 153 108 198
65	2,890	208	147 194 167 170
69	3,621	208	119 114 199 187

Table 19 New solution for instance ppw-200x10-2b with cost 374280

q	*cost*	*f*	*customers*
885	113,643	201	
1,245	74,099	204	
971	92,628	208	
147	3,781	201	15 8 66 61 26 100 11 69 18 98
146	4,060	201	44 32 59 94 35 64 53 1 39
146	5,903	201	93 85 43 90 38 78 97 48 58 62
149	3,653	201	60 13 45 20 28 16 52 7 30
150	3,281	201	80 75 51 57 33 21 68 2 89 42
147	3,757	201	49 82 56 95 23 54 46 17 77 9
150	4,010	204	180 157 131 200 135 112 139 189 120
149	6,715	204	124 151 160 173 178 161 164 110 172 185
143	4,009	204	128 122 107 179 134 115 188 111 123
150	3,633	204	71 3 96 25 36 91 47 63 70
111	2,459	204	141 186 133 113 181 121 137
150	4,525	204	50 67 79 41 5 34 73 195 101
101	3,684	204	99 6 31 87 19 74 86
147	5,596	204	29 12 4 84 14 65 88 37 55 24
144	5,510	204	83 27 40 22 76 92 81 72 10
118	2,899	208	165 159 176 163 184 192 155 145
148	4,874	208	154 148 150 117 127 102 140 103 171 168
145	2,940	208	149 126 138 182 129 177 162 166 130
149	4,720	208	198 108 153 199 187 114 132 174 119
119	4,500	208	158 156 118 193 175 106 144 190
143	5,505	208	196 105 142 143 116 197 125 109 183 191
149	3,896	208	169 104 136 152 170 167 194 147 146

Table 20 New solution for instance ppw-200x10-3b with cost 362653

q	*cost*	*f*	*customers*
1,114	82,534	204	
977	79,185	206	
986	72,941	210	
150	7,157	204	1 59 22 36 41 4 51 46 21
148	6,932	204	115 97 178 131 188 182 195 169 81
141	5,530	204	85 74 148 143 157 167 173 199
150	9,071	204	155 193 192 137 154 160 152 135 190 186
148	9,366	204	187 170 168 174 200 144 172 191 158 194
150	11,043	204	197 150 136 156 139 145 159 185 142 133 151
146	6,219	204	198 184 166 163 141 176 161 181 165 132
81	2,125	204	84 108 95 78 113
93	3,661	206	47 13 23 49 48 63
150	5,369	206	31 38 53 57 42 37 65 10 34 62
147	6,198	206	7 61 35 16 25 8 14 44 29 27
150	6,991	206	2 20 58 30 52 24 60 45 56 33
143	6,292	206	3 177 175 180 153 171 149 140 179
146	6,132	206	147 146 189 196 164 138 162 183 134
148	4,298	206	19 18 15 26 17 12 64 11 55 9
146	5,096	210	123 79 82 105 126 103 101 88 70
145	3,931	210	91 129 125 75 83 100 66 93 114 67
149	5,366	210	69 111 76 68 92 89 77 110 121 94
145	3,729	210	71 80 124 116 128 127 109 112 130
149	4,689	210	98 86 99 119 104 87 102 117 107 73
146	6,274	210	6 28 32 5 39 54 43 50 40
106	2,524	210	90 118 96 120 106 72 122

Table 21 New solution for instance Das-150x10 with cost 43952.3

q	*cost*	*f*	*customers*
28,366,790	5,000	151	
27,894,615	5,000	156	
21,706,980	5,000	159	
7,393,809	0.00	151	1
5,036,110	861.556	151	70 84 111 47 48 76 54 43 87 93 137 15 75 114
7,978,160	4,051.73	151	73 149 147 56 28 96 13 148 86 132 69 136 3 119 62 120 145
7,958,711	6,512.53	151	124 51 79 98 103 77 91 36 117 61 150 24 71 37 100 60 134 78 22 29 143 80 125
6,969,974	1,132.42	156	6 109 2 49 95 50 99 138 102
7,960,791	2,089.64	156	68 128 59 106 88 104 131 5 115 17 46 31 44 23 107 140 130
7,037,516	1,048.44	156	40 139 26 144 89 72 16 35 39 116 30 81 55 32 90 38
5,926,334	4,828.51	156	122 141 12 101 112 34 135 10 41 20 133 118 108

Table 21 continued

q	*cost*	*f*	*customers*
7,354,925	2,307.61	159	9 18 45 110 53 126 8 105 63 19 65 27
6,778,743	2,267.29	159	142 94 123 83 121 113 7 11 97 42 33 74 82 85 129 66
7,573,312	3,852.61	159	57 14 67 127 52 21 146 92 64 25 4 58

Table 22 New solution for instance P131122 with cost 1823.2

q	*cost*	*f*	*customers*
533	100	155	
440	100	160	
571	100	163	
696	100	169	
115	86.9	155	46 93 39 146 143 114 142 64
144	101.3	155	44 123 15 2 128 137 107 140 14 49
127	64.0	155	76 136 7 105 61 147 86 52 89
147	106.2	155	55 22 82 28 100 65 29 37 150 6 63
144	100.0	160	9 59 72 70 115 85 134 101 34
148	63.5	160	5 127 148 97 131 62 77 79 35 42
148	74.0	160	138 75 81 56 57 20 24 4 31 45
145	60.1	163	88 111 120 132 87 90 30 60 47
145	87.6	163	67 78 18 119 102 129 139 16 48
139	115.8	163	54 36 125 98 74 10 124 25 149
142	97.1	163	73 40 145 118 96 110 19 23 108
148	96.9	169	11 126 53 8 43 133 69 112 144 33
125	67.3	169	27 91 104 130 32 38 12 122 99
136	101.0	169	21 66 50 121 71 141 135 58 117
139	90.8	169	103 113 83 94 80 17 13 106 26
148	110.7	169	51 84 41 3 92 95 116 109 68 1

Table 23 New solution for instance P131212 with cost 1964.3

q	*cost*	*f*	*customers*
842	100	156	
677	100	158	
734	100	160	
149	85.5	156	116 17 18 19 1 88 143 58 83 99
148	79.9	156	48 106 107 129 123 43 59 66 78 54
141	141.5	156	132 50 141 114 110 96 144 51 124
150	138.3	156	14 7 103 5 16 60 27 37 146 149
141	118.4	156	121 40 104 113 57 137 81 101 91
113	85.9	156	3 79 34 31 118 148 74
92	74.3	158	11 145 92 24 53 13

Table 23 continued

q	*cost*	*f*	*customers*
149	85.4	158	2 126 30 6 63 102 12 135 25
147	167.5	158	112 8 45 125 95 46 147 67 84 134 133
145	74.8	158	128 29 139 94 127 86 21 111 68 117
144	117.3	158	15 23 4 119 77 41 38 9 108
150	128.2	160	69 142 76 70 75 44 122 36 33
149	96.5	160	80 49 64 22 85 32 73 47 98 150
141	92.1	160	90 109 89 39 138 42 65 97 105 52
148	96.8	160	71 140 131 93 87 100 82 61 20 130
146	81.8	160	62 55 56 26 120 115 136 28 35 10 72

Table 24 New solution for instance P131222 with cost 1792.8

q	*cost*	*f*	*customers*
842	100	154	
721	100	165	
644	100	168	
116	57.2	154	43 108 76 141 118 37 14 55
141	101.2	154	87 97 127 59 42 88 117 109 21 36
150	129.7	154	101 15 144 9 20 81 57 26 39 96
141	79.5	154	92 91 80 130 8 27 142 41 61 24
148	104.6	154	12 107 119 116 123 52 150 128 85 82
146	94.1	154	100 106 126 3 110 4 90 131 99
142	72.3	165	147 33 140 79 48 70 16 122
145	113.8	165	77 105 145 86 60 46 56 98 62 34 94
148	103.7	165	49 45 137 143 63 66 6 40 64 111 67
148	75.5	165	53 58 115 133 74 129 135 146 22 112
138	78.9	165	32 1 139 104 124 13 35 132 65 5
150	117.1	168	134 25 103 23 89 19 114 136 50 68
136	73.9	168	121 11 95 17 138 29 10 120 93 51
76	75.1	168	102 125 78 31 30
141	109.7	168	38 75 69 2 83 148 113 18 54
141	106.3	168	71 73 47 44 149 7 84 72 28

Table 25 New solution for instance P133212 with cost 1198.2

q	*cost*	*f*	*customers*
1,079	100	151	
468	100	155	
730	100	159	
135	35.3	151	68 72 64 89 63 84 73 65 80
149	71.8	151	50 60 33 37 47 59 46 52 55 34 45
149	89.0	151	97 106 104 105 100 108 110 96 119
144	93.1	151	109 93 94 98 111 120 101 103 118

Table 25 continued

q	*cost*	*f*	*customers*
150	86.1	151	95 107 114 116 115 102 99 112 117 113 62
138	35.1	151	70 69 66 75 79 76 88 67
68	23.0	151	82 91 90 78 61
146	29.7	151	71 77 86 87 81 74 83 92 85
145	40.4	155	142 133 141 146 122 145 138 135 149
150	50.3	155	143 123 124 137 121 144 132 150 129 126
144	42.7	155	130 136 148 140 147 128 134 131 127
29	19.8	155	139 125
144	59.7	159	8 32 7 23 6 29 12 3 25 11
150	61.2	159	56 49 57 39 40 48 58 27 13
142	68.3	159	42 43 35 54 51 41 44 53 36 38
144	45.9	159	1 5 20 17 28 16 18 9 19 30
150	46.8	159	4 10 24 14 15 2 31 26 21 22

Table 26 New solution for instance P121112 with cost 2243.4

q	*cost*	*f*	*customers*
1,147	100	206	
1,134	100	209	
708	100	210	
113	62.5	206	4 5 116 80 52 85 122 181
149	88.0	206	26 157 127 99 140 87 92 49 8 198
141	109.7	206	45 130 94 51 126 170 193 11 19 199
148	128.9	206	197 183 154 150 31 137 175 67 21 113
149	132.6	206	182 91 48 37 192 28 160 186 57 102 40
148	101.6	206	108 56 168 16 78 132 43 149 114 29
150	124.5	206	55 100 98 64 82 185 58 144 65 83
149	90.9	206	128 138 161 77 155 105 194 15 189 190
138	63.3	209	133 135 163 191 7 148 169 158
143	75.8	209	47 38 17 166 146 119 34 167 39 134
148	93.9	209	12 184 33 97 120 129 109 71 53 6
126	65.6	209	195 142 69 84 187 177 152 143
146	86.2	209	2 9 121 79 139 46 96 66 200
140	82.9	209	68 103 86 75 153 59 136 174 162
150	125.3	209	104 124 61 164 110 196 90 112 115 54
143	103.5	209	18 10 27 62 24 63 159 60 101 188
133	104.3	210	32 145 88 22 70 50 30 111 171
150	85.5	210	172 81 44 156 125 123 165 151 131 106
139	60.5	210	74 118 73 42 107 180 14 93 117
136	85.9	210	36 95 13 178 41 176 141 1 76
150	72.1	210	147 179 23 35 3 173 25 72 89 20

Table 27 New solution for instance P121122 with cost 2138.4

q	*cost*	*f*	*customers*
880	100	204	
863	100	205	
402	100	210	
884	100	214	
147	89.4	204	54 129 80 140 127 21 49 73 195 27 43
143	75.6	204	146 6 187 174 67 191 143 131 186 12
149	114.8	204	155 95 94 193 117 53 35 2 8 144 110
149	90.1	204	29 170 125 139 159 132 50 28 197 75
148	79.6	204	135 74 116 162 134 44 163 147 24
144	69.4	204	23 112 196 138 56 161 104 160 114
137	71.9	205	177 83 113 16 32 41 180 181 90
149	104.4	205	13 7 98 190 157 128 158 76 37 89
147	98.1	205	22 123 175 179 79 20 185 164 39
146	73.8	205	36 97 34 63 122 183 88 199 65
145	88.9	205	99 109 64 4 182 31 47 118 62
139	79.8	205	102 9 103 169 46 91 105 142 176
146	88.2	210	151 171 86 48 77 172 200 192 42 198
148	75.3	210	130 71 59 137 5 108 51 119 173 96
108	62.7	210	120 3 141 66 165 93 152
147	76.4	214	126 166 188 70 194 87 58 60 15 149
148	76.5	214	84 178 124 168 69 111 81 184 19 1
146	78.6	214	14 189 107 133 121 156 167 72 33 18
147	83.5	214	26 55 148 40 11 150 38 154 145 68
147	110.5	214	85 136 25 10 57 17 106 115 52
149	51.0	214	78 153 100 30 61 45 92 82 101

Table 28 New solution for instance P121212 with cost 2209.3

q	*cost*	*f*	*customers*
995	100	204	
1,415	100	206	
551	100	209	
144	88.0	204	45 92 17 160 47 147 126 136 87 159
125	59.9	204	95 117 9 185 99 194 161 198
147	139.8	204	30 143 80 119 49 184 63 171 1 39
147	88.1	204	23 22 67 200 90 178 120 57 64 191
141	100.6	204	12 182 24 54 174 28 170 141 122 199
143	88.9	204	105 111 128 175 18 177 13 96 44 173
148	84.8	204	34 72 62 7 61 156 100 157 145 74
146	59.2	206	75 163 77 29 35 150 46 195 19 102 88
148	98.8	206	60 110 48 139 187 176 116 121 53 56 158
148	70.3	206	98 155 25 167 59 68 135 26 33
149	83.1	206	188 41 149 93 91 11 109 113 154 21

Table 28 continued

q	*cost*	*f*	*customers*
91	44.9	206	69 15 144 114 151 40
146	93.4	206	169 2 82 168 186 123 162 5 43 32
146	89.2	206	73 134 165 10 38 37 115 192 103
145	86.9	206	142 148 58 124 16 36 132 180
146	161.0	206	140 125 14 4 153 70 71 127 106 107
150	113.7	206	65 108 129 183 94 89 189 193 172 51
132	57.6	209	3 196 66 197 181 146 6 84 76
124	66.1	209	42 52 133 138 85 97 55 20 104
145	109.0	209	164 131 78 27 118 86 137 190 130
150	126.0	209	81 79 152 166 179 83 101 50 31 8 112

Table 29 New solution for instance P121222 with cost 2222.9

q	*cost*	*f*	*customers*
724	100	201	
969	100	203	
588	100	205	
706	100	208	
149	96.7	201	59 117 4 95 197 36 142 180 104
142	87.0	201	56 16 13 92 45 85 133 113
146	111.4	201	25 177 86 49 128 87 70 136 83
145	83.8	201	123 176 81 46 125 80 73 12 174
142	47.1	201	44 58 114 38 102 189 17 43 22
111	53.9	203	47 137 61 124 172 39 3 98
148	106.0	203	50 132 191 165 33 159 26 187 127
135	86.8	203	194 20 110 82 121 57 9 195 34 119
145	63.6	203	152 51 144 84 27 93 116 100 41
147	86.0	203	131 198 175 66 65 164 68 29 67 23
147	107.2	203	74 192 182 148 101 155 42 53 96 72 167
136	98.7	203	10 186 97 145 107 108 138 163 139 11
142	96.3	205	5 153 140 126 14 120 106 89 147 171
146	70.4	205	122 79 94 77 2 178 30 150 6 146
150	102.6	205	154 156 199 143 88 60 105 103 115 169
150	74.8	205	18 19 78 21 173 135 99 32 151 185
143	71.2	208	200 52 196 168 71 7 157 193 109
149	99.9	208	48 111 130 161 8 55 40 28 184 37 112
137	89.8	208	15 24 129 62 162 188 118 90 31 54 1
141	117.1	208	166 160 35 75 149 63 190 69 170
136	72.5	208	76 134 183 181 158 64 141 91 179

Table 30 New solution for instance P123112 with cost 1954.7

q	*cost*	*f*	*customers*
1,184	100	202	
529	100	207	
577	100	209	
736	100	210	
149	49.2	202	42 181 198 174 183 200 180 195 179 186
147	55.2	202	172 188 178 185 191 192 184 189 173 171
146	70.7	202	193 196 194 170 197 169 182 187 199 190 175
150	93.2	202	176 177 119 125 111 131 128 135 132 124 27
148	100.1	202	7 126 123 121 109 31 3 156 157 149
149	64.5	202	161 145 140 151 158 146 154 139 19
145	73.3	202	164 159 142 166 155 144 167 143 163
150	72.3	202	147 148 150 141 153 165 162 152 160 168 20
111	29.9	207	99 86 95 35 88 98 83 84
149	45.6	207	14 107 100 90 105 92 79 97 96 89
120	31.0	207	87 104 91 81 80 101 94 93 108
149	87.5	207	30 37 32 77 82 85 103 106 78 102
146	58.0	209	45 73 55 57 54 50 69 63 46 62
147	49.3	209	61 68 71 64 74 59 60 66 65
144	60.4	209	51 48 52 49 75 58 76 44 18
140	122.0	209	22 17 16 26 40 5 33 39
145	87.3	210	6 24 47 70 72 53 56 67 25
149	70.3	210	138 113 117 133 136 110 134 130 41
150	80.7	210	129 120 29 118 114 115 137 127 10
147	113.4	210	28 9 11 15 21 4 116 112 122 43
145	140.6	210	12 2 38 1 23 8 34 36 13

References

Baldacci, R., Mingozzi, A., Wolfler-Calvo, R.: An exact method for the capacitated location-routing problem. Oper. Res. **59**, 1284–1296 (2011)

Barreto, S.: Análise e Modelização de Problemas de localização-distribuição. Ph.D. thesis, University of Aveiro, Campus Universitário de Santiago, 3810–193 Aveiro, Portugal. In Portuguese (2004)

Belenguer, J.M., Benavent, E., Prins, C., Prodhon, C., Wolfler-Calvo, R.: A branch-and-cut algorithm for the capacitated location routing problem. Comput. Oper. Res. **38**, 931–941 (2011)

Clarke, G., Wright, J.W.: Scheduling of vehicles from a central depot to a number of delivery points. Oper. Res. **12**, 568–581 (1964)

Contardo, C., Cordeau, J.F., Gendron, B.: A computational comparison of flow formulations for the capacitated location-routing problem. Technical Report CIRRELT-2011-47, Université de Montréal, Canada (2011)

Contardo, C., Cordeau, J.F.: Gendron. B. An exact algorithm based on cut-and-column generation for the capacitated location-routing problem, INFORMS J. Comput. (2013). (Forthcoming)

Croes, G.A.: A method for solving traveling-salesman problems. Oper. Res. **6**, 791–812 (1958)

de Franceschi, R., Fischetti, M., Toth, P.: A new ILP-based refinement heuristic for vehicle routing problems. Math. Program. Ser. B **105**, 471–499 (2006)

Duhamel, C., Lacomme, P., Prins, C., Prodhon, C.: A GRASP×ELS approach for the capacitated location-routing problem. Comput. Oper. Res. **37**, 1912–1923 (2010)

Feo, T., Resende, M.: A probabilistic heuristic for a computationally difficult set covering problem. Oper. Res. Lett. **8**, 67–71 (1989)

Hemmelmayr, V.C., Cordeau, J.F., Crainic, T.G.: An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics. Comput. Oper. Res. **39**, 3185–3199 (2012)

Lin, S., Kernighan, B.W.: An effective heuristic algorithm for the traveling-salesman problem. Oper. Res. **21**, 498–516 (1973)

Pirkwieser, S., Raidl, G.R. Variable neighborhood search coupled with ILP-based very large-neighborhood searches for the (periodic) location-routing problem. In: Blesa, M., Blum, C., Raidl, G., Roli, A., Sampels, M. (eds.) Hybrid Metaheuristics, volume 6373 of Lecture Notes in Computer Science, pp. 174–189 (2010)

Potvin, J.Y., Rousseau, J.M.: An exchange heuristic for routeing problems with time windows. J. Oper. Res. Soc. **46**, 1433–1446 (1995)

Prins, C., Prodhon, C., Wolfler-Calvo, R.: Solving the capacitated location-routing problem by a GRASP complemented by a learning process and path relinking. 4OR **4**, 221–238 (2006)

Prins, C., Prodhon, C., Ruiz, A., Soriano, P., Wolfler-Calvo, R.: Solving the capacitated location-routing problem by a cooperative Lagrangean relaxation-granular tabu search heuristic. Transp. Sci. **41**, 470–483 (2007)

Prodhon, C.: An ELS x path relinking hybrid for the periodic location-routing problem. In: Blesa, M., Blum, C., Di Gaspero, L., Roli, A., Sampels, M., Schaerf, A. (eds.) Hybrid Metaheuristics, volume 5818 of Lecture Notes in Computer Science, pp. 15–29 (2009)

Prodhon, C.: A hybrid evolutionary algorithm for the periodic location-routing problem. Eur. J. Oper. Res. **210**, 204–212 (2011)

Prodhon, C., Prins, C.: A memetic algorithm with population management (MA|PM) for the periodic location-routing problem. In: Blesa, M., Blum, C., Cotta, C., Fernández, A., Gallardo, J., Roli, A., Sampels, M. (eds.) Hybrid Metaheuristics, volume 5296 of Lecture Notes in Computer Science, pp. 43–57 (2008)

Rochat, Y., Taillard, E.D.: Probabilistic diversification and intensification in local search for vehicle routing. J. Heuristics **1**, 147–167 (1995)

Røpke, S., Pisinger, D.: An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transp. Sci. **40**, 455–472 (2006)

Salhi, S., Sari, M., Saidi, D., Touati, N.: Adaptation of some vehicle fleet mix heuristics. Omega **20**, 653–660 (1992)

Savelsbergh, M.W.P.: The vehicle routing problem with time windows: minimizing route duration. ORSA J. Comput. **4**, 146–154 (1992)

Subramanian, A., Uchoa, E., Ochi, L.S.: A hybrid algorithm for a class of vehicle routing problems. Comput. Oper. Res. **40**, 2519–2531 (2013)

Tuzun, D., Burke, L.I.: A two-phase tabu search approach to the location routing problem. Eur. J. Oper. Res. **116**, 87–99 (1999)

Yu, V.F., Lin, S.W., Lee, W., Ting, C.J.: A simulated annealing heuristic for the capacitated location routing problem. Comput. Ind. Eng. **58**, 288–299 (2010)