CIRRELT

Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

A GRASP + ILP-Based Metaheuristic for the Capacitated Location-Routing Problem

Claudio Contardo
Jean-François Cordeau
Bernard Gendron

September 2011

CIRRELT-2011-52

Bureaux de Montréal :
Université de Montréal
C.P. 6128, succ. Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
2325, de la Terrasse, bureau 2642
Québec (Québec)
Canada G1V 0A6
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

Université Laval · UQÀM · HEC Montréal · École Polytechnique Montréal · Université de Montréal

A GRASP + ILP-Based Metaheuristic for the Capacitated Location-Routing Problem

Claudio Contardo[1,2,*], Jean-François Cordeau[1,3], Bernard Gendron[1,2]

[1] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT)

[2] Department of Computer Science and Operations Research, Université de Montréal, P.O. Box 6128, Station Centre-Ville, Montréal, Canada H3C 3J7

[3] Department of Logistics and Operations Management, HEC Montréal, 3000 Côte-Sainte-Catherine, Montréal, Canada H3T 2A7

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 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.

Acknowledgements. The authors would like to thank the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Fonds québécois de la recherche sur la nature et les technologies (FQRNT) for their financial support.

Results and views expressed in this publication are the sole responsibility of the authors and do not necessarily reflect those of CIRRELT.

Les résultats et opinions contenus dans cette publication ne reflètent pas nécessairement la position du CIRRELT et n'engagent pas sa responsabilité.

---

* Corresponding author: Claudio.Contardo@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2011

© Copyright Contardo, Cordeau, Gendron and CIRRELT, 2011

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 two 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 [3, 5] and column generation [1, 6]. 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. [15], Prodhon and Prins [18], Prodhon [16, 17] and Duhamel et al. [9] 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 [12] 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. [10] 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. [21] proposed a simulated annealing heuristic for the problem, in which CLRP solutions are coded as genes and then modified using mutation and crossover operators.

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. [14] and Duhamel et al. [9] 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 de Franceschi et al. [8], 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.

iii. to introduce a new technique based on the solution of an ILP, for combining a bundle of

reasonably good solutions with the objective of eventually producing another solution of better quality.

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 [3, 5]. 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 om 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 Section 2 we give a general description of our solution approach. In Section 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 Section 4 we introduce the location-reallocation model. We strengthen it with valid inequalities and describe the pricing algorithm used to derive columns of negative reduced cost. In Section 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 Section 6 and by conclusions in Section 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, namely a GRASP, local search (LS), a solution blender (SB) and a local improvement heuristic (LIH).

2.1 GRASP

A GRASP is a metaheuristic based on the randomization of a greedy criterion. In this paper, we propose a GRASP based on a variation of the extended Clarke and Wright savings algorithm (ECWSA) introduced by Prins et al. [14].

2.2 Local search

Local search procedures are greedy algorithms applied to a feasible solution to further improve its quality. Here, we use seven different methods that are applied iteratively until no further improvements are found.

2.3 Solution blender

The solution blender (SB) is a method based on the solution of an integer-linear program, called the location-reallocation model (LRM). 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 the set of routing variables is restricted to contain a fixed number of columns defined in advance, and therefore no column generation is applied. We complement this with the use of local branching constraints used to fix a large number of variables.

2.4 Local improvement heuristic

The local improvement heuristic (LIH) is a *destroy-and-repair* method inspired from the ALNS metaheuristic. 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 $LIH(\Gamma)$.

2.5 The complete algorithm

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.

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 $\mathcal{T}$.
7: **if** it found a solution $\mathcal{T}' \notin \mathcal{P}$ **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** 6.
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** 6.
16: **end if**
17: Increase Γ by some positive value.
18: **until** some stopping criterion is met

3 Pure metaheuristics

In this section we describe two metaheuristic procedures used in our algorithm, namely a GRASP and a local search (LS) method. We refer to these as pure metaheuristics to distinguish them from the ILP-based metaheuristics that will be introduced later.

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 many 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* $x \in \mathcal{X}$ *with smallest value of* $f(x)$*. Pick* x' *randomly in* $\mathcal{L}$*"*. For the CLRP, Prins et al. [14] 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. 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. 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 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 [4] and to Prins et al. [14]. 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, A denotes the set of already assigned customers, $\gamma(\cdot)$ represent the facilities to which customers are assigned (a customer $j \notin A$ is such that $\gamma(j) = -1$), and $l(\cdot)$ represents the current loads of facilities.

Algorithm 2 ECWSA

$F \leftarrow \emptyset, A \leftarrow \emptyset, \gamma(j) \leftarrow -1$ 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**
  $j' \leftarrow \arg\min\{\sum_{i \in F} c_{ij} : j \notin A\}$.
  $i' \leftarrow \arg\min\{2c_{ij'} + f_i \delta_{i \notin F} : i \in I, l(i) + d_{j'} \leq b_i\}$.
  $F \leftarrow F \cup \{i'\}, A \leftarrow A \cup \{j'\}, \gamma(j') \leftarrow i', l(i') \leftarrow l(i') + d_{j'}$.
end while
$\mathcal{R} \leftarrow \{\{\gamma(j), j\} : j \in J\}$.
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, A, γ and l accordingly.
  end if
until $s \leq 0$

In our GRASP, we replace the three optimization problems appearing in the pseudo-code with some randomized variants. The deterministic statement $j' \leftarrow \arg\min\{\sum_{i \in F} c_{ij} : j \notin A\}$ is changed to randomly picking a customer j' among the five customers not in A with minimum value of $\sum_{i \in F} c_{ij}$. 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 (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 $i' \in I$, 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 possibe 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 local search (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 solution blender heuristic which will be described in Section 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 local search 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 [19]. 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 Swap two customers from different routes [7]. This procedure is performed using a best-improvement criterion.

2-OPT* Two routes are split and re-merged [13]. This procedure is performed using a best-improvement criterion.

3-OPT Pick three customers in different routes and evaluate all possible swaps between them [11]. 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.

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 de Franceschi et al. [8], 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 solution blender and the local improvement heuristics. 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. short-cutting the remaining consecutive nodes in the routes,

iii. deleting the edges linking facilities to customers,

iv. 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. Figure 1 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 $e \in \cup_{r\in\mathcal{R}} E(r)$ we associate an insertion point $p = (i,e)$, at which customers in T can be reinserted. Let us denote, for a given facility i, $\mathcal{I}_i(\mathcal{R}) = \{p = (i,e) : e \in E(r) \text{ for some } r \in \mathcal{R}\}$. Also, for each $r \in \mathcal{R}$ and for each $i \in I$, $p = (i, \{i, h(i,r)\})$ represents an insertion point from which a subtour can be connected to facility i. For a given $i \in I$, we denote $\mathcal{I}(h(i,\mathcal{R})) = \{p = (i,e) : r \in \mathcal{R}, e = \{i, h(i,r)\}\}$. Analogously, $p = (i, \{i, t(i,r)\})$ represents the other insertion point from which the subtour is linked to facility i and we denote the set of insertion points as $\mathcal{I}(t(i,\mathcal{R}))$. Finally, the insertion point $p = (i, \{i, i\})$ is used for routes starting and ending at facility i and serving only customers in T. For every facility $i \in I$ the set of insertion points associated with i is defined as

$$\mathcal{I}_i = \mathcal{I}_i(\mathcal{R}) \cup \mathcal{I}(h(i,\mathcal{R})) \cup \mathcal{I}(t(i,\mathcal{R})) \cup \{(i, \{i,i\})\}. \tag{1}$$

For every insertion point $p = (i,e) \in \mathcal{I}_i$ we define $i(p) = i$, $e(p) = e$. Also, note that unless $p = (i, \{i,i\})$, $e(p)$ must contain at least one node in a subtour r, and if both nodes belong to a subtour then it must be the same. Therefore, one can define $r(p)$ equal to r in that case, and equal to -1 in the case $p = (i, \{i,i\})$. For every insertion point p, we denote by $\mathcal{S}_p$ the set of sequences or partial paths that can be inserted in p. Note that all the sequences that result in a violation of the capacities can be safely removed from $\mathcal{S}_p$. For every $s \in \mathcal{S}_p$ we let $E(s)$ 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:

$$c(s) = \begin{cases} \sum_{e\in E(s)} c_e - c_{e(p)} & \text{if } p \in \mathcal{I}_i(\mathcal{R}), s \in \mathcal{S}_p \\ \sum_{e\in E(s)} c_e & \text{otherwise.} \end{cases} \tag{2}$$

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 location-reallocation model is as follows:

(a) Complete solution. Set T surrounded by dotted circles

(b) Incomplete solution after the removal of nodes in T

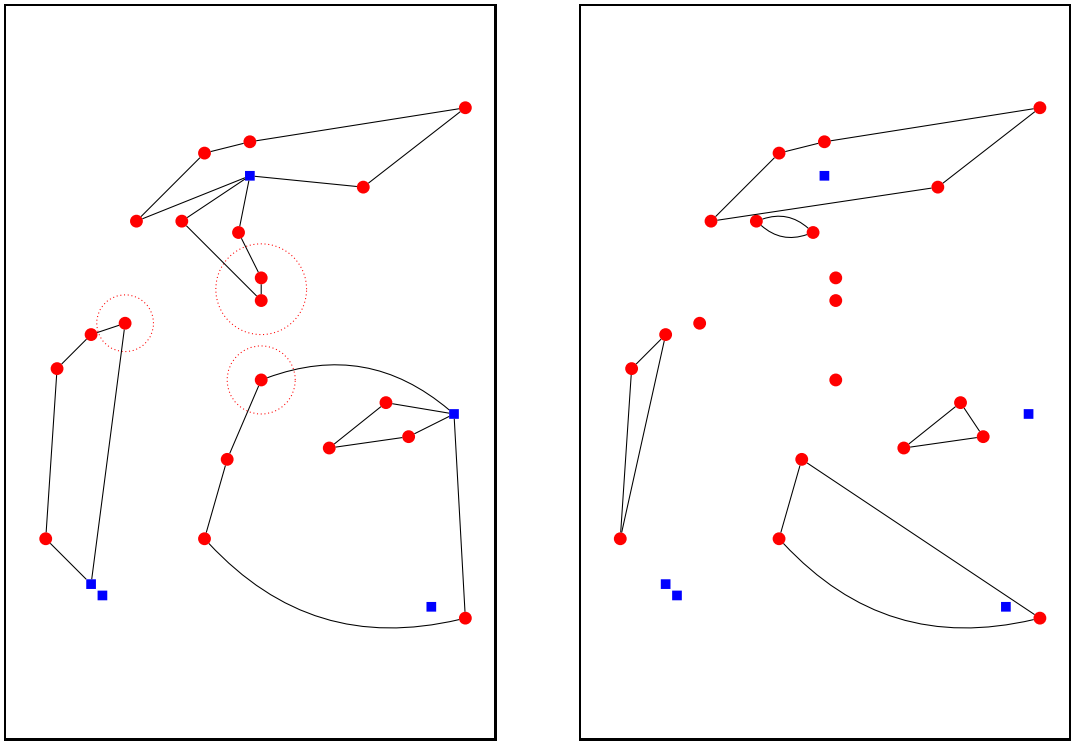

Figure 1: Example of node removal from a CLRP solution

$$\sum_{r\in\mathcal{R}} c(r) \quad + \quad \min \quad \sum_{i\in I} f_i z_i - \sum_{i\in I, r\in\mathcal{R}} c_{h(i,r)t(i,r)} u^{\mathcal{R}}_{ir} + 2\sum_{e\in\delta(I)} c_e y_e + \sum_{s\in\mathcal{S}} c(s) w_s \tag{3}$$

subject to

$$\sum_{i\in I} u^T_{ij} = 1 \qquad j \in T \tag{4}$$

$$\sum_{i\in I} u^{\mathcal{R}}_{ir} = 1 \qquad r \in \mathcal{R} \tag{5}$$

$$y_{ij} + \sum_{p\in\mathcal{I}_i} \sum_{s\in\mathcal{S}_p, j\in V(s)} w_s = u^T_{ij} \qquad i \in I, j \in T \tag{6}$$

$$\sum_{s\in\mathcal{S}_{(i,\{i,h(i,r)\})}} w_s = u^{\mathcal{R}}_{ir} \qquad i \in I, r \in \mathcal{R} \tag{7}$$

$$\sum_{s\in\mathcal{S}_{(i,\{i,h(i,r)\})}} w_s - \sum_{s\in\mathcal{S}_{(i,\{i,t(i,r)\})}} w_s = 0 \qquad i \in I, r \in \mathcal{R} \tag{8}$$

$$\sum_{s\in\mathcal{S}_p} w_s \le u^{\mathcal{R}}_{ir} \qquad i \in I, p \in \mathcal{I}_i(\mathcal{R}) \tag{9}$$

$$\sum_{i\in I} \sum_{p\in\mathcal{I}_i, r(p)=r} \sum_{s\in\mathcal{S}_p} q(s) w_s \le Q - q(r) \qquad r \in \mathcal{R} \tag{10}$$

$$\sum_{j\in T} d_j u^T_{ij} + \sum_{r\in\mathcal{R}} q(r) u^{\mathcal{R}}_{ir} \le b_i z_i \qquad i \in I \tag{11}$$

$$z, y, u, w \text{ binary.} \tag{12}$$

The objective function (3) contains two parts: a constant term given by the first expression, which takes into account the cost of the remaining part of the solution after the removal of the nodes in set T; and a linear term, combining setup costs with routing costs. Constraints (4)-(5) are the assignment constraints of customers to facilities. Constraints (6) are the degree constraints which ensure that customers in T will be reinserted. Constraints (7)-(8) ensure that partial routes $r \in \mathcal{R}$ will be linked to a facility. Constraints (9) ensure that for every insertion point $p \in \mathcal{I}_i(\mathcal{R})$ 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 (10) are the vehicle capacity inequalities. They make sure that the final routes will not exceed vehicle capacities. Constraints (11) 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 $p \in \mathcal{I}_i(\mathcal{R})$, 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 $p = (i, \{i, i\})$ then the minimum size is 2. Finally, if $p \in \mathcal{I}(h(i, \mathcal{R})) \cup \mathcal{I}(t(i, \mathcal{R}))$ for some i, 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 [6] 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:

$$x_e^i = \begin{cases} u_{ir}^{\mathcal{R}} - \sum_{s \in \mathcal{S}_{(i,e)}} w_s & \text{if } e \in E(r) \setminus \{\{h(i,r), t(i,r)\}\} \text{ for some } r \in \mathcal{R} \\ 1 - u_{ir}^{\mathcal{R}} & \text{if } e = \{h(i,r), t(i,r)\} \text{ for some } i \in I, r \in \mathcal{R} \\ \sum_{p \in \mathcal{I}_i} \sum_{s \in \mathcal{S}_p, e \in E(s)} w_s & \text{otherwise.} \end{cases} \tag{13}$$

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

$$u_{ij} = \begin{cases} u_{ir}^{\mathcal{R}} & \text{if } j \in V(r), r \in \mathcal{R} \\ u_{ij}^{T} & \text{if } j \in T. \end{cases} \tag{14}$$

Finally, for every facility $i \in I$ and $j \in J \setminus T$ we set $y_{ij} = 0$.

It suffices to use identities (13)-(14) to include the valid inequalities from the three-index vehicle-flow formulation. In particular, it is 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. [3] and Contardo et al. [5]. 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. [6] for solving the CLRP by branch-and-cut-and-price.

4.3 Column generation

The reduced cost of a column w_s will be computed differently depending on the position of 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 (6)-(10). The reduced cost associated to a column s with an insertion point $p \in \mathcal{I}_i$ will be given by

$$\overline{c}(s) = \begin{cases} c(s) - \sum_{j \in T(s)} \alpha_j - \sum_{j \in T(s)} d_j \theta_{r(p)} - \gamma_p & \text{if } p \in \mathcal{I}_i(\mathcal{R}) \\ c(s) - \sum_{j \in T(s)} \alpha_j - \beta_{ir(p)} - \sigma_{ir(p)} & \text{if } p \in \mathcal{I}(h(i,\mathcal{R})) \\ c(s) - \sum_{j \in T(s)} \alpha_j + \sigma_{ir(p)} & \text{if } p \in \mathcal{I}(t(i,\mathcal{R})) \\ c(s) - \sum_{j \in T(s)} \alpha_j & \text{if } p = (i, \{i,i\}). \end{cases} \tag{15}$$

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 (15) (modified by the dual information associated to valid inequalities) but they work along the exact same principle. The complete 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 de Franceschi et al. [8]. 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 a solution blender heuristic (SB), a method based on the existence of a pool of reasonably good solutions. We then describe a local improvement heuristic (LIH) based on the iterative solution of the LRM and solved by column and cut generation.

5.1 Solution blender

We present a heuristic procedure based on the solution of a particular case of the LRM. We refer to this method as the solution blender (SB). 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 solution blender is the solution pool $\mathcal{P}$ containing the 100 best solutions found by the GRASP combined with local search. Every new solution found is also subject to local search. Note also that the blending procedure is a generalization of the procedure introduced by Prins et al. [15] in which the size of the pool is fixed to one. Moreover, in that case this method also coincides with the solution of the CFLP.

5.1.1 Local branching

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 that are smaller than or equal to 0.1, we pick at most two variables z_{i_1}, z_{i_2} with the smallest reduced costs. For these variables we impose the following constraint:

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

The remaining location variables satisfying $z_i \leq 0.1$ are all fixed to zero. In particular, note that this method gives preference to the variables taking strictly positive values at the root relaxation, over the variables that are at their lower bound 0. In the case where three or more location variables take positive values (all of which having the same reduced cost equal to zero), we give preference to the ones taking the largest values.

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 local improvement heuristic. Note that every newly found solution is subject to local search. 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 de Franceschi et al. [8] and Pirkwieser and Raidl [12]. 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 local search 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. Each time, we save into 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 success, we use the RANDOM rule five more times.

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 at most five 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 solution blender.

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 12.2. 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. [3] and contains 30 instances with capacitated vehicles and facilities. The second set of instances ($\mathcal{F}_2$) has been introduced by Tuzun and Burke [20] 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 [2] 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. [1] 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. In Tables 1-4 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 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 10. Finally, gap_{best} is the relative gap (in %) of the best solution found, computed as $100 \times (z^*_{best} - z^*_{BKS})/z^*_{BKS}$. As the results show, our solutions are 0.22% above the best known solution on average for the instances of set $\mathcal{F}_1$, 0.59% for the instances of set $\mathcal{F}_2$, 0.61% for the instances of set $\mathcal{F}_3$ and 0.34% for the instances of set $\mathcal{F}_4$. Moreover, we are able to improve these values in 11 out of the 89 instances considered in our study. Regarding the CPU times, they lie around 45 minutes on average, and usually stay below 3 hours.

In Tables 5-8 we report the evolution of our algorithm during the different stages. In these tables, instances are grouped according to their size. The headers *GRASP*, *SB*, *LIH 1, 2, 3* stand for the different parts of our algorithm, including the three major iterations of the local improvement heuristic. 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 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.

In Table 9 we compare our algorithm against several of the most recent heuristics developed for the CLRP. The algorithms considered are: GRASP+PR [14], MA|PM [18], LRGTS [15], GRASP+ELS [9], VNS+ILP [12], SALRP [21] and ALNS [10]. 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 correspond 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. [14] 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 [9] since they only report best results after 5 runs, therefore any comparison to their method would be biased.

Finally, in Table 10 we report 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. [14] for most of the instances considered in this study. We have introduced the location-reallocation model that generalizes the CFLP and the RM of de Franceschi et al. [8] by simultaneously determining the locations of facilities as well as the reallocation of customers and routes to those facilities. We have introduced a new heuristic 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 based on the iterative solution of the LRM solved by column and cut generation. This local improvement heuristic 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 capacitated location routing problem (2E-CLRP), an important problem arising in the operation of city-logistics systems.

Instance	z^*_{BKS}	z^*_{avg}	$stdev$	gap_{avg}	T_{avg}	z^*_{best}	gap_{best}
ppw-20x5-1a	54793	54793	0.00	0.00	1.28	54793	0.00
ppw-20x5-1b	39104	39104	0.00	0.00	2.25	39104	0.00
ppw-20x5-2a	48908	48908	0.00	0.00	1.21	48908	0.00
ppw-20x5-2b	37542	37542	0.00	0.00	2.28	37542	0.00
ppw-50x5-1a	90111	90111	0.00	0.00	12.18	90111	0.00
ppw-50x5-1b	63242	63248	0.03	0.01	17.40	63242	0.00
ppw-50x5-2a	88298	88332	0.12	0.04	14.77	88298	0.00
ppw-50x5-2b	67308	67554	0.34	0.37	18.53	67373	0.10
ppw-50x5-2bis	84055	84055	0.00	0.00	17.72	84055	0.00
ppw-50x5-2bbis	51822	51898	0.02	0.15	24.06	51883	0.12
ppw-50x5-3a	86203	86203	0.00	0.00	14.76	86203	0.00
ppw-50x5-3b	61830	61836	0.03	0.01	20.16	61830	0.00
ppw-100x5-1a	274814	275626	0.06	0.30	188.51	275406	0.22
ppw-100x5-1b	213615	214699	0.12	0.51	178.81	214308	0.32
ppw-100x5-2a	193671	194118	0.17	0.23	106.96	193769	0.05
ppw-100x5-2b	157095	157238	0.05	0.09	94.29	157157	0.04
ppw-100x5-3a	200079	200341	0.02	0.13	86.76	200277	0.10
ppw-100x5-3b	152441	152737	0.26	0.19	95.87	152441	0.00
ppw-100x10-1a	287983	293117	2.79	1.78	1840.90	288415	0.15
ppw-100x10-1b	231763	233416	0.79	0.71	2329.90	**230989**	-0.33
ppw-100x10-2a	243590	244022	0.11	0.18	211.45	243695	0.04
ppw-100x10-2b	203988	204200	0.17	0.10	242.75	203988	0.00
ppw-100x10-3a	250882	252371	0.36	0.59	2576.34	250882	0.00
ppw-100x10-3b	204317	204996	0.14	0.33	1005.74	204602	0.14
ppw-200x10-1a	477248	476674	0.12	-0.12	3785.47	**475344**	-0.40
ppw-200x10-1b	378065	378781	0.27	0.19	3646.74	**377043**	-0.27
ppw-200x10-2a	449571	449469	0.05	-0.02	5215.70	**449152**	-0.09
ppw-200x10-2b	374330	375053	0.13	0.19	2831.53	374469	0.04
ppw-200x10-3a	469433	471218	0.13	0.38	4356.16	469706	0.06
ppw-200x10-3b	362817	363755	0.18	0.26	4936.13	**362743**	-0.02
Average			0.22	0.22	1129.22		0.01

Table 1: Results on instances of set $\mathcal{F}_1$

Instance	z^*_{BKS}	z^*_{avg}	$stdev$	gap_{avg}	T_{avg}	z^*_{best}	gap_{best}
P111112	1467.7	1475.4	0.24	0.52	171.94	1468.2	0.03
P111122	1449.2	1454.2	0.35	0.35	474.10	1449.2	0.00
P111212	1394.8	1405.0	0.36	0.73	161.80	1396.6	0.13
P111222	1432.3	1445.4	0.45	0.92	505.91	1432.9	0.04
P112112	1167.2	1178.3	0.15	0.95	225.19	1176.3	0.78
P112122	1102.2	1106.0	0.25	0.34	415.47	1102.8	0.05
P112212	791.7	796.9	0.50	0.67	196.60	791.9	0.03
P112222	728.3	728.4	0.03	0.02	370.49	728.3	0.00
P113112	1238.5	1241.9	0.21	0.28	224.21	1239.4	0.08
P113122	1245.3	1246.4	0.07	0.09	471.62	1245.5	0.02
P113212	902.3	902.5	0.06	0.02	177.30	902.3	0.00
P113222	1018.3	1019.6	0.11	0.13	496.25	1018.3	0.00
P131112	1866.8	1934.7	0.24	3.64	1073.37	1928.0	3.28
P131122	1823.5	1834.2	0.32	0.58	2020.47	**1823.2**	-0.02
P131212	1965.1	1978.2	0.34	0.66	781.64	1969.8	0.24
P131222	1796.5	1800.2	0.23	0.21	1646.47	**1792.8**	-0.20
P132112	1443.3	1452.5	0.26	0.64	757.08	1447.5	0.29
P132122	1434.6	1448.1	0.34	0.94	2863.10	1443.8	0.64
P132212	1204.4	1206.1	0.05	0.14	958.65	1204.9	0.04
P132222	931.0	932.3	0.07	0.15	2466.29	931.7	0.08
P133112	1694.2	1711.7	0.43	1.03	991.68	1700.3	0.36
P133122	1392.0	1401.7	0.07	0.70	2016.18	1400.1	0.58
P133212	1198.3	1200.5	0.12	0.19	895.12	**1198.2**	-0.01
P133222	1151.8	1159.0	0.08	0.62	2640.94	1157.7	0.51
P121112	2251.9	2258.8	0.27	0.30	2094.06	**2249.0**	-0.13
P121122	2159.9	2161.4	0.18	0.07	4911.06	**2153.8**	-0.28
P121212	2220.0	2223.9	0.35	0.17	2304.13	**2212.4**	-0.34
P121222	2230.9	2238.6	0.17	0.34	5175.85	2232.5	0.07
P122112	2073.7	2094.5	0.31	1.00	3520.46	2085.0	0.54
P122122	1692.2	1709.0	0.24	1.00	7177.74	1703.8	0.69
P122212	1453.2	1469.2	0.18	1.10	4162.82	1465.9	0.87
P122222	1082.7	1087.2	0.17	0.41	7194.32	1083.9	0.11
P123112	1960.3	1971.7	0.23	0.58	3060.73	1966.7	0.33
P123122	1918.9	1941.6	0.23	1.18	9341.61	1932.7	0.72
P123212	1762.0	1769.8	0.18	0.44	3813.81	1765.8	0.22
P123222	1391.7	1393.9	0.11	0.16	5422.38	1392.4	0.05
Average			0.22	0.59	2255.02		0.27

Table 2: Results on instances of set $\mathcal{F}_2$

Instance	z^*_{BKS}	z^*_{avg}	$stdev$	gap_{avg}	T_{avg}	z^*_{best}	gap_{best}
Perl-12x2	203.98	203.98	0.00	0.00	0.16	203.98	0.00
Gas-21x5	424.90	424.90	0.00	0.00	1.24	424.90	0.00
Gas-22x5	585.11	585.11	0.00	0.00	2.54	585.11	0.00
Min-27x5	3062.02	3062.02	0.00	0.00	2.68	3062.02	0.00
Gas-29x5	512.10	512.10	0.00	0.00	4.53	512.10	0.00
Gas-32x5	562.22	562.25	0.03	0.00	5.04	562.22	0.00
Gas-32x5b	504.33	504.33	0.00	0.00	6.05	504.33	0.00
Gas-36x5	460.37	460.37	0.00	0.00	6.91	460.37	0.00
Chr-50x5ba	565.62	575.60	3.14	1.76	14.13	570.03	0.78
Chr-50x5be	565.60	580.98	10.20	2.72	15.26	565.60	0.00
Perl-55x15	1112.06	1112.66	0.54	0.05	42.14	1112.32	0.02
Chr-75x10ba	844.40	848.34	2.71	0.47	74.14	844.40	0.00
Chr-75x10be	848.85	853.87	1.38	0.59	84.97	850.93	0.24
Chr-75x10bmw	802.08	809.78	3.16	0.96	86.13	803.10	0.13
Perl-85x7	1622.50	1626.01	1.26	0.22	65.78	1623.86	0.08
Das-88x8	355.78	356.12	0.44	0.09	164.40	355.78	0.00
Chr-100x10	833.43	851.00	6.47	2.11	350.88	841.68	0.99
Min-134x8	5709.00	5816.73	66.78	1.89	1188.96	5719.25	0.18
Das-150x10	43963.60	44321.33	83.83	0.81	1311.31	44179.00	0.49
Average			9.47	0.61	180.38		0.15

Table 3: Results on instances of set $\mathcal{F}_3$

Instance	z^*_{BKS}	z^*_{avg}	$stdev$	gap_{avg}	T_{avg}	z^*_{best}	gap_{best}
M-n150x14a	1352.93	1354.73	1.38	0.13	1089.83	1353.46	0.04
M-n150x14b	1212.46	1219.44	4.16	0.58	942.44	1215.14	0.22
M-n199x14a	1644.35	1645.97	1.74	0.10	3107.52	1644.35	0.00
M-n199x14b	1480.43	1488.37	2.53	0.54	3050.01	1483.55	0.21
Average			2.45	0.34	2047.45		0.12

Table 4: Results on 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}
ppw-20x5	0.03	0.42	0.00	0.53	0.00	0.68	0.00	1.00	0.00	1.75
ppw-50x5	0.70	7.76	0.14	8.58	0.12	10.34	0.08	12.93	0.07	17.45
ppw-100x5	1.52	67.56	0.38	81.93	0.35	88.86	0.30	100.19	0.24	125.20
ppw-100x10	3.51	78.92	2.33	376.89	1.01	815.67	0.81	1033.83	0.62	1367.85
ppw-200x10	1.57	1036.77	0.60	1704.85	0.25	2633.61	0.19	3133.39	0.15	4128.62
Average	1.51	238.78	0.70	435.09	0.35	710.48	0.28	857.06	0.22	1129.22

Table 5: 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}
100x10	2.24	93.71	0.72	100.64	0.64	118.31	0.56	146.15	0.53	192.84
100x20	1.82	147.23	0.40	165.46	0.36	220.71	0.34	297.23	0.31	455.64
150x10	3.11	490.65	1.24	543.03	1.13	626.82	1.10	716.21	1.05	909.59
150x20	2.71	678.91	0.67	771.96	0.60	1202.04	0.57	1533.36	0.53	2275.57
200x10	3.29	1591.95	1.02	1836.23	0.81	2409.46	0.71	2602.60	0.60	3159.34
200x20	3.63	2253.24	0.71	2540.08	0.59	4072.30	0.57	4732.70	0.53	6537.16
Average	2.80	875.95	0.79	992.90	0.69	1441.61	0.64	1671.38	0.59	2255.02

Table 6: 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}
≤ 50 custs	1.18	2.41	0.65	2.56	0.57	3.06	0.50	4.13	0.45	5.85
> 50 custs	2.93	91.68	1.21	97.85	1.01	144.40	0.88	221.50	0.80	374.30
Average	2.01	44.69	0.91	47.70	0.78	70.01	0.68	107.09	0.61	180.38

Table 7: 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}
150 custs	5.42	677.23	0.43	759.88	0.40	817.56	0.36	892.03	0.35	1016.14
199 custs	6.15	2078.87	0.44	2362.29	0.35	2530.13	0.33	2689.19	0.32	3078.77
Average	5.78	1378.05	0.44	1561.08	0.38	1673.85	0.35	1790.61	0.34	2047.45

Table 8: Algorithm evolution for instances of set $\mathcal{F}_4$

Method	$\mathcal{F}_1$		$\mathcal{F}_2$		$\mathcal{F}_3^{\S}$	
	gap	T	gap	T	gap	T
GRASP+PR[1]	3.60	96.5	3.42	159.56	1.49	21.15
MA|PM[1]	1.38	76.7	1.78	203.13	2.01	37.8
LRGTS[1]	0.74	17.5	1.76	21.24	1.64	18.21
GRASP+ELS[2]	1.07	65.2	1.22	606.6	0.02	187.7
VNS+ILP[3]	0.86	6.7	–	–	–	–
SALRP[1]	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	1.51	238.78	2.80	875.95	1.90	54.87
GRASP+SB	0.70	435.09	0.79	992.90	1.01	57.74
GRASP+ILP	0.22	1129.22	0.59	2255.02	0.64	254.98

§ Subset of instances considered by all authors reporting results for $\mathcal{F}_3$

[1] Results reported on a single run

[2] Best results after 5 runs

[3] Instances in $\mathcal{F}_2$ and $\mathcal{F}_3$ not tested

Table 9: Comparison against other heuristics

Instance	z^*_{BKS}	z^*_{NEW}	Instance	z^*_{BKS}	z^*_{NEW}
ppw-100x10-1a	287983	**287695**	P111222	1432.3	**1432.2**
ppw-100x10-1b	231763	**230989**	P113212	902.3	**902.2**
ppw-200x10-1a	477248	**475294**	P113222	1018.3	**1018.2**
ppw-200x10-1b	378065	**377043**	P131122	1823.5	**1823.2**
ppw-200x10-2a	449571	**449115**	P131222	1796.5	**1792.7**
ppw-200x10-2b	374330	**374280**	P133212	1198.3	**1198.2**
ppw-200x10-3b	362817	**362653**	P121112	2251.9	**2248.9**
			P121122	2159.9	**2153.8**
			P121212	2220.0	**2212.4**
			P121222	2230.9	**2222.9**

Table 10: New best known solutions

References

[1] R. Baldacci, A. Mingozzi, and R. Wolfler-Calvo. An exact method for the capacitated location-routing problem. *Operations Research*, 2011. Forthcoming.

[2] S. Barreto. *Análise e Modelização de Problemas de localização-distribuição*. PhD thesis, University of Aveiro, Campus Universitário de Santiago, 3810-193 Aveiro, Portugal. In Portuguese, 2004.

[3] J. M. Belenguer, E. Benavent, C. Prins, C. Prodhon, and R. Wolfler-Calvo. A branch-and-cut algorithm for the capacitated location routing problem. *Computers & Operations Research*, 38:931–941, 2010.

[4] G. Clarke and J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12:568–581, 1964.

[5] C. Contardo, J.-F. Cordeau, and B. Gendron. A computational comparison of flow formulations for the capacitated location-routing problem. Technical Report CIRRELT-2011-47, Université de Montréal, 2011.

[6] C. Contardo, J.-F. Cordeau, and B. Gendron. A branch-and-cut-and-price algorithm for the capacitated location-routing problem. Technical Report CIRRELT-2011-44, Université de Montréal, 2011.

[7] G. A. Croes. A method for solving traveling-salesman problems. *Operations Research*, 6:791–812, 1958.

[8] R. de Franceschi, M. Fischetti, and P. Toth. A new ILP-based refinement heuristic for vehicle routing problems. *Mathematical Programming*, 105:471–499, 2006.

[9] C. Duhamel, P. Lacomme, C. Prins, and C. Prodhon. A GRASP×ELS approach for the capacitated location-routing problem. *Computers & Operations Research*, 37:1912–1923, 2010.

[10] V. C. Hemmelmayr, J.-F. Cordeau, and T. G. Crainic. An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics. Technical Report CIRRELT-2011-42, Université de Montréal, 2011.

[11] S. Lin and B. W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21:498–516, 1973.

[12] S. Pirkwieser and G. R. Raidl. Variable neighborhood search coupled with ILP-based very large-neighborhood searches for the (periodic) location-routing problem. In *Hybrid Metaheuristics - Seventh International Workshop, HM 2010*, volume 6373 of *Lecture Notes in Computer Science*, pages 174–189, Vienna, 2010.

[13] J.-Y. Potvin and J.-M. Rousseau. An exchange heuristic for routeing problems with time windows. *The Journal of the Operational Research Society*, 46:1433–1446, 1995.

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

[15] C. Prins, C. Prodhon, A. Ruiz, P. Soriano, and R. Wolfer Calvo. Solving the capacitated location-routing problem by a cooperative Lagrangean relaxation-granular tabu search heuristic. *Transportation Science*, 41:470–483, 2007.

[16] C. Prodhon. An ELS x path relinking hybrid for the periodic location-routing problem. In M. Blesa, C. Blum, L. Di Gaspero, A. Roli, M. Sampels, and A. Schaerf, editors, *Hybrid Metaheuristics*, volume 5818 of *Lecture Notes in Computer Science*, pages 15–29. Springer Berlin / Heidelberg, 2009.

[17] C. Prodhon. A hybrid evolutionary algorithm for the periodic location-routing problem. *European Journal of Operational Research*, 210:204–212, 2011.

[18] C. Prodhon and C. Prins. A memetic algorithm with population management (MA|PM) for the periodic location-routing problem. In M. Blesa, C. Blum, C. Cotta, A. Fernández, J. Gallardo, A. Roli, and M. Sampels, editors, *Hybrid Metaheuristics*, volume 5296 of *Lecture Notes in Computer Science*, pages 43–57. Springer Berlin / Heidelberg, 2008.

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

[20] D. Tuzun and L. I. Burke. A two-phase tabu search approach to the location routing problem. *European Journal of Operational Research*, 116:87–99, 1999.

[21] V. F. Yu, S.-W. Lin, W. Lee, and C.-J. Ting. A simulated annealing heuristic for the capacitated location routing problem. *Computers & Industrial Engineering*, 58:288–299, 2010.