INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

WILEY

INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

Intl. Trans. in Op. Res. 00 (2014) 1–19
DOI: 10.1111/itor.12103

Matheuristics for the capacitated p-median problem

Fernando Stefanello[a], Olinto C. B. de Araújo[b] and Felipe M. Müller[c]

[a]*Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, RS, Brazil*
[b]*Colégio Técnico Industrial de Santa Maria, Universidade Federal de Santa Maria, Santa Maria, RS, Brazil*
[c]*Programa de Pós-Graduação em Informática, Universidade Federal de Santa Maria, Santa Maria, RS, Brazil*
E-mail: stefanello@inf.ufsm.br [Stefanello]; olinto@ctism.ufsm.br [de Araújo]; felipe@inf.ufsm.br [Müller]

Received 30 November 2012; received in revised form 5 May 2014; accepted 10 May 2014

Abstract

The improvement in the performance of computers and mathematical programming techniques has led to the development of a new class of algorithms called matheuristics. Associated with an improvement of Mixed Integer Programming (MIP) solvers, these methods have successfully solved plenty of combinatorial optimization problems. This paper presents a matheuristic approach that hybridizes local search based metaheuristics and mathematical programming techniques to solve the capacitated p-median problem. The proposal considers reduced mathematical models obtained by a heuristic elimination of variables that are unlikely to belong to a good or optimal solution. In addition, a partial optimization algorithm based on the reduction is proposed. All mathematical models are solved by an MIP solver. Computational experiments on five sets of instances confirm the good performance of our approach.

Keywords: capacitated p-median problem; matheuristic algorithm; size reduction

1. Introduction

Capacitated clustering problems have many practical applications, such as facility location, hospitals, schools, and garbage collection among others. In all these cases, given a set of nodes associated with the reality to be modeled, the set is partitioned into p disjoint clusters so that the total dissimilarity within each cluster is minimized. More specifically, in this paper we deal with a particular capacitated clustering location problem, the capacitated p-median problem (CPMP). The CPMP consists of allocating p facilities (medians) to serve n demand points (nodes). The objective is to minimize the sum of the distances between medians and nodes, with the constraint that the total demand of the nodes assigned to each median does not exceed its given capacity.

The CPMP is known to be NP-hard (Garey and Johnson, 1979). In scientific literature, there are several approaches using exact methods as well as heuristics to find good quality solutions within acceptable computational time.

The first work on the CPMP appeared in scientific literature in the 1980s (Mulvey and Beck, 1984; Pirkul, 1987). Osman and Christofides (1994) used a hybrid approach that combines simulated annealing and tabu search and randomly generated 20 instances with size ranging from 50 to 100 customers to test the proposed methods. Maniezzo et al. (1998) presented an evolutionary method and an effective local search technique to solve the CPMP. Computational results showed the effectiveness of the proposed approach on five sets of instances, including those proposed by Osman and Christofides. More recently, Baldacci et al. (2002) proposed a new method based on a set partitioning formulation. The authors presented computational results on instances from the literature and proposed new sets of test problems with additional constraints: bounds on the cluster cardinality and incompatibilities between entities. Lorena and Senne (2002, 2004) presented a column-generation method integrated to Lagrangean/surrogate relaxation to calculate lower bounds. Their proposed method identifies new productive columns, accelerating the computational process. Computational results were presented on instances generated based on a geographic database from the city São José dos Campos. Ahmadi and Osman (2005) proposed a combination of metaheuristics in a framework called GRAMPS (greedy random adaptive memory search method). A scatter search approach was proposed by Scheuerer and Wendolsky (2006), who evaluated it on instances from the literature, obtaining several new best solutions. Díaz and Fernández (2006) presented a hybrid scatter search and path relinking algorithm. The authors have run a series of computational experiments evaluating the proposed methods on instances from the literature, including instances corresponding to 737 cities in Spain. Both algorithms were evaluated separately; however, the combination of path relinking and scatter search gave the best results. Fleszar and Hindi (2008) solved the CPMP using variable neighborhood search to define sets of medians and the CPLEX package to solve assignment problems. Chaves et al. (2007) presented a hybrid heuristic called clustering search (CS), which consists in detecting promising search areas based on clustering. Boccia et al. (2007) proposed a cutting plane algorithm based on the Fenchel cuts. Computational results using the CPLEX showed that the approach is effective in solving hard instances or reducing their integrality gap.

The evolution of computers in the past years, characterized by a continuously increasing processing power and advances in parallelization techniques, has allowed to solve difficult computational problems in a shorter time when compared to computers a few years ago. Added to this scenario, we also observed an improvement on the performance of mixed integer programming solvers. Furthermore, Nievergelt (2000) claimed that complexity theory provides an overly pessimistic run time bounds for NP-hard problem, and the fact that many instances of this class of problems, including the practical ones, can be solved efficiently is often ignored. This scenario is propitious to the development of methods that hybridizes heuristic and mathematical programming techniques, the so-called "Matheuristics."

Talbi (2002), Dumitrescu and Stützle (2003), Puchinger and Raidl (2005), Fernandes and Lourenço (2006), and Jourdan et al. (2009) presented different taxonomies for such hybrid methods and a survey of the published studies which use these techniques. A general overview on combinations of metaheuristics with mathematical programming techniques is given in Maniezzo et al. (2010).

One of these techniques that have aroused interest in the scientific community is to use an exact method to solve the subproblems originated by fixing some variables of the original model. This idea is used in Mautor and Michelon (1997, 2001) for the quadratic assignment problem, where at each iteration of the method a set of variables is fixed to their current value. The remaining reduced

problem is optimized using an exact method based on branch and bound. A similar approach has been applied by Büdenbender et al. (2000) in a search procedure to the direct flight network design problem. A framework called POPMUSIC comprising these techniques is presented in Taillard and Voss (2002). Another similar technique is to eliminate some variables of the original model that are unlikely to appear in good solutions and solve the remaining reduced model. This technique is called "size reduction" and differs from the previous technique by the fact that it is not an iterative process. Fanjul-Peyro and Ruiz (2011) apply this method for the unrelated parallel machine scheduling problem, proposing a set of reductions and solving the reduced model by CPLEX.

This paper proposes a matheuristic called "Iterated Reduction Matheuristic Algorithm" (IRMA) to solve the CPMP in three phases. The first phase is an adaptation of the primal heuristic of Mulvey and Beck (1984) to obtain an initial solution. In the second phase, two simple and intuitive strategies are used to eliminate variables from the mathematical model, which is solved in order to improve the initial solution. In the last phase, a partial optimization algorithm based on reductions is used as a postoptimization method. Computational experiments were performed on a broad set of benchmark instances, including large size instances that were not considered before in the literature. The results showed that our approach outperformed the previously proposed methods achieving solutions in small computational time as well as providing good results.

The paper is organized as follows. Section 2 provides a formal description of the CPMP and Section 3 describes the proposed method. Extensive computational results for five different sets of instances are presented in Section 4. Conclusions and some future directions are presented in Section 5.

2. Mathematical formulation

The CPMP can be represented as a directed graph $G = (V, A)$ where $V = \{1, \ldots, n\}$ is the set of nodes or customers and A is the set of arcs, where the cost of each arc represents the distance between two nodes. Let $M = \{1, \ldots, m\}$ be the set of candidate medians or facilities. Each node $i \in V$ is associated with a demand q_i and each candidate median $j \in M$ with a capacity Q_j that must be respected. Without loss of generality, we can assume that $M = V$ and $n = m$. The goal is to find a subset of nodes M_p with $M_p \subseteq M$ and $|M_p| = p$ to be medians and to assign each node $v \in V$ to only one median $j \in M_p$ such that the sum of the distances d_{ij} between each node $i \in V$ and its median $j \in M_p$ is minimized.

Let $x_{ij} = 1$ if customer $i \in V$ is assigned to a median $j \in M$, and 0 otherwise. According to Lorena and Senne (2004), the CPMP can be formulated as follows:

$$z = \min \sum_{i \in V} \sum_{j \in M} d_{ij} x_{ij}, \tag{1}$$

Subject to

$$\sum_{j \in M} x_{jj} = p, \tag{2}$$

$$\sum_{j \in M} x_{ij} = 1 \quad \forall i \in V, \tag{3}$$

$$x_{ij} \leq x_{jj} \quad \forall i \in V, \quad \forall j \in M, \tag{4}$$

$$\sum_{i \in V} q_i x_{ij} \leq Q_j x_{jj} \quad \forall j \in M, \tag{5}$$

$$x_{ij} \in \{0, 1\} \quad \forall i \in V, \quad \forall j \in M. \tag{6}$$

Constraints (2) ensure the existence of exactly p clusters and constraints (3) require each node to be assigned to only one median. Disaggregated constraints (4) impose that each node must be assigned to a node that is selected as median. Constraints (5) limit the capacity of medians and constraints (6) define the domains of the variables.

3. Proposed method

Although nowadays the solution of combinatorial optimization problems by MIP solvers is an increasingly viable option, these solvers became impractical for large instances due to the number of variables involved. However, exploiting the structure of a problem, it is possible to observe that certain variables in the mathematical model are very unlikely to belong to good solutions. Such variables can be eliminated by a process referred in this paper as size-reduction heuristic. We call "reduced model" the mathematical model resulting from these strategies, and "full model" the model containing all variables.

Based on the idea of eliminating variables, we propose IRMA to solve the CPMP in three phases. A general schema of IRMA is shown in Fig. 1. In phase 1, we build an initial solution used in the next step. Phase 2 starts applying the size-reduction heuristics *R1* and *R2* described in Section 3.1 to obtain a reduced model. The size-reduction heuristic *R2* is applied only if $n > \lambda$, where by default $\lambda = 500$. We used this condition because, in general, the solver can deal and provide good results using only the size-reduction heuristics *R1* for small instances (say, 500 nodes or less). The reduced model in this phase is solved until an optimal solution is found or a preset time limit τ is reached. Phase 3 works as a postoptimization procedure, which starts from the solution of the previous phase and uses the partial optimization algorithm described in Section 3.2. This phase uses the

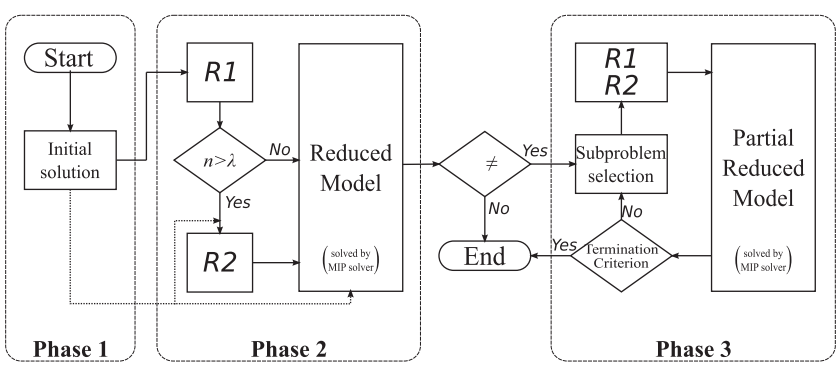

Fig. 1. IRMA schema.

size-reduction heuristics described in Section 3.1 and is applied only if the solver is not able to prove the optimality of the reduced model ($gap \neq 0$).

The initial solution is built based on the primal heuristic proposed in Mulvey and Beck (1984). In this heuristic, the first step is to select a random set of p starting medians. Nodes are assigned to the selected medians in a decreasing order of regret, where regret is defined as the absolute value of the difference in distance between the first and second nearest median. The median is recomputed after all assignments. This process is repeated until no changes in the median set occur or until an iteration limit is reached. The difference to our approach is that a node is chosen uniformly at random to be assigned to a median, and the only stop criterion is the iteration limit.

3.1. Size-reduction heuristics

In this subsection, we present two simple and intuitive size-reduction heuristics to eliminate variables of the mathematical model. The first size-reduction heuristic eliminates variables based on costs (distances) between nodes and candidate medians. In the second size-reduction heuristic, a node is considered as a candidate median only if it is near a median in the current solution.

3.1.1. Demand size-reduction (R1)

The first strategy considers the elimination of the variables x_{ij} if the candidate median $j \in M$ does not belong to the set of nearest nodes of i. Formally, we define the subset $K_i \subseteq V$ of the nearest nodes of i as $K_i = \{k \in V | \sum q_k \leq \alpha Q_i - q_i\}$, where k is nearer to i than k' if $d_{ik} < d_{ik'}$. Thus, for a candidate median $j \in M$, we eliminate the variable x_{ij} if $j \notin K_i$. Also, if there is a node i such that $q_i > Q_j - q_j$, then the variable x_{ij} is disposed. Finally, the parameter α is an expand capacity factor used to control the number of variables considered in the model.

This method provides a significant reduction in the number of variables and, although it does not guarantee optimality, it allows handling larger problems (say more than 4000 nodes).

Figure 2 illustrates the proposed reduction, indicating which nodes are likely to be associated with two candidate medians for the instance *sjc*1 proposed by Lorena and Senne (2004). In this figure, plots (a) and (b) show the representation of the remaining variables for two nodes and the optimal solution for this instance, respectively. We see that the reduction, in this case, preserves all variables belonging to the optimal solution.

An alternative reduction strategy is to eliminate variables x_{ij} if $i \notin K_j$ for a candidate median $j \in M$. This strategy is more intuitive and has the same number of eliminated variables as the first described method, but in the tested instances the obtained solution quality is slightly worse.

3.1.2. Neighborhood median size-reduction (R2)

This reduction starts from an initial solution and allows variables that define a node as median candidate only in regions adjacent to the current median. More specifically, given a cluster, only the β nodes nearest to the median are considered as candidate medians, where $\beta > 0$ is a integer parameter that defines the scope of the reduction method.

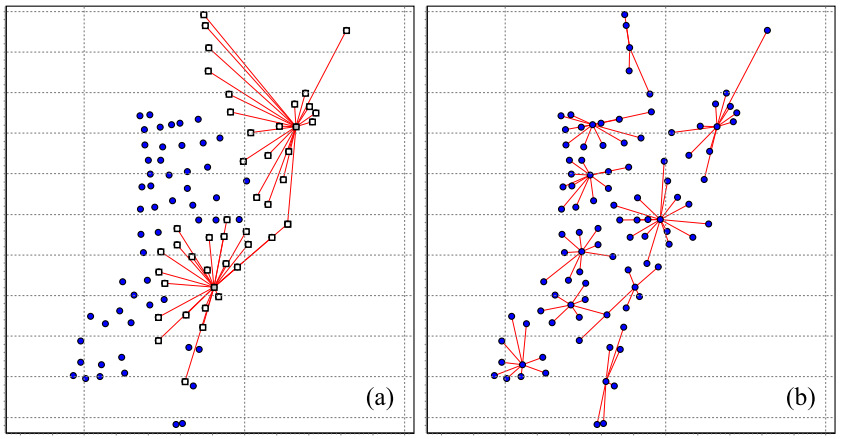

Fig. 2. Comparison between remaining variables and optimal solution. (a) Graphical representation of the remaining variables for two candidate medians; unfilled squares correspond to nodes that can be assigned to the respective candidate medians (connected by lines), and filled circles correspond to nodes that cannot be assigned to these two candidate medians. (b) Graphical representation of the optimal solution.

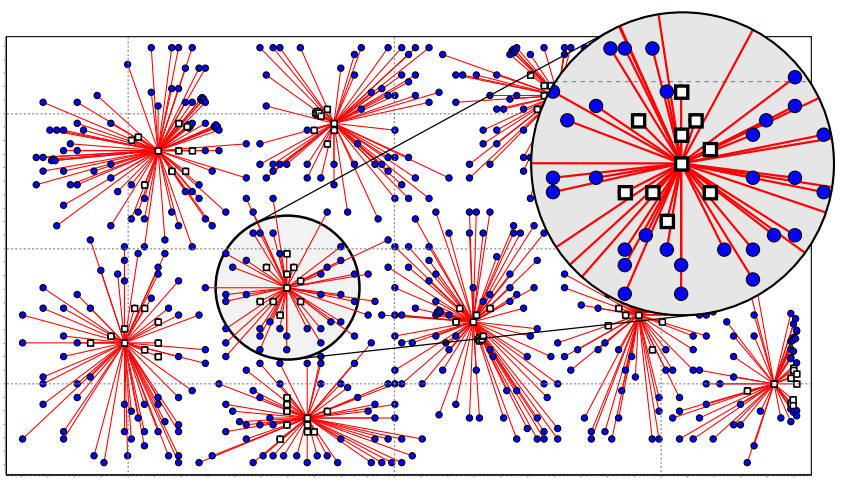

Fig. 3. Graphical representation of the size-reduction *R2*. Unfilled squares correspond to candidate medians.

The problem is restricted to the assignment problem for $\beta = 1$. When the number of nodes in a specific cluster is smaller than β, then all nodes in the cluster are considered candidate medians.

Figure 3 shows a representation of an initial solution for the instance $u724\_010$ described in Section 4. The unfilled squares correspond to candidate medians ($\beta = 10$) and the nodes depicted as filled circles cannot be medians.

3.2. Partial optimization algorithm

After some analyses of several good solutions, we observed that most clusters remain unchanged when there is an improvement in the objective function value. This fact suggests the possibility of iteratively reducing the size of the problem in small subregions and exploring each subregion in order to improve a solution. In the approach described in this subsection, we iteratively select subregions by freeing some variables in the mathematical model and solving it, until all subregions of the problem are explored.

Let $M' \subseteq M$ be the set of medians m in a solution $\mathcal{S}$, and the cluster $CL(m)$ be the set of nodes $v \in V$ associate to the median $m \in M'$ in the solution $\mathcal{S}$. Furthermore, let $\mathcal{D}\left(m', k\right)$ be the set of k closest clusters to the median m', that is, the k clusters with the shortest distance $d_{m',m},\ \forall m \in M'$.

Given an initial solution $\mathcal{S}$, Algorithm 1 describes the partial optimization algorithm used in the third phase of IRMA to solve the CPMP.

Algorithm 1. Partial optimization algorithm.

Input: $\mathcal{S}, n'$
Output: $\mathcal{S}$
1 Initialize parameters and mark all clusters as nonoptimized;
2 **while** *there is a nonoptimized cluster* **do**
3 Find the median m' of a nonoptimized cluster nearest to the node n' in order to define $\mathcal{D}\left(m', nc\right)$;
4 Optimize the model, fixing the variables **not** related to the nodes in $\mathcal{D}\left(m', nc\right)$;
5 **if** *there is an improvement* **then**
6 Mark a set of clusters $\mathcal{D}\left(m', \lfloor nc * h \rfloor\right)$ as nonoptimized;
7 **else**
8 Mark the clusters in $\mathcal{D}\left(m', nc\right)$ as optimized;
9 **endif**
10 Update nc;
11 **endw**
12 Return $\mathcal{S}$;

A cluster $CL(m)$ is considered as optimized when the solver cannot improve the solution in a subregion containing the cluster $CL(m)$. Thus, in line 1, the parameters are initialized and all clusters are set as nonoptimized. The loop in lines 2–11 is repeated while there are clusters to be optimized. In line 3, a reference median from the nonoptimized cluster, which is the nearest median to the node n', is selected and used for defining the set $\mathcal{D}\left(m', nc\right)$, where nc is the number of free clusters (as defined below). This set contains clusters whose nodes determine the free variables. A variable x_{ij} is defined as freed if i and j belong to $\mathcal{D}\left(m', nc\right)$ and thus is maintained in the model; otherwise, the variable is eliminated or fixed to its value in the solution $\mathcal{S}$. The number of free clusters nc should be large enough to explore a good subregion but, at the same time, small enough to allow the solver to optimize the subproblem within a reasonable time. We set this number to $nc = \max(\lfloor \frac{p}{n}\omega \rfloor, C_{lm})$, where ω is a parameter that indicates a suitable number of nodes that the solver can handle and

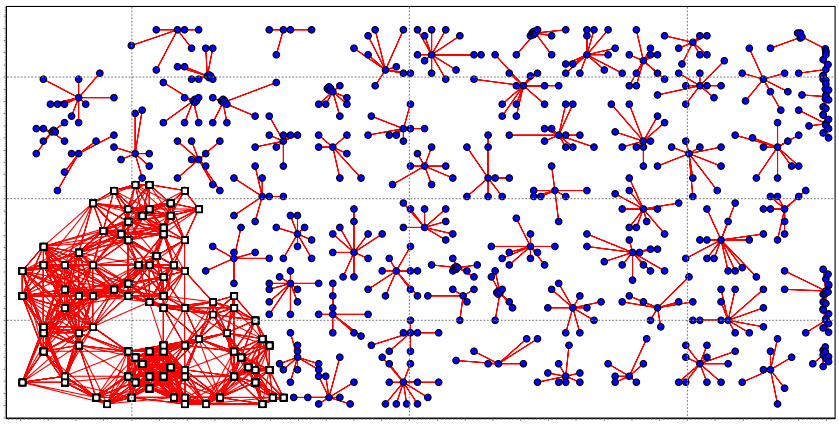

Fig. 4. Representation of free variables in the first iteration of the partial optimization algorithm. Unfilled squares correspond to nodes belonging to clusters to be optimized.

C_{lm} is the minimum number of clusters to be analyzed. The parameter ω is dynamically updated, as explained below, and C_{lm} is set to 5 by default. The reduced model is solved by an MIP solver in line 4. In line 5, it is checked if the solution was improved. In affirmative case, the subset of the $\lfloor nc * h \rfloor$ nearest clusters of m' is defined as nonoptimized in line 6 ($h = 1.8$ by default); otherwise, the clusters in $\mathcal{D}\left(m', nc\right)$ are defined as optimized (line 8). Finally, the parameter nc is updated and the algorithm returns the incumbent solution $\mathcal{S}$ (lines 10 and 12, respectively).

In order to ensure that the solution $\mathcal{S}$ is a local optimum in the considered neighborhood, Algorithm 1 can be repeated until there is no improvement. In this case, on each iteration, we choose the node n' alternately between the node with smallest and largest sum of coordinates x and y of all nodes $v \in V$.

Inspired by the local branching procedure proposed by Fischetti and Lodi (2003), who impose a time limit to control the execution time in a branch exploration, we use two features to control the time limit to explore subregions in line 4. First, we use a parameter τ that delimits the maximum time used by the solver. Second, we control and dynamically update the parameter ω that is responsible to define the size of the subregion to be explored. In the first iteration, we set an initial value ω_0 and for iteration i, $i \geq 1$, we empirically defined the formula $\omega_i = \omega_0 - (2\frac{t_{i-1}}{\tau} - 1)\omega_{i-1}0.1e^{-5(\frac{\omega-\omega_{i-1}}{\omega})^2}$, where t_{i-1} is the time to solve the previous iteration.

Figure 4 shows a representation of free variables in the first iteration of the partial optimization algorithm. In this figure, free variable are represented as lines connecting unfilled squares, while the fixed variables connect filled circles. Note that the size-reduction heuristic *R1* was applied to the clusters related to the free variables.

4. Computational results

In this section, we report the experiments to evaluate the performance of IRMA on large sets of instances, including comparisons with previous methods proposed in the scientific literature. Initially, we describe the dataset used in the experiments and then we detail the parameters and

specify the values used in the algorithm. The experiments were performed on four previously proposed sets of instances and a new large and diverse set, providing the most general experiments reported for this problem.

The first experiment was performed for two classic sets of instances reported in the literature, which were also used to calibrate some parameters. The next two experiments were performed on two sets of instances of the literature, including non-Euclidean distances and a set of large-scale instances. Finally, in the fourth experiment we considered a new and diverse set of instances, for which results are reported to provide a reference for future comparisons.

The computational results presented were obtained on an Intel i5-2300 2.80GHz CPU with 4 GB RAM running under Ubuntu 10.10 using CPLEX 12.3[1] with tree memory parameter set equal to 2000 and all other set to their default values.

The five sets of instances are detailed as follows. The classical set 1 was proposed by Osman and Christofides (1994) and the instances are named cpmp01 to cpmp20.[2] In this set, the first 10 instances have 50 nodes and 5 medians. The other 10 instances have 100 nodes and 10 medians. In this set, the distances were taken as the truncated Euclidean distance between corresponding pairs of coordinates.

Set 2 was proposed by Lorena and Senne (2004)[3] and comprises six real instances named $sjc1$ to $sjc4b$ that have been obtained from the city a geographic database of the city São José dos Campos, SP-Brazil.

We also used instances described by Díaz and Fernández (2006), which contain 737 nodes representing cities in Spain. In this set, the number of medians is defined to 74 and 148 and are used as two different vectors of demand, totaling four instances. This is considered set 3 and the number of nodes and medians are coded in the instance names. For example, we have "*spain*737_74_1," where 737 indicates the number of nodes representing cities in Spain; the number 74 indicates the number of medians and 1 indicates the vector of demands used. This set of instances uses a non-Euclidean distance matrix.

The fourth set of instances includes five large instances proposed by Lorena et al. (2003), named $p3038\_600$ to $p3038\_1000$, originated from one instance with 3038 nodes from TSP-LIB,[4] which was adapted to include 600, 700, 800, 900, and 1000 clusters.

A new set of instances, adapted from TSP-LIB, is also proposed in this paper (instance set 5). Most of the instances cannot be solved to optimality considering the current state-of-the-art MIP solvers, which justifies the use of heuristics. Moreover, the idea is to provide a more diverse set of data covering different characteristics. For each instance, a subset of five new instances with different numbers of medians was generated, so that different n/p ratios were included. A demand

[1] www-01.ibm.com/software/integration/optimization/cplex-optimizer/.

[2] Available at people.brunel.ac.uk/mastjjb/jeb/info.html.

[3] Available at www.lac.inpe.br/lorena/instancias.html.

[4] comopt.ifi.uni-heidelberg.de/software/TSPLIB95/.

Table 1
Summary of the parameters used in the three phases of IRMA

Parameters[a]	Phase 1	Phase 2	Phase 3
Iteration	1000	–	–
α	–	2	4
β	–	10	15
τ	–	$0.7n$	30
ω_0	–	–	300

[a]Iteration is the stop criterion in Phase 1; α and β are, respectively, the expanded factor of the reductions *R1* and *R2*; τ is a time limit to the solver and ω_0 is the number of free nodes.

using an integer random value chosen uniformly in the interval [1100] was added to each item. The same capacity is assigned to all medians and calculated by the expression:

$$Q_j = \left\lceil \sum_{i \in N} \frac{q_i}{p * r} \right\rceil, \quad j \in J,$$

where r belongs to the interval [0.8, 0.9] and n and p are shown in Table 7.

All sets of instances and their best-known solutions are referenced or available at www.inf.ufsm.br/stefanello/instances/.

Each set of instances has particular characteristics and a specific set of parameters allows to find better solutions. However, we provided a generic set of parameters for any set of instances. The parameters and their values in each phase are presented in Table 1. The most important parameter is the parameter α in phase 2 and we present experiments that help define this value in the next subsection. The remaining parameters are empirically defined in order to obtain near optimal solutions in an acceptable computational time.

As shown in the flowchart in Fig. 1, we used phase 2 without the reduction *R2* when $n < 500$, and phase 3 is applied only if the solver is not able to prove optimality of the reduced model. Also, by default, we used reduction *R2* when $n/p > 10$.

4.1. Results for instance sets 1 and 2

In the first experiment, we evaluated IRMA on the instance sets 1 and 2, and results to support the definition of the parameter α are shown in details. In this case, IRMA was configured only with phase 2 and reduction *R1*.

Table 2 shows the results for the instance sets 1 and 2 for different values of α on reduction *R1*. The value of the objective function of the optimal solution and the runtime to solve the full model by CPLEX are provided for comparison. The first three columns present the name of the instances, the number of nodes, and medians. The next columns show the values of objective function for the full model and IRMA for five different values of α (optimal solution values are given in bold type). Finally, we present the computational time. Note that in this set of instances, the solver proves the optimality of the reduced model using phase 2, and, as defined before, does not execute phase 3.

Table 2
Computational results of full model and IRMA for instance sets 1 and 2

			Objective function						Time (seconds)					
				IRMA—values of α						IRMA—values of α				
ID	n	p	Full model	**1.6**	**1.8**	**2.0**	**2.2**	**2.4**	Full model	**1.6**	**1.8**	**2.0**	**2.2**	**2.4**
cpmp01	50	5	**713**	opt	opt	opt	opt	opt	0.25	0.14	0.09	0.10	0.17	0.20
cpmp02	50	5	**740**	opt	opt	opt	opt	opt	0.10	0.06	0.04	0.06	0.05	0.05
cpmp03	50	5	**751**	759	759	752	opt	opt	0.23	0.12	0.14	0.11	0.12	0.16
cpmp04	50	5	**651**	opt	opt	opt	opt	opt	0.12	0.07	0.08	0.08	0.08	0.08
cpmp05	50	5	**664**	opt	opt	opt	opt	opt	0.32	0.08	0.08	0.13	0.09	0.15
cpmp06	50	5	**778**	opt	opt	opt	opt	opt	0.16	0.06	0.06	0.07	0.07	0.08
cpmp07	50	5	**787**	opt	opt	opt	opt	opt	0.45	0.31	0.32	0.27	0.28	0.35
cpmp08	50	5	**820**	opt	opt	opt	opt	opt	10.43	3.37	5.23	5.35	5.64	4.65
cpmp09	50	5	**715**	opt	opt	opt	opt	opt	0.41	0.14	0.16	0.26	0.17	0.24
cpmp10	50	5	**829**	832	832	832	832	opt	2.51	0.88	0.80	0.94	0.93	0.89
cpmp11	100	10	**1006**	opt	opt	opt	opt	opt	2.24	1.29	1.99	1.77	1.20	1.83
cpmp12	100	10	**966**	opt	opt	opt	opt	opt	2.25	0.88	1.10	1.35	1.01	1.23
cpmp13	100	10	**1026**	opt	opt	opt	opt	opt	0.98	0.31	0.22	0.35	0.39	0.43
cpmp14	100	10	**982**	opt	opt	opt	opt	opt	11.57	5.01	5.25	6.75	5.60	5.15
cpmp15	100	10	**1091**	opt	opt	opt	opt	opt	11.73	2.37	1.96	5.38	5.29	6.57
cpmp16	100	10	**954**	opt	opt	opt	opt	opt	2.55	0.70	0.71	0.77	0.79	0.80
cpmp17	100	10	**1034**	1038	1038	1036	1036	opt	9.42	5.64	6.17	2.23	2.19	2.16
cpmp18	100	10	**1043**	opt	opt	opt	opt	opt	3.86	1.70	6.44	2.05	1.59	2.26
cpmp19	100	10	**1031**	1032	1032	opt	opt	opt	4.28	0.95	2.54	6.51	3.09	2.42
cpmp20	100	10	**1005**	1009	1007	1007	opt	opt	72.64	39.91	46.61	51.06	38.81	64.49
Average									6.83	3.20	4.00	4.28	3.38	4.71
sjc1	100	10	**17,288.99**	opt	opt	opt	opt	opt	4.89	1.01	1.25	1.43	1.35	1.90
sjc2	200	15	**33,270.94**	33,293.40	33,293.40	33,293.40	opt	opt	11.46	1.37	1.95	2.01	1.70	3.25
sjc3a	300	25	**45,335.16**	45,384.49	45,338.01	45,338.01	45,338.01	opt	62.20	17.41	17.08	22.58	21.86	23.96
sjc3b	300	30	**40,635.90**	40,666.09	opt	opt	opt	opt	16.14	2.40	1.76	2.28	2.41	2.47
sjc4a	402	30	**61,925.51**	opt	opt	opt	opt	opt	215.60	17.00	14.19	37.87	46.90	56.67
sjc4b	402	40	**52,458.02**	opt	opt	opt	opt	opt	35.90	3.89	6.11	7.88	7.56	6.26
Average									57.70	7.18	7.06	12.34	13.63	15.75

Table 3
Performance comparison of IRMA and approaches from the literature

			SS[a]		VNS[b]		CS[c]		Fen-Cplex[d]		IRMA[e] ($\alpha = 2.4$)	
ID	n	p	Z	Time	Z	Time	Z	Time	Z	Time	Z^*	Time
sjc1	100	10	**17,288.99**	60.00	**17,288.99**	50.50	**17,288.99**	22.72	**17,288.99**	37.60	**17,288.99**	1.90
sjc2	200	15	33,293.40	600.00	**33,270.94**	44.08	**33,270.94**	112.81	**33,270.94**	127.90	**33,270.94**	3,25
sjc3a	300	25	45,338.02	2307.00	**45,335.16**	8580.30	**45,335.16**	940.75	**45,335.16**	495.10	**45,335.16**	23.96
sjc3b	300	30	**40,635.90**	2308.00	**40,635.90**	2292.86	**40,635.90**	1887.97	**40,635.90**	72.20	**40,635.90**	2.47
sjc4a	402	30	**61,925.52**	6109.00	**61,925.51**	4221.47	61,928.72	2885.11	**61,925.51**	1209.50	**61,925.51**	56.67
sjc4b	402	40	52,531.46	6106.00	52,469.96	3471.44	52,531.27	7626.33	**52,458.00**	669.70	**52,458.00**	6.26
Average			41,835.55	2915.00	41,821.08	3110.11	41,831.83	2245.95	**41,819.08**	435.33	**41,819.08**	15.75

[a] Intel Celeron 2.2 GHz.
[b] Pentium IV 3.2 GHz and CPLEX 10.1.
[c] Pentium IV 3.02 GHz.
[d] ASUS S5 notebook 1.6 GHz and CPLEX 9.0.
[e] Intel i5-2300 2.80 GHz and CPLEX 12.3.

According to the data in Table 2, even for small values of α, IRMA provides good solutions in less computational time than the full model. For example, using $\alpha = 1.6$ the time to solve the reduced model is less than half of the time to solve the full model for the instance set 1 and around 12.4% for the instance set 2. The worst gap to the optimal solution is around 1% in instance cpmp03. IRMA found optimal solutions for $\alpha \geq 2.4$ with an average of approximately 69% and 27% of the time to solve the full model for instance sets 1 and 2, respectively. Note that the value of α impacts the solution quality, that is, the larger the value of α, the better the solution obtained. Clearly, this expanded factor can be defined for a specific group or instance depending of the accuracy required and/or the available computational time. Considering the diversity of the instances covered in this paper, we empirically suggest to use $\alpha = 2$ by default and for small instances, such as sets 1 and 2, $\alpha = 2.4$.

We also observed a great capacity of CPLEX to solve these sets of instances, showing that given the current computational power, these instances are outdated and the use of heuristics and metaheuristics only in these sets of instances is not justified.

Table 3 provides a comparison of the performance of our proposed approach against the best results in the recent literature for those instances (optimal solution value is given in boldface).

The first columns give the instance identifier. For each approach, we show the objective function and computational time, respectively. The first approach, SS, is a hybrid metaheuristic that combines scatter search with path relinking proposed by Scheuerer and Wendolsky (2006). The VNS approach described in Fleszar and Hindi (2008) uses variable neighborhood search combined with CPLEX version 10.1. The third approach is described by Chaves et al. (2007) and named CS. "Fen-Cplex" is proposed by Boccia et al. (2007) based on the Fenchel cuts solving by CPLEX. Finally, we show the results obtained with our approach.

Table 3 shows that the computational time of IRMA is significantly smaller than those of the other approaches, even if we would account for the different hardware platforms. IRMA with $\alpha = 2.4$ finds the optimal solutions for all instances.

The comparison in Table 3 is presented for different hardwares and CPLEX versions. To provide a fair comparison, we evaluated the reduction of the computational time of our approach relative to CPLEX computational time to solve the full model. The same comparison is made with the results reported in Boccia et al. (2007), with the "Fen-Cplex" approach. Both comparisons are presented in Table 4, where the columns "CPLEX" present the computational time to solve the full model in the respective computer and solver version. The columns "Time (seconds)" show the computational time in seconds to solve the respective approach. The columns "Time gap" present the percentage of computation time reduction of the proposed approach over the respective time to solve the full model (the best value between both approaches is given in bold type).

From the results in Table 4, we observe that the gap of reduction time of our approach is higher than the one obtained by Boccia et al. (2007) in most cases. The average reduction is notably higher, showing that our approach overcame the previous methods. However, the main advantage of our method is the ability to deal with large-scale instances, which in many cases is impracticable with the full model.

Table 4
Computational time of CPLEX, "Fen-Cplex," and IRMA

			Fen-Cplex[a]			IRMA[b]		
ID	n	p	CPLEX	Time (seconds)	Time gap (%)	CPLEX	Time (seconds)[c]	Time gap (%)
sjc1	100	10	58.9	37.6	36.2	4.9	1.9	**61.1**
sjc2	200	15	214.0	127.9	40.2	11.5	3.3	**71.6**
sjc3a	300	25	640.6	495.1	22.7	62.2	24.0	**61.5**
sjc3b	300	30	148.9	72.2	51.5	16.1	2.5	**84.7**
sjc4a	402	30	5244.2	1209.5	**76.9**	215.6	56.7	73.7
sjc4b	402	40	1068.6	669.7	37.3	35.9	6.3	**82.6**
Average			1229.2	435.3	44.2	57.7	15.8	**72.7**

[a]ASUS S5 notebook 1.6 GHz and CPLEX 9.0.
[b]Intel i5-2300 2.80GHz and CPLEX 12.3.
[c]$\alpha = 2.4$.

Table 5
Computational results of SS-PR and IRMA for the instance set 3

	SS-PR[a]		Phase 2		Phase 3[b]				
Instance	Z	Time (seconds)	Avg	Time (seconds)	Min.	Avg	Max.	SD	Time (seconds)
*spain*737_74_1	8967	9717.45	9049.00	517.66	8845	8875.60	8914	21.85	1131.32
*spain*737_74_2	8970	57,239.12	8988.50	517.62	8870	8894.00	8949	27.63	979.12
*spain*737_148_1	6012	43,361.02	5906.20	516.58	**5901**	5902.30	5905	1.49	652.34
*spain*737_148_2	6009	17,714.15	5919.20	516.76	**5914**	5917.00	5920	1.41	653.79
Average	7489.5	32,007.94	7465.73	517.16	7382.50	7397.23	7422.00	13.10	854.14

[a]Sun Blade 1000/750.
[b]Intel i5-2300 2.80GHz and CPLEX 12.3.

4.2. Results for instance set 3

Table 5 shows a comparison between the results obtained by the hybrid scatter search and path relinking (SS-PR), proposed by Díaz and Fernández (2006), and the average results for 10 runs of IRMA for the third set of instances.

The first column gives the instance name followed by the objective function values and computational times reported by Díaz and Fernández (2006). The objective function value found in the second phase and the cumulative computational time of the first and second phase of the IRMA are shown in the fourth and fifth columns, respectively. The objective function value for the best solution found and average, maximum, standard deviation and average computational time to compute all phases are shown in the last five columns of this table (optimal solution value is given in boldface), respectively.

We observed that the computational time for instances with 148 medians was smaller than the other two instances with 74 medians. This happened as phase 2 obtained good results in the same time, showing to be more favorable in instances with low ratio p/n.

Table 6
Computational results of column-generation algorithm and IRMA for instance set 4

			Column generation[a]		IRMA[b]				
ID	n	p	Lower bound	Total time (seconds)	Min.	Avg	Max.	SD	Avg time (seconds)
p3038_600	3038	600	122,020.66	59,593.02	122,711.17	122,724.79	122,743.15	11.44	2685.38
p3038_700	3038	700	108,685.59	46,705.53	109,677.30	109,695.61	109,729.94	17.04	2239.84
p3038_800	3038	800	98,530.99	33,844.27	100,064.94	100,084.41	100,105.03	13.05	2819.26
p3038_900	3038	900	90,239.65	25,306.54	92,310.09	92,317.78	92,335.74	9.18	1578.17
p3038_1000	3038	1000	83,231.58	20,210.25	85,854.05	85,856.85	85,864.74	3.02	1874.08
Average			100,541.69	34,738.52	102,123.51	102,135.89	102,155.72	10.75	2239.35

[a]Sun Ultra30 workstation.
[b]Intel i5-2300 2.80GHz and CPLEX 12.3.

In comparison with the results reported by Díaz and Fernández (2006), we have improved the results for all instances. Solutions found in phase 2 were already better for instances with 148 medians. Although the data reported are from different hardware platforms, the improved best-known solution and the small computational time showed that our approach outperformed the SS-PR method.

In a further experiment, we used CPLEX to solve the full model. For the first and second instance with 148 medians the solver proved the optimality in around of 20, 000 and 10, 000 seconds, respectively. The solver required more memory than available in the system for the instances with 74 medians, and it stopped after around 10,000 seconds with a best integer solution worse than the best we found in phase 2.

4.3. Results for instance set 4

Table 6 details the average for 10 runs of IRMA for the fourth set of instances. The first three columns give the instance name, the number of nodes, and the number of medians, respectively. The column-generation algorithm proposed by Lorena and Senne (2004) shows the values and the computational time to compute the lower bound to this set of instances. The minimum, average, maximum, and standard deviation values of the objective function, plus an average runtime in seconds to compute all phases of our procedure are shown in the last five columns.

We can observe that the average objective function value is very close to the best solutions found, which are shown in column "Min," resulting in relatively low values of the standard deviation.

Regarding the average computational time, we can conclude that the proposed procedure is efficient to find good quality solution in an acceptable computational time even on large-scale instances. Compared to the lower bound provided by the column-generation approach, the smallest gap is 0.57% for instance p3038_600 and the largest gap is 3.15% for instance p3038_1000.

Table 7
Computational results of IRMA for instance set 5

				Phase 2			Phase 3				
ID	n	p	BKS	Avg	GAP	Time (seconds)	Min.	Avg	Max.	GAP	Time (seconds)
lin318_005	318	5	**180,281.20**	180,281.20	0.00	9.15	–	–	–	–	–
lin318_015	318	15	**88,901.56**	88,901.56	0.00	26.35	–	–	–	–	–
lin318_040	318	40	**47,988.38**	48,040.24	1.01	222.41	47,988.38	48,003.88	48,061.18	0.14	319.46
lin318_070	318	70	**32,198.64**	32,290.39	0.01	127.45	–	–	–	–	–
lin318_100	318	100	**22,942.69**	23,639.70	2.23	222.65	22,942.69	22,942.69	22,942.69	0.00	364.87
ali535_005	535	5	**9956.77**	10,337.57	0.00	7.08	9956.77	10,210.58	11,225.83	0.00	45.42
ali535_025	535	25	**3695.15**	3770.45	0.24	311.36	3695.15	3701.88	3731.91	0.00	544.26
ali535_050	535	50	2461.41	2497.05	1.69	377.28	2464.01	2478.04	2483.21	0.00	726.30
ali535_100	535	100	1438.42	1454.48	2.61	362.75	1441.26	1448.00	1454.40	0.02	637.64
ali535_150	535	150	1032.28	1044.60	2.54	366.54	1035.52	1037.70	1042.77	0.00	761.31
u724_010	724	10	181,782.96	184,031.19	0.00	6.64	181,782.96	182,611.19	185,551.93	0.00	59.65
u724_030	724	30	**95,034.01**	96,513.51	0.01	158.05	95,034.01	95,159.96	95,494.15	0.00	300.72
u724_075	724	75	54,735.05	54,742.43	0.08	507.56	54,735.05	54,735.05	54,735.05	0.00	546.39
u724_125	724	125	38,976.76	38,992.44	0.28	509.01	38,976.76	38,976.76	38,976.76	0.02	643.31
u724_200	724	200	28,079.97	28,117.06	0.10	508.81	28,079.97	28,082.72	28,089.37	0.11	706.29
rl1304_010	1304	10	2,146,484.10	2,172,994.88	0.00	123.88	2,146,484.10	2,166,552.03	2,202,693.81	0.00	181.66
rl1304_050	1304	50	802,283.41	825,624.76	0.11	752.11	803,106.89	806,425.28	810,323.02	0.00	1199.98
rl1304_100	1304	100	498,090.74	503,281.32	0.15	837.11	498,093.12	498,411.69	498,721.16	0.01	1634.18
rl1304_200	1304	200	276,977.60	277,163.25	0.44	874.06	276,977.60	276,983.73	276,997.47	0.03	1227.78
rl1304_300	1304	300	191,224.85	191,339.81	0.34	746.01	191,224.85	191,258.69	191,400.09	0.00	951.75
pr2392_020	2392	20	2,235,376.73	2,283,429.04	0.02	439.03	2,236,720.55	2,250,292.41	2,270,032.30	0.00	551.82
pr2392_075	2392	75	1,092,294.02	1,119,259.20	0.00	518.49	1,094,979.83	1,098,559.96	1,105,430.00	0.00	825.85
pr2392_150	2392	150	711,111.25	720,668.86	0.10	1477.13	711,128.32	711,315.15	711,656.01	0.00	2019.23
pr2392_300	2392	300	458,145.29	459,186.43	0.54	1011.66	458,157.54	458,221.63	458,298.21	0.01	2382.39
pr2392_500	2392	500	316,042.97	316,967.13	0.77	1220.51	316,046.35	316,092.46	316,230.72	0.00	2402.60
fnl4461_0020	4461	20	1,283,536.73	1,301,186.19	0.00	63.69	1,283,536.73	1,292,621.57	1,301,776.37	0.00	538.97
fnl4461_0100	4461	100	548,909.01	560,854.73	0.06	3140.40	548,909.01	550,758.21	552,728.58	0.00	3880.37
fnl4461_0250	4461	250	335,888.87	339,758.98	0.15	3152.33	335,888.87	336,006.96	336,184.11	0.00	4592.66
fnl4461_0500	4461	500	224,662.49	224,788.81	0.19	3159.85	224,664.24	224,684.37	224,723.78	0.00	3912.36
fnl4461_1000	4461	1000	145,862.38	145,903.53	0.16	2539.32	145,862.47	145,870.78	145,879.02	0.00	3433.26

4.4. Results of IRMA for instance set 5

In this subsection, we report the computational results of IRMA for the instance set 5. Table 7 shows the results of phases 2 and 3, where the first three columns refer to information of the instance, such as name, number of nodes and medians, respectively. The next column provides the objective function value of the Best Know Solution (BKS) obtained in all experiments, including the not reported experiments to calibrate the parameters. Boldface entries correspond to optimal values.

For the second and third phases, we report the minimum, average and maximum values of the objective function for 10 runs, represented by the columns "min, avg, and max," respectively. Column "GAP" shows the average gap for each instance. This value is calculated by $GAP = (Z - BKS) * 100/BKS$, where, Z is the objective function value under consideration and BKS is the best-known feasible solution. Finally, the column "Time" presents the average time to complete each phase from the start, it means that the last column corresponds to the average total execution time of the algorithm.

We observed that in phase 2 the highest average gap occurred on instance set $ali535$. We attributed it to the fact that this instance set has some very scattered demand points and others grouped in certain regions. Phase 2 had an average gap lower than 2.61%, and an average total value of the gap equal to 0.47%. Phase 3 proved to improve the solutions found in phase 2 and also in obtaining the best-known solutions for these instances.

5. Conclusions

In this paper, we propose a matheuristic called "Iterated Reduction Matheuristic Algorithm" (IRMA) to solve the CPMP. The proposed approach is composed of three phases based on two size-reduction methods to eliminate variables of the mathematical model and an iterative partial optimization algorithm in which subproblems are solved by an MIP solver.

The computational experiments were performed using five sets of instances in order to validate the performance of the proposed method. For small- and medium-sized instances, for which current state-of-the-art MIP solvers find the optimal solution for the vast majority of instances using the full model, IRMA requires less computation time at the expense of a small loss in solution quality. The strategies proposed in this paper were efficient for the instance sets 1 and 2 because they were able to find the optimal solutions in less computation time than the best methods from the literature. Regarding instance set 3, we not only found solutions with better objective function value, but did so in less computational time.

Furthermore, IRMA can solve large size instances with a number of nodes that had not yet been addressed in the literature in realistic computational time. In instance set 4, we provided feasible solutions close to the best-known lower bounds, and in set 5 we provided new and diverse instances and solutions for future comparison. In most of the instances of the sets 4 and 5, the solver was not able to obtain the optimal solution, justifying the use of heuristics methods.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper.

The first author was partially supported by CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior), Brazil.

References

Ahmadi, S., Osman, I.H., 2005. Greedy random adaptive memory programming search for the capacitated clustering problem. *European Journal of Operational Research* 162, 1, 30–44.

Baldacci, R., Hadjiconstantinou, E., Maniezzo, V., Mingozzi, A., 2002. A new method for solving capacitated location problems based on a set partitioning approach. *Computers & Operations Research* 29, 4, 365–386.

Boccia, M., Sforza, A., Sterle, C., Vasilyev, I., 2007. A cut and branch approach for the capacitated p-median problem based on Fenchel cutting planes. *Journal of Mathematical Modelling and Algorithms* 7, 1, 43–58.

Büdenbender, K., Grünert, T., Sebastian, H.-J., 2000. A hybrid tabu search/branch-and-bound algorithm for the direct flight network design problem. *Transportation Science* 34, 4, 364–380.

Chaves, A., Assis Correa, F., Lorena, L., 2007. Clustering search heuristic for the capacitated p-median problem. In Corchado, E., Corchado, J., Abraham, A. (eds) *Innovations in Hybrid Intelligent Systems,* Vol. 44 of *Advances in Soft Computing*. Springer, Berlin/Heidelberg, pp. 136–143.

Díaz, J.A., Fernández, E., 2006. Hybrid scatter search and path relinking for the capacitated p-median problem. *European Journal of Operational Research* 169, 2, 570–585.

Dumitrescu, I., Stützle, T., 2003. Combinations of local search and exact algorithms. In Cagnoni, S., Johnson, C., Cardalda, J., Marchiori, E., Corne, D., Meyer, J.-A., Gottlieb, J., Middendorf, M., Guillot, A., Raidl, G., Hart, E. (eds) *Applications of Evolutionary Computing.* Vol. 2611 of *Lecture Notes in Computer Science*. Springer, Berlin/Heidelberg, pp. 211–223.

Fanjul-Peyro, L., Ruiz, R., 2011. Size-reduction heuristics for the unrelated parallel machines scheduling problem. *Computers & Operations Research* 38, 1, 301–309.

Fernandes, S., Lourenço, H.R., 2006. Optimised search heuristics: a mapping of procedures and combinatorial optimisation problems. Technical Report, Universidade do Algarve, Faro, Portugal.

Fischetti, M., Lodi, A., 2003. Local branching. *Mathematical Programming* 98, 1–3, 23–47.

Fleszar, K., Hindi, K., 2008. An effective VNS for the capacitated p-median problem. *European Journal of Operational Research* 191, 3, 612–622.

Garey, M.R., Johnson, D.S., 1979. Computers and intractability: a guide to the theory of NP-completeness. W.H. Freeman & Co., New York, NY.

Jourdan, L., Basseur, M., Talbi, E.-G., 2009. Hybridizing exact methods and metaheuristics: a taxonomy. *European Journal of Operational Research* 199, 3, 620–629.

Lorena, L.A.N., Pereira, M.A., Salomao, S.N.A., 2003. A relaxação Lagrangeana/surrogate e o método de geração de colunas: novos limitantes e novas colunas. *Pesquisa Operacional* 23, 1, 29–47.

Lorena, L.A.N., Senne, E.L.F., 2002. Abordagens de Geração de Colunas para um Problema de p-medianas Capacitado. In: *Proceedings of XXXIV SBPO–Simpo´sio Brasileiro de Pesquisa Operacional*, 11.

Lorena, L.A.N., Senne, E.L.F., 2004. A column generation approach to capacitated p-median problems. *Computers & Operations Research* 31, 6, 863–876.

Maniezzo, V., Mingozzi, A., Baldacci, R., 1998. A bionomic approach to the capacitated p-median problem. *Journal of Heuristics* 4, 3, 263–280.

Maniezzo, V., Stützle, T., Voß, S., 2010. *Matheuristics: Hybridizing Metaheuristics and Mathematical Programming* (1st edn). Springer, Berlin.

Mautor, T., Michelon, P., 1997. MIMAUSA: a new hybrid method combining exact solution and local search. Proceedings of the 2nd International Conference on Metaheuristics, Sophia Antipolis, France, Vol. 1, p. 15.

Mautor, T., Michelon, P., 2001. MIMAUSA: an application of referent domain optimization. Rapport technique, Laboratoire dAvignon.

Mulvey, J.M., Beck, M.P., 1984. Solving capacitated clustering problems. *European Journal of Operational Research* 18, 3, 339–348.

Nievergelt, J., 2000. Exhaustive search, combinatorial optimization and enumeration: exploring the potential of raw computing power. In Hlavác, V., Jeffery, K., Wiedermann, J. (eds) *SOFSEM 2000: Theory and Practice of Informatics.* Vol. 1963 of *Lecture Notes in Computer Science*, Springer, Berlin, pp. 87–125.

Osman, I.H., Christofides, N., 1994. Capacitated clustering problems by hybrid simulated annealing and tabu search. *International Transactions in Operational Research* 1, 3, 317–336.

Pirkul, H., 1987. Efficient algorithms for the capacitated concentrator location problem. *Computers & Operations Research* 14, 3, 197–208.

Puchinger, J., Raidl, G., 2005. Combining metaheuristics and exact algorithms in combinatorial optimization: a survey and classification. In Mira, J., Álvarez, J. (eds) *Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach.* Vol. 3562 of *Lecture Notes in Computer Science*. Springer, Berlin, pp. 41–53.

Scheuerer, S., Wendolsky, R., 2006. A scatter search heuristic for the capacitated clustering problem. *European Journal of Operational Research* 169, 2, 533–547.

Taillard, E.D., Voss, S., 2002. POPMUSIC—partial optimization metaheuristic under special intensification conditions. In Ribeiro, C.C., Hansen, P. (eds) *Essays and Surveys in Metaheuristics SE-27.* Vol. 15 of *Operations Research/ Computer Science Interfaces Series*. Springer, New York, pp. 613–629.

Talbi, E.-G., 2002. A taxonomy of hybrid metaheuristics. *Journal of Heuristics* 8, 5, 541–564.