J Heuristics (2009) 15: 597–615
DOI 10.1007/s10732-008-9078-y

An effective heuristic for large-scale capacitated facility location problems

Pasquale Avella · Maurizio Boccia · Antonio Sforza · Igor Vasil'ev

Received: 13 May 2004 / Revised: 19 November 2007 / Accepted: 22 April 2008 /
Published online: 23 May 2008

Abstract The Capacitated Facility Location Problem (*CFLP*) consists of locating a set of facilities with capacity constraints to satisfy the demands of a set of clients at the minimum cost. In this paper we propose a simple and effective heuristic for large-scale instances of *CFLP*. The heuristic is based on a Lagrangean relaxation which is used to select a subset of "promising" variables forming the core problem and on a Branch-and-Cut algorithm that solves the core problem. Computational results on very large scale instances (up to 4 million variables) are reported.

Keywords Capacitated facility location problem · Lagrangean relaxation · Branch-and-cut

P. Avella · M. Boccia
RCOST—Research Center on Software Technology, Università del Sannio, Viale Traiano, 82100 Benevento, Italy
e-mail: avella@unisannio.it

M. Boccia
e-mail: maurizio.boccia@unisannio.it

A. Sforza (✉)
Dipartimento di Informatica e Sistemistica, Università degli Studi di Napoli "Federico II", Via Claudio 21, 80125 Napoli, Italy
e-mail: sforza@unina.it

I. Vasil'ev
Institute of System Dynamics and Control Theory, Siberian Branch of Russian Academy of Sciences, Lermontov Srt., 134, 664033 Irkutsk, Russia
e-mail: igor@diima.unisa.it

Introduction

The *Capacitated Facility Location Problem* (*CFLP*) consists of locating a set of facilities with capacity constraints, with the aim of satisfying the demands of a set of clients at the minimum cost.

CFLP has been widely addressed in literature. Approximation results for instances where the service costs obey the triangle inequality have been proposed by Korupolu et al. (1998) and by Chudak and Williamson (1999), who proved that a local search heuristic produces solutions which are within a factor of $6(1+\varepsilon)$ of the value of an optimal solution. In Cornuejols et al. (1991) a comparison between greedy, interchange and Lagrangean heuristics is reported. A Lagrangean heuristic for larger instances (up to 500 facilities and 1000 clients) is described in Beasley (1988, 1993). In Barahona and Chudak (1999) a new heuristic is proposed, where a randomized rounding technique is embedded into a new generation Lagrangean solver, the Volume Subgradient, which generates both primal and dual solutions (Anbil and Barahona 2000). The authors report on computational experience showing that instances up to 1000 facilities and 1000 clients are solved, yielding duality gaps never exceeding 1.25%.

Polyhedral properties of *CFLP* have been investigated in depth by Aardal et al. (1995). For a detailed analysis we refer the reader to Aardal (1992). Separation algorithms and a cutting plane algorithm are proposed in Aardal (1998).

Due to the progress of hardware and of MIP solver technology, it is now possible to solve *CFLP* instances with up to 300 facilities and 300 clients on personal computers. Above this problem size exact solution become prohibitive because of the difficulty of solving the LP-relaxation.

The goal of this paper is to propose a simple and efficient heuristic for the approximate solution of large-scale *CFLP* instances. The key idea of our approach is to solve a Lagrangean relaxation to select a subset of "promising" variables, forming a *core problem*. The Lagrangean function is maximized by subgradient optimization. A *subgradient deflection* technique and a new *step-size rule* are introduced to achieve convergence to a fairly good lower bound in less than 250 iterations. A Branch-and-Cut algorithm is finally applied to the core problem to yield an upper bound for *CFLP*.

The approach has proved to be efficient: it yields the optimal solution for instances up to 300 facilities and 300 clients and allows instances up to 1000 facilities and 1000 clients to be solved to 1%-optimality in less than 100 seconds. Instances with 2000 facilities and 2000 clients are solved to 2.3%-optimality in 8 minutes at most. Computational experiments were carried out on a Pentium IV 1.8 GHz personal computer with 1 GB RAM.

Section 1 describes the formulation of *CFLP*. Section 2 presents the general structure of the algorithm. Section 3 describes the Lagrangean relaxation. Section 4 presents the core problem selection procedure. Finally, Sect. 5 reports computational tests on medium and large-scale instances, comparing the results with those yielded by other approaches.

1 Problem formulation

Let $I = \{1, \dots, n\}$ be a set of facilities, let $J = \{1, \dots, m\}$ be a set of clients and let $G(I \cup J, A)$ be a complete bipartite graph where A is a set of arcs (i, j) with $i \in I$ and $j \in J$. Let d_j be the j-th client demand, let c_{ij} be the cost of sending one unit of flow from facility i to client j and let f_i be the fixed cost of opening facility i. Every facility i has a capacity s_i.

CFLP is the problem of choosing a feasible subset $S \subseteq I$ of open facilities which minimizes the sum of the transportation and fixed costs.

Let y_i be the binary variable associated with each facility $i \in I$. It is 1 if facility i is open ($i \in S$), 0 otherwise. Let x_{ij} be a continuous variable expressing the fraction of client j's demand assigned to facility i. A formulation of *CFLP* is:

$$\begin{aligned}
\min \quad & \sum_{i \in I} f_i y_i + \sum_{i \in I} \sum_{j \in J} c_{ij} d_j x_{ij} && \\
\text{s.t.} \quad & \sum_{i \in I} x_{ij} = 1, \quad j \in J && (1)\\
& \sum_{j \in J} d_j x_{ij} \le s_i y_i, \quad i \in I && (2)\\
& x_{ij} \le y_i, \quad i \in I, j \in J && (3)\\
& x_{ij} \ge 0, \quad i \in I, j \in J && \\
& y_i \in \{0, 1\}, \quad i \in I &&
\end{aligned}$$

Constraints (1) ensure that the whole demand of each client must be satisfied. Capacity constraints (2) ensure that the total demand supplied from a facility does not exceed its capacity. Variable Upper Bounds (3) ensure that no client can be supplied from a closed facility.

2 Main steps of the heuristic

It is well-known that the linear relaxation of the *CFLP* formulation (1)–(3) yields a good lower bound for the optimal solution of the problem.

Let $O = \{y_i^* : i \in I\}$ be the optimal solution of *CFLP*, let $\delta(i)$ be the reduced cost of y_i in the optimal solution of the LP-relaxation, let γ be a given threshold and let $L = \{i \in I : \delta(i) < \gamma\}$. Preliminary computational experience has shown that the sets O and L have a broad overlap, as reported in Table 1 for 12 sets of small instances, each composed of 5 randomly generated instances with the same dimension and structure. For these small instances the optimal solution was obtained using the commercial Xpress Branch-and-Cut.

The instances were generated according to the procedure described in Sect. 5, and we set $\gamma = 0.1$. Column $|I|$ reports on the number of facilities of the corresponding set of instances. Column $\frac{|O \cap L|}{|O|}$ reports on the average value of the ratio $\frac{|O \cap L|}{|O|}$. As we can see, this value is always greater than 99% and this suggests that reduced costs

Table 1 Comparison between LP relaxation and optimal solution for small instances

Name	P200-2	P250-2	P300-2	P200-3	P250-3	P300-3	P200-5	P250-5	P300-5	P200-10	P250-10	P300-10
$\|I\|$	200	250	300	200	250	300	200	250	300	200	250	300
$\frac{\|O \cap L\|}{\|O\|}$	99.68	99.59	99.51	99.49	99.52	99.40	99.68	99.31	99.38	99.71	99.53	99.37

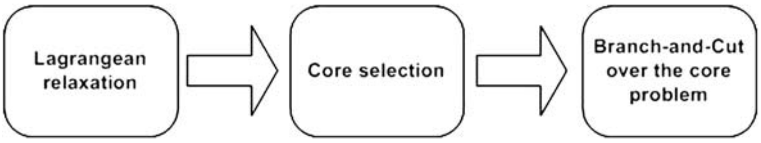

Fig. 1 The three stages of the heuristic

are useful in selecting a set of promising variables. The proposed heuristic of three stages (Fig. 1). In the first stage a *Lagrangean relaxation* of the formulation (1)–(3) is solved to achieve a lower bound. We maximize the Lagrangean dual function by the subgradient method, whose efficiency is improved by a subgradient deflection technique and by a new step-size rule. These settings allow convergence in less than 250 iterations to a fairly good lower bound. Computational experience on small-medium size instances ($|I|, |J| \leq 300$) shows that the Lagrangean bound differs at most by 0.2% from the optimal value of the LP-relaxation.

In the second stage, the outcomes of the Lagrangean relaxation are exploited to select a *core problem*, defined by the pair $(S, X(S))$, where $S \subseteq I$ and $X(S) \subseteq S \times J$.

The third stage consists of running a *Branch-and-Cut* algorithm over the core problem, truncated as soon as 1%-optimal upper bound is found.

The following sections describe each stage of the heuristic in more detail.

3 Lagrangean relaxation of *CFLP*

Consider the Lagrangean relaxation of the demand constraints (see Beasley 1993; Cornuejols et al. 1991).

For a given set $\bar{\lambda}$ of multipliers, the Lagrangean dual function is $\theta(\bar{\lambda}) = \sum_{i \in I} \theta_i(\bar{\lambda}) + \sum_{j \in J} \bar{\lambda}_j$, where each $\theta_i(\bar{\lambda})$ is computed independently (Barahona and Chudak 1999):

$$\begin{aligned} \theta_i(\bar{\lambda}) = \min\ & f_i y_i + \sum_{j \in J} (c_{ij} d_j - \bar{\lambda}_j) x_{ij} \\ \text{s.t.}\ & \sum_{j \in J} d_j x_{ij} \leq s_i y_i \\ & x_{ij} \leq y_i, \quad j \in J \\ & x_{ij} \geq 0, \quad j \in J \\ & y_i \in \{0, 1\} \end{aligned} \tag{4}$$

To find the optimal solution of the previous problem $\theta_i(\bar{\lambda})$, we can initially suppose that $y_i = 1$ and solve the following:

$$\begin{aligned} \min \quad & f_i + \sum_{j \in J} (c_{ij} d_j - \bar{\lambda}_j) x_{ij} \\ \text{s.t.} \quad & \sum_{j \in J} d_j x_{ij} \leq s_i \\ & x_{ij} \leq y_i, \quad j \in J \\ & x_{ij} \geq 0, \quad j \in J \end{aligned} \tag{5}$$

For any optimal solution of this problem $x_{ij} = 1$ only if $(c_{ij}d_j - \bar{\lambda}_j) < 0$, so we can invert the direction of the objective function and consider only the variables x_{ij} with $(\bar{\lambda}_j - c_{ij}d_j) > 0$. In this way we obtain the LP-relaxation of a knapsack problem, solvable in linear time by the algorithm of Balas and Zemel (1980).

Let $\{x_{ij}^*, j \in J\}$ be the optimal solution of the problem (5), the optimal solution of the problem (4) is:

$$y_i = 1 \quad \text{and} \quad x_{ij} = x_{ij}^* \quad \forall j \in J \text{ if } f_i + \sum_{j \in J} (c_{ij} d_j - \bar{\lambda}_j) x_{ij}^* < 0$$

$$y_i = 0 \quad \text{and} \quad x_{ij} = 0 \quad \forall j \in J \text{ otherwise.}$$

Finally, we use the subgradient method to determine the set λ^* maximizing the Lagrangean function $\theta(\lambda)$. The standard subgradient method requires generating a sequence of multipliers:

$$\lambda_{k+1} = \lambda_k + t_k h_k$$

where t_k is the step size and the moving direction h_k is the subgradient g. For any $j \in J$, the j-th component of the subgradient is:

$$g_j = 1 - \sum_{i \in I} x_{ij} \tag{6}$$

The step size t_k defined by:

$$t_k = \frac{\varphi(Z_{UB} - \theta(\lambda^*))}{\|h_k\|^2} \tag{7}$$

where Z_{UB} is an upper bound to the original problem, $\theta(\lambda^*)$ is the best value of $\theta(\lambda)$ found so far and φ is a constant parameter usually initialized to 2 and halved if a given number of iterations (say 20–30) of the subgradient procedure have been performed without improving the lower bound (Beasley 1993).

3.1 Subgradient deflection and step-size rule

In order to improve the efficiency of the subgradient method, we modified its standard implementation by adopting:

a) a subgradient deflection technique;
b) a new step size rule.

Deflection techniques are known to significantly improve the convergence of subgradient algorithms (Camerini et al. 1975). In our case, at the generic iteration k we compute the moving direction h_k by the following formula:

$$h_k = \frac{(g_k + 0.3h_{k-1} + 0.1h_{k-2})}{1.4}$$

where g_k is the current subgradient vector at the generic iteration k, and h_{k-1} and h_{k-2} are the directions used in the last two iterations. To also compute h_k in the first two iterations, we set $h_{-1} = h_{-2} = 0$. Moreover at the first iteration we set the Lagrangean multipliers to 0.

For the step-size rule, we run an extensive computational experience showing that a more effective updating strategy of φ consists of "gradually" reducing the value of φ at each iteration, independently of the behavior of $\theta(\lambda)$ (see Sect. 5). In this way the Lagrangean heuristic converges very quickly to fairly good lower bounds.

3.2 Upper bound computation

Heuristics based on Lagrangean relaxation usually provide both a lower bound and upper bound (value of a feasible solution) to the MIP problem. During the subgradient optimization, an upper bound heuristic can be repeatedly performed (Beasley 1993). Let S be the set of open facilities related to the current lower bound solution. An upper bound to the problem solution can be computed by solving the following transportation problem where the origins are the facilities in S and the destinations are the clients in J:

$$\begin{aligned} \min \quad & \sum_{i\in S}\sum_{j\in J} c_{ij} v_{ij} + \sum_{i\in S} f_i \\ \text{s.t.} \quad & \sum_{i\in S} v_{ij} = d_j, \quad j \in J \qquad (8) \\ & \sum_{j\in J} v_{ij} \le s_i, \quad i \in S \qquad (9) \\ & v_{ij} \ge 0, \quad i \in S, j \in J \end{aligned}$$

where $v_{ij} = d_j x_{ij}$ denotes the amount of flow from the facility i to the client j.

As will be shown in Sect. 5, the number of calls to transportation problems required to generate good upper bounds makes such an approach inefficient. We therefore prefer using the Lagrangean relaxation only to yield a lower bound of the problem, computing a good quality upper bound in the third stage of the procedure, which we address in the next sections.

We solve the problem (8)–(9) only once, before starting the subgradient optimization, to initialize Z_{UB} in (7). The starting set S of facilities for the transportation problem (8)–(9) is obtained as a solution of the following knapsack problem:

$$\begin{aligned} \min \quad & \sum_{i \in I} f_i y_i \\ \text{s.t.} \quad & \sum_{i \in I} s_i y_i \geq \sum_{j \in J} d_j \qquad (10) \\ & y_i \in \{0, 1\}, i \in I \end{aligned}$$

where the constraint (10) guarantees that a minimal set of facilities must be open to satisfy the total demand of the clients.

4 Core problem

In this section we present a procedure to select a core problem based on the Lagrangean lower bound solution. The core problem is a reduced size problem defined by a subset $S \subset I$ of the facilities, by the set of clients J, and by a subset $X(S)$ of the arcs leaving the nodes of S. We denote the core problem by the pair $(S, X(S))$. For each $i \in I$, let F_i be the Lagrangean reduced cost of y_i, i.e.

$$F_i = f_i + \sum_{j \in J} (c_{ij} d_j - \lambda_j^*) x_{ij}^*, \quad i \in I \qquad (11)$$

where λ^* is the vector of multipliers computed by the subgradient method and x^* is the corresponding solution of (5).

As pointed out in Sect. 2, the variables with lower LP reduced cost have a high probability of corresponding to open facilities in the optimal solution. Moreover, for near optimal Lagrangean multipliers λ^*, the Lagrangean reduced cost F_i gives reliable information on the overall utility of selecting facility i (Caprara et al. 1997). Based on this property, we use the Lagrangean reduced cost to approximate LP reduced cost. Therefore, in order to select a “promising” pair $(S, X(S))$, we sort the facilities in I in non-decreasing order according to the value F_i, then we choose its elements by extracting them in order from this list by the following rule.

Let r be the ratio between the total capacity of the facilities in I and the total demand of the clients in J. To decide the value of $|S|$, we could observe that $\lceil |I|/r \rceil$ is an upper bound on the minimal number of open facilities needed to satisfy the total client demand. In other words, we can say that at least a feasible solution with less than or equal to $\lceil |I|/r \rceil$ open facilities exists. To estimate the more convenient value of $|S|$ we can observe that the number of open facilities in the optimal solution depends on the cost structure. Let $c_j^{\min} = \min_{i \in I} c_{ij}$ be the minimum unit cost to satisfy the client j. Considering equation (1), the objective function of the *CFLP* formulation (1)–(3) can be written as:

$$\min \sum_{i \in I} f_i y_i + \sum_{i \in I} \sum_{j \in J} (c_{ij} - c_j^{\min}) d_j x_{ij} + \sum_{j \in J} c_j^{\min} d_j \qquad (12)$$

where the last term is a constant.

The cost structure of a *CFLP* instance is characterized by the ratio between the average of the fixed costs f_i and the average of the *scaled* transportation costs $(c_{ij} - c_j^{\min})d_j$. If this ratio is large, the optimization will tend to open as few facilities as possible while, if it is small, the optimization will tend to reduce transportation costs and to open more facilities. So we can refine the estimate of $|S|$ by multiplying $\lceil |I|/r \rceil$ by a factor $w < 1$ if fixed costs are much larger than *scaled* transportation costs, $w \geq 1$ otherwise.

Let $\gamma = \lceil w|I|/r \rceil$. We define S as the set of the first γ facilities sorted in increasing order according to the F_i.

To complete the core definition, we need to describe how the set $X(S)$ is selected. In our heuristic $X(S)$ is defined by setting a threshold value δ and by selecting all the arcs leaving the nodes of S whose reduced cost $\bar{c}_{ij} = c_{ij}d_j - \lambda_j^*$ does not exceed δ.

With the aim of keeping the core size small, we initially set

$$\delta = \text{average}\{\bar{c}_{ij} | i \in S,\ j \in J\} \tag{13}$$

and then we attempt to reduce it until the condition that at least three arcs belonging to $X(S)$ must leave each client is satisfied. It is obvious that this rule cannot guarantee the feasibility of the core problem $(S, X(S))$. However, our experiments showed that the core problem has a very high probability of containing a feasible solution if this condition is satisfied. In any case, if such a feasible solution does not exist, it is possible to reduce δ until the core problem becomes feasible.

The core problem $(S, X(S))$ is solved by a commercial Branch-and-Cut code. The efficiency of the MIP solver has been improved by solving the LP relaxation by the dual simplex, by setting the strong branching option for the variable selection and by activating the built in Cover inequalities.

Our computational experience showed that often the Branch-and-Cut procedure often provides many near-integer solutions. This suggested embedding a *plunging heuristic* into the search tree. It consists of evaluating, in some nodes of the enumeration tree, the feasible solution obtained approximating to the nearest integer value the variables that have to be integer. It is thus possible to find a better upper bound. In our heuristic, at every node where the fractional values of the y_i variables are near-integer, i.e. $y_i < \epsilon$ or $y_i > 1 - \epsilon$ with a suitably small $\epsilon > 0$ for each $i \in S$, feasible solution is obtained by setting the variables to their nearest integer values (in the computational tests we set $\epsilon = 0.1$). We then compute an upper bound by solving the transportation problem (8)–(9) described in Sect. 3.

This plunging heuristic makes the Branch-and-Cut much more robust and efficient. In Table 2 we report the size of the core problem, the number of *B&C* nodes and the computation time spent in solving the core problem to 1%-optimality, respectively without and with plunging, for a representative subset of instances.

5 Computational experience

The following reports our computational experience on three test-beds, A, B and C, with different cost structures. It were carried out on a Pentium IV 1.8 GHz personal

Table 2 Computational results for the plunging heuristic

Name	Ratio variables	$\|X(S)\|$	$\|S\|$ No Plunging	#B&C nodes No Plunging	Time (s) With Plunging	#B&C nodes With Plunging	Time (s)
P1000-1000-20	3	42938	260	237	76.3	1	18.5
P1000-1000-25	5	10312	160	250	68.2	1	12.1
P1000-1000-30	10	6102	85	167	42.8	67	17.6
P2000-2000-20	3	149524	530	233	584.3	12	116.2
P2000-2000-25	5	37828	325	485	116.4	7	21.6
P2000-2000-30	10	19119	170	178	93.3	43	42.5

computer with 1 GB RAM. All the tested instances are available at the web page http://wpage.unina.it/sforza/test.

Test-bed A contains the instances generated as in Cornuejols et al. (1991), Aardal (1998) and Barahona and Chudak (1999). The test generating procedure can be summarized as follows:

a) Points representing the facilities and the clients are uniformly randomly generated in a unit square. Transportation costs per unit flow are then computed by multiplying the euclidean distances among them by 10.
b) Demands are generated from $U[5, 35]$, i.e. they are uniformly distributed in the interval $[5, 35]$.
c) Capacities s_i are generated from $U[10, 160]$ and are then scaled by using the ratio $r = \sum_i s_i / \sum_j d_j$. In our tests r assumes the values: 1.1, 1.5, 2, 3, 5, 10.
d) Fixed costs are generated by the formula $f_i = U[0, 90] + U[100, 110]\sqrt{s_i}$ to take into account the economies of scale.
e) The fixed costs f_i are doubled when $r \leq 2$.

The state-of-the-art results are reported in Barahona and Chudak (1999) where instances up to 1000 facilities have been solved to 1.25%-optimality in about 1 hour on an IBM RISC 6000/7043P-240. The IBM RISC 6000 is an old machine that is much slower than the Pentium IV used for our computational experience, so we can compare the two algorithms exclusively in terms of a duality gap. Instances of the test bed A with 1000 facilities and 1000 clients are solved by our algorithm yielding a duality gap that never exceeds 0.90%, while Barahona and Chudak's algorithm yields a significantly higher duality gap, up to 1.25%.

Due to the transportation cost generation rule (point a), the value of $c_j^{\min}$ for these instances is very small for each client j. In fact, if the number of facilities and clients generated in a unit square is high, the probability that each client is very near to a facility is close to 1. Therefore it is reasonable to suppose $c_j^{\min} = 0, \forall j \in J$ and consider the transportation costs as the *scaled* transportation costs in the evaluation of the cost structure.

In particular, the instances of test-bed A are characterized by average values of transportation costs that are two orders of magnitude smaller than opening fixed costs. This cost structure is realistic but in some cases it can imply that the initial upper bound (see Sect. 3.2) is already a very good upper bound, because the minimization of

Table 3 Computational results for test instances with 1000 facilities and 1000 clients (test bed A)

$\|I\| \times \|J\|$	Ratio	Lagr Iter.	Lagr time (s)	$\|X(S)\|$	$\|S\|$	B&C nodes	B&C Time (s)	%gap	Tot. Time (s)
1000×1000	1.1	300	102.8	–	–	–	–	0.78	102.8
1000×1000	1.5	300	93.5	–	–	–	–	0.68	93.5
1000×1000	2	250	93.4	–	–	–	–	0.77	93.4
1000×1000	3	200	55.4	40352	260	11	27.4	0.36	77.3
1000×1000	5	200	28.2	9883	160	51	18.5	0.37	46.7
1000×1000	10	200	14.8	6130	85	79	23.7	0.43	38.5

the fixed costs play a significant role in the problem solution. To avoid this situation, we generated a second test-bed where the average value of fixed costs is one order of magnitude greater than the average values of the transportation costs. Instances of the test-bed B are obtained by skipping step (e) and by replacing step (a) with the following:

a′) Multiply by 100 the euclidean distances between the points representing the clients and the facilities.

Finally, we have considered also a third test bed C whose instances are characterized by transportation and fixed costs of the same order of magnitude.

The instances in A, B and C are organized into five groups, each containing instances of the same size. The sizes (*number of clients* $\times$ *number of facilities*) here considered are 1000×1000, 2000×2000, 3000×1200, 4000×1000 and 4400×800. These sizes correspond to a ratio R between the number of clients and the number of facilities varying from 1 to 5.5 and generate instances (except those belonging to the first group) with 3.5–4 million variables. Greater instances can generate memory requirement problems. Each group includes six different subsets of problems according to the ratio r (we set $r = 1.1$, 1.5, 2, 3, 5, 10). Five instances have been randomly generated for each value of r.

We used the XPRESS Optimisation Subroutine Library (XOSL) Release 12.5 as MIP solver. It was used not only to solve the core problem but also to solve transportation problems and the binary knapsack problems.

The outcome of the computational experience is reported in Tables 3–17. They report on the average results over five instances for each value of a ratio, the results obtained for each instance are available at the web page http://wpage.unina.it/sforza/test. The column *Ratio* reports on the value of r, column *Lagr iter* reports on the number of subgradient iterations, while column *Lagr time* reports on the time spent in computing the lower bound, which is shown in column *LB*. Columns $|S|$ and $|X(S)|$ report on the size of the core problem. The columns *B&C Nodes* and *B&C time* report the number of B&C nodes and the computation time, respectively, spent on yielding the upper bound, reported in the column *UB*. Finally column *%gap* reports on the duality gap, computed as $(UB - LB)/UB * 100$.

Tables 3–7 show results for the instances of test-bed A. For these instances, minimisation of the fixed costs is critical and so the parameter w in the core selection

Table 4 Computational results for test instances with 2000 facilities and 2000 clients (test bed A)

$\lvert I\rvert \times \lvert J\rvert$	Ratio	Lagr Iter.	Lagr time (s)	$\lvert X(S)\rvert$	$\lvert S\rvert$	B&C nodes	B&C Time (s)	%gap	Tot. Time (s)
2000 × 2000	1.1	300	145.1	–	–	–	–	0.65	145.1
2000 × 2000	1.5	300	140.3	–	–	–	–	0.97	140.3
2000 × 2000	2	250	139.5	–	–	–	–	0.72	139.5
2000 × 2000	3	200	118.8	132897	530	44	168.1	0.20	286.9
2000 × 2000	5	200	98.5	54714	325	22	54.5	0.19	153.1
2000 × 2000	10	200	69.5	18819	170	29	57.9	0.35	127.5

Table 5 Computational results for test instances with 1200 facilities and 3000 clients (test bed A)

Name	Ratio	Lagr Iter.	Lagr time (s)	$\lvert X(S)\rvert$	$\lvert S\rvert$	B&C nodes	B&C Time (s)	%gap	Tot. Time (s)
1200 × 3000	1.1	300	180.1	–	–	–	–	0.78	180.1
1200 × 3000	1.5	300	151.7	–	–	–	–	0.62	151.7
1200 × 3000	2	250	122.4	–	–	–	–	0.64	122.4
1200 × 3000	3	200	54.6	250559	320	1	121.7	0.45	176.4
1200 × 3000	5	200	47.4	55814	200	1	44.6	0.41	91.3
1200 × 3000	10	200	52.9	50609	100	4	127.8	0.90	180.7

Table 6 Computational results for test instances with 1000 facilities and 4000 clients (test bed A)

Name	Ratio	Lagr Iter.	Lagr time (s)	$\lvert X(S)\rvert$	$\lvert S\rvert$	B&C nodes	B&C Time (s)	%gap	Tot. Time (s)
1000 × 4000	1.1	300	180.1	–	–	–	–	0.82	180.1
1000 × 4000	1.5	300	151.9	–	–	–	–	0.54	151.9
1000 × 4000	2	250	140.1	547568	450	1	657.8	0.56	802.4
1000 × 4000	3	200	108.0	140754	300	7	55.1	0.51	161.1
1000 × 4000	5	200	72.4	64060	175	7	73.5	0.81	146.0
1000 × 4000	10	200	104.5	121371	90	1	280.8	0.80	385.2

procedure is chosen in the range $[0.75, 0.85]$. As we can see from the tables (column $|S|$), we have used a smaller value of w (near to 0.75) when r is small because in this case the dimension of the core problem may became too large, while we have chosen a greater value of w when r is equal to 5 or 10 because in this case the size of the core problem is very small.

For two instances ($P2000 - 2000 - 6$ and $P2000 - 2000 - 9$) with $r = 1.5$, the first stage of the heuristic holds a gap greater than 1%. Moreover, for these instances, the core problem generated by the second stage is too big ($|S| \geq 1000$) to run the *branch-and-cut* algorithm.

Table 7 Computational results for test instances with 800 facilities and 4400 clients (test bed A)

Name	Ratio	Lagr Iter.	Lagr time (s)	$\lvert X(S)\rvert$	$\lvert S\rvert$	B&C nodes	B&C Time (s)	%gap	Tot. Time (s)
800×4400	1.1	300	137.1	–	–	–	–	0.77	137.1
800×4400	1.5	300	128.7	–	–	–	–	0.80	128.7
800×4400	2	250	103.3	511542	380	1	295.5	0.57	398.9
800×4400	3	200	77.1	83329	260	4	54.4	0.76	131.5
800×4400	5	200	73.1	72460	150	10	123.6	0.84	196.7
800×4400	10	200	79.0	71230	75	40	183.7	0.50	262.7

Table 8 Computational results for test instances with 1000 facilities and 1000 clients (test bed B)

Name	Ratio	Lagr Iter.	Lagr time (s)	$\lvert X(S)\rvert$	$\lvert S\rvert$	B&C nodes	B&C Time (s)	%gap	Tot. Time (s)
1000×1000	1.1	300	46.6	–	–	–	–	0.65	46.6
1000×1000	1.5	300	28.7	301364	690	1	102.4	0.34	131.1
1000×1000	2	250	20.9	100366	520	1	25.2	0.43	46.2
1000×1000	3	200	11.5	10767	360	82	19.9	0.87	31.4
1000×1000	5	200	11.2	5315	220	76	9.8	0.79	20.9
1000×1000	10	200	9.6	5376	110	8	2.4	0.49	12.0

Table 9 Computational results for test instances with 2000 facilities and 2000 clients (test bed B)

Name	Ratio	Lagr Iter.	Lagr time (s)	$\lvert X(S)\rvert$	$\lvert S\rvert$	B&C nodes	B&C Time (s)	%gap	Tot. Time (s)
2000×2000	1.1	300	153.5	–	–	–	–	0.92	153.5
2000×2000	1.5	300	135.7	–	–	–	–	1.7	135.7
2000×2000	2	250	83.7	181867	1050	1	263.6	0.61	347.3
2000×2000	3	200	60.9	41242	700	1	52.0	0.81	112.9
2000×2000	5	200	43.7	17034	450	116	90.2	0.89	125.6
2000×2000	10	200	49.6	20574	300	1	41.9	0.81	92.5

All the other instances of test-bed A are solved by the proposed heuristic to 1%-optimality. The instances of 1000 facilities and 1000 clients are solved in less than 115 seconds, while instances of 2000 facilities and 2000 clients are solved in less than 8 minutes. The most difficult instances of test-bed A are those of 1000 facilities and 4000 clients with ratio r equal to 2. In particular, the heuristic requires about 20 minutes to solve to 1% optimality an instance ($P1000 - 4000 - 14$) with $r = 2$.

Computational results with test-bed B are reported in Tables 9–12. Here transportation costs play a more relevant role and so the parameter w in the core selection procedure is chosen in the range $[1.00, 1.10]$. As stated before for test-bed A, we

Table 10 Computational results for test instances with 1200 facilities and 3000 clients (test bed B)

Name	Ratio	Lagr Iter.	Lagr time (s)	$\|X(S)\|$	$\|S\|$	B&C nodes	B&C Time (s)	%gap	Tot. Time (s)
1200 × 3000	1.1	300	107.1	–	–	–	–	1.31	107.1
1200 × 3000	1.5	300	60.4	289562	850	1	202.6	0.45	251.6
1200 × 3000	2	250	49.4	99863	630	7	64.8	0.75	114.2
1200 × 3000	3	200	43.9	17646	425	35	32.7	0.85	76.6
1200 × 3000	5	200	37.4	16327	265	17	12.0	0.62	48.4
1200 × 3000	10	200	45.7	21452	135	1	9.1	0.67	54.8

Table 11 Computational results for test instances with 1000 facilities and 4000 clients (test bed B)

Name	Ratio	Lagr Iter.	Lagr time (s)	$\|X(S)\|$	$\|S\|$	B&C nodes	B&C Time (s)	%gap	Tot. Time (s)
1000 × 4000	1.1	300	131.9	–	–	–	–	2.58	131.9
1000 × 4000	1.5	300	95.8	166948	750	1	81.2	0.40	176.9
1000 × 4000	2	250	69.4	72526	520	3	44.5	0.65	113.9
1000 × 4000	3	200	69.4	67180	360	1	41.7	0.86	111.2
1000 × 4000	5	200	75.7	74135	220	3	30.0	0.75	105.7
1000 × 4000	10	200	95.7	92593	110	38	43.9	2.02	139.5

Table 12 Computational results for test instances with 800 facilities and 4400 clients (test bed B)

Name	Ratio	Lagr Iter.	Lagr time (s)	$\|X(S)\|$	$\|S\|$	B&C nodes	B&C Time (s)	%gap	Tot. Time (s)
800 × 4400	1.1	300	127.8	–	–	–	–	3.56	127.8
800 × 4400	1.5	300	103.7	89883	580	1	56.0	0.54	161.3
800 × 4400	2	250	80.8	51345	440	1	33.2	0.73	114.1
800 × 4400	3	200	73.7	51710	295	7	31.7	0.81	105.4
800 × 4400	5	200	73.8	52839	180	7	16.1	2.42	89.9
800 × 4400	10	200	73.6	42400	90	1	7.2	3.79	80.8

choose w nearer to the first extreme of the range when r is less than or equal to 3 to keep the core size small.

To our knowledge, instances of this type have never been tested in literature. Furthermore, it is worth noting that the cost structure of the problem affects the performances of the algorithm. It was not possible to always keep the gap less than 1% for test-bed B. In particular, this is true for the instances with ratio r equal to 1.1 and 1.5 and for the instances of 800 facilities and 4400 clients with r equal to 5 and 10.

Computational results with test-bed C are reported in Tables 13–17. The results obtained with these instances are not competitive if compared with other Lagrangean heuristics. This behaviour can be explained by bearing in mind that a good selection

Table 13 Computational results for test instances with 1000 facilities and 1000 clients (test bed C)

Name	Ratio	Lagr Iter.	Lagr time (s)	$\|X(S)\|$	$\|S\|$	B&C nodes	B&C Time (s)	%gap	Tot. Time (s)
1000 × 1000	1.1	300	50.4	–	–	–	56.4	6.03	106.8
1000 × 1000	1.5	300	44.9	32154	720	821	51.2	1.08	96.1
1000 × 1000	2	250	35.2	25109	550	1329	49.2	2.49	84.4
1000 × 1000	3	200	20.7	20206	390	227	25.9	3.30	46.6
1000 × 1000	5	200	16.5	17972	250	1	3.4	9.96	19.9
1000 × 1000	10	200	11.6	15457	130	1	2.8	24.3	14.4

Table 14 Computational results for test instances with 2000 facilities and 2000 clients (test bed C)

Name	Ratio	Lagr Iter.	Lagr time (s)	$\|X(S)\|$	$\|S\|$	B&C nodes	B&C Time (s)	%gap	Tot. Time (s)
2000 × 2000	1.1	300	150.5	–	–	–	–	8.30	150.5
2000 × 2000	1.5	300	111.3	–	–	–	–	15.89	111.3
2000 × 2000	2	250	79.4	140737	1100	3512	186.5	1.11	265.9
2000 × 2000	3	200	63.7	66655	750	1205	92.4	2.04	166.1
2000 × 2000	5	200	56.7	79565	500	1	4.4	11.87	61.1
2000 × 2000	10	200	49.1	77145	350	1	4.0	26.20	53.1

Table 15 Computational results for test instances with 1200 facilities and 3000 clients (test bed C)

Name	Ratio	Lagr Iter.	Lagr time (s)	$\|X(S)\|$	$\|S\|$	B&C nodes	B&C Time (s)	%gap	Tot. Time (s)
1200 × 3000	1.1	300	99.2	–	–	–	–	17.00	99.2
1200 × 3000	1.5	300	90.4	293710	900	18	119.7	1.01	210.1
1200 × 3000	2	250	63.7	181236	700	1	236.8	2.94	300.5
1200 × 3000	3	200	46.4	163926	450	1	93.5	24.05	139.9
1200 × 3000	5	200	42.9	145561	300	1	41.2	35.70	84.1
1200 × 3000	10	200	33.0	105836	150	1	13.7	63.7	46.7

of the core problem is crucial to the success of the heuristic. Moreover, the core problem selection is based on the choice of the set of facilities $|S|$. If the role played by fixed costs is not relevant, the choice of set S containing the facilities opened in an optimal solution becomes much more difficult.

Moreover, our computational experience shows that it is also useful to take the cost structure into account in the settings of Lagrangean heuristics. Particularly, for test-bed A, a good choice of the value of the reduction rate of φ is 1.01. For the instances of test-bed B and C, the initial upper bound is worse and this makes a faster reduction of φ necessary for better tuning of subgradient optimization. In this case a good choice is to set the reduction rate of φ to 1.025.

Table 16 Computational results for test instances with 1000 facilities and 4000 clients (test bed C)

Name	Ratio	Lagr Iter.	Lagr time (s)	$\|X(S)\|$	$\|S\|$	B&C nodes	B&C Time (s)	%gap	Tot. Time (s)
1000×4000	1.1	300	130.6	–	–	–	–	9.19	130.6
1000×4000	1.5	300	121.5	282998	775	1	105.7	1.86	227.2
1000×4000	2	250	86.4	224392	550	1	126.9	2.82	213.3
1000×4000	3	200	77.9	188635	400	1	44.5	31.87	122.4
1000×4000	5	200	73.8	156753	250	1	20.8	50.95	84.6
1000×4000	10	200	87.9	126759	150	1	14.6	50.85	102.5

Table 17 Computational results for test instances with 800 facilities and 4400 clients (test bed C)

Name	Ratio	Lagr Iter.	Lagr time (s)	$\|X(S)\|$	$\|S\|$	B&C nodes	B&C Time (s)	%gap	Tot. Time (s)
800×4400	1.1	300	138.0	–	–	–	–	29.22	138.0
800×4400	1.5	300	116.2	164230	600	1	45.5	1.21	161.7
800×4400	2	250	88.3	147269	460	1	20.0	4.95	108.3
800×4400	3	200	73.6	138689	315	1	18.4	34.27	92.0
800×4400	5	200	77.3	125128	200	1	15.1	35.12	92.4
800×4400	10	200	76.5	130864	100	1	17.6	46.03	94.1

Table 18 Computational results of a standard Lagrangean heuristic

Name	Ratio	Lagr Iter.	LB K^{th} iterations	LB	UB	Gap(%)	# of upper bound computation	Total time of upper bound computation	Total time (s)
P1000-1000-11	2	443	334562.3	347661.2	350976.4	0.94	21	1243.2	1283.4
P1000-1000-12	2	398	302705.9	309158.5	311556.1	0.77	19	1008.9	1055.6
P1000-1000-16	3	875	93502.4	95752.4	96243.7	0.51	25	658.1	679.3
P1000-1000-17	3	1000	90989.5	93893.6	94825.8	0.98	27	906.3	1062.1.8
P1000-1000-21	5	1000	47885.1	49700.6	50255.4	1.10	31	336.9	443.1
P1000-1000-22	5	1000	49415.3	51082.0	51791.7	1.37	32	365.6	445.2
P1000-1000-26	10	1000	25143.4	27410.6	27980.0	2.03	29	150.8	199.4
P1000-1000-27	10	1000	22764.1	26537.3	26941.5	1.50	35	175.4	235.7

For sake of clarify, Figs. 2, 3, 4 and 5 show the diagrams related to the results obtained on the set of instances of test-beds A and B. They show the value of the gap (Figs. 2 and 3) and the CPU time required by the heuristic (Figs. 4 and 5), parameterized in relation to the value of the ratio r for each group of instances.

Table 19 Computational results of a Lagrangean heuristic with only one upper bound computation

Name	Ratio	Lagr Iter.	LB K^{th} iterations	LB	Initial UB	Gap(%)	# of upper bound computation	Total time of upper bound computation	Total time (s)
P1000-1000-11	2	443	334562.3	347661.2	350976.4	0.94	1	55.2	103.5
P1000-1000-12	2	398	302705.9	309158.5	311556.1	0.77	1	56.2	109.4
P1000-1000-16	3	843	92546.4	95663.5	98539.5	2.92	1	28.6	161.7
P1000-1000-17	3	769	89371.3	93809.2	95475.1	1.74	1	27.9	174.6
P1000-1000-21	5	746	46163.6	49658.4	51207.5	3.02	1	11.4	97.8
P1000-1000-22	5	705	49155.4	50914.5	53772.8	5.31	1	12.0	88.4
P1000-1000-26	10	912	25108.8	27305.1	32279.0	15.4	1	4.9	55.0
P1000-1000-27	10	1000	22296.1	26417.4	36784.5	28.2	1	5.5	54.4

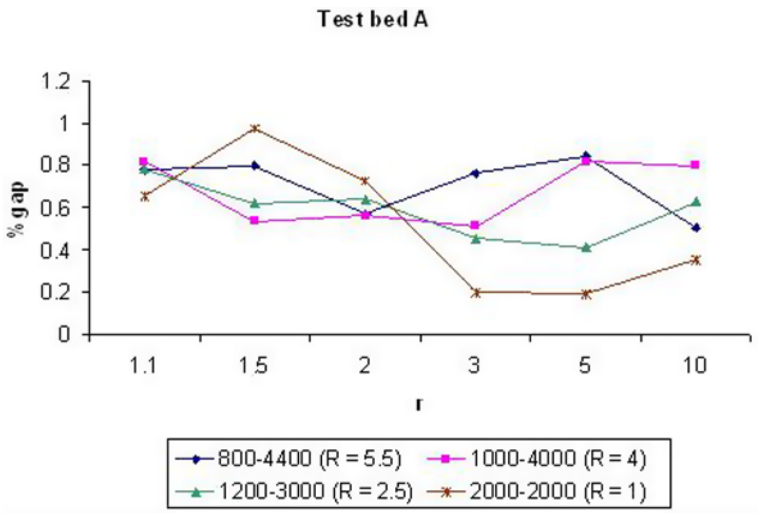

Fig. 2 Test-bed A, $\%gap$ vs. r

Figure 2 shows that the proposed heuristic is very effective on test-bed A for all values of r. In fact the value of the gap is always less than 1%.

For test-bed B (Fig. 3) the behaviour of the heuristic depends on the values of parameters R and r. The worst combination is obtained for $R = 5.5$ and $r = 1.1$ or $r = 10$. In these cases the gap is within the range 2.5–3.8%. For $R = 2.5$ and $R = 1$ the behaviour of the heuristic is much better.

On the other hand, if we consider the CPU time (Figs. 4 and 5), our heuristic is faster in solving instances of test-bed B than test-bed A. The CPU time for the test-bed A (Fig. 4) is in the range 100–400 seconds, except for the combination $r = 2$ and

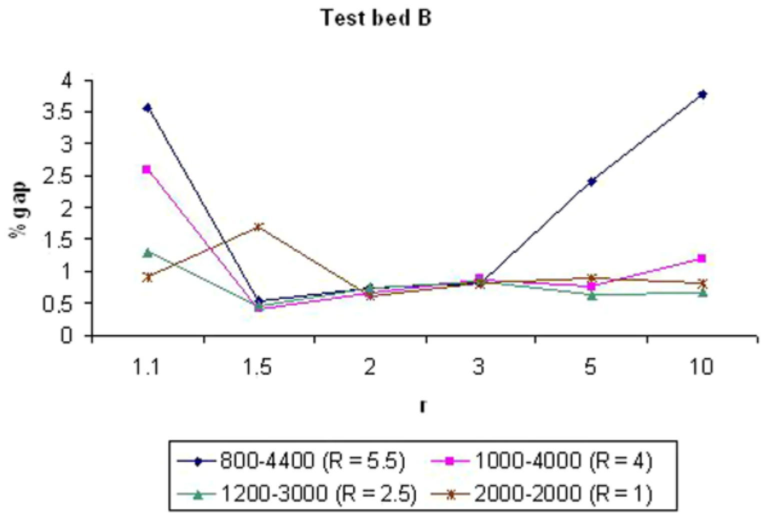

Fig. 3 Test-bed B, *%gap* vs. *r*

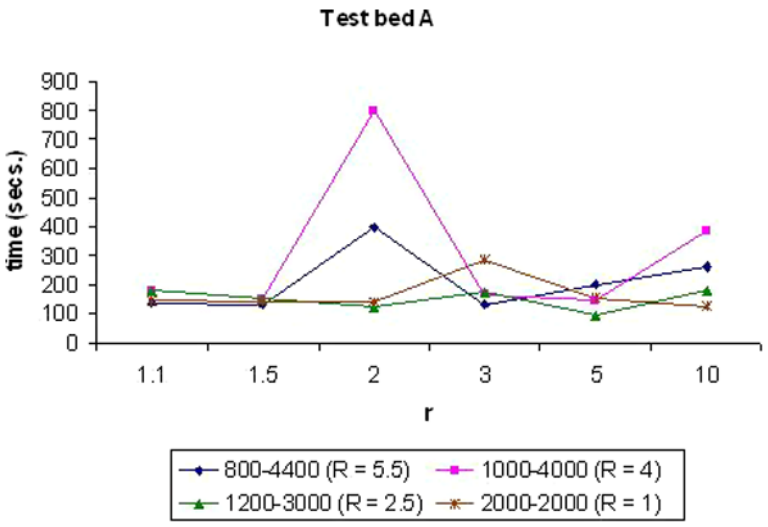

Fig. 4 Test-bed A, *CPU Time* (s) vs. *r*

$R = 4$ that requires up to 800 seconds of CPU time. For test-bed B, the CPU time required by the heuristic is at most 350 seconds.

Finally, to validate the effectiveness of the proposed heuristic, we compared it with the standard Lagrangean heuristic (Tables 18 and 19). In Table 18 we report results by a Lagrangean heuristic on a subset of the test instances. Column *LB* K^{th} *iterations* reports on the value of the lower bound computed after k iterations, k being the number of Lagrangean iterations performed by our algorithm as in Table 3.

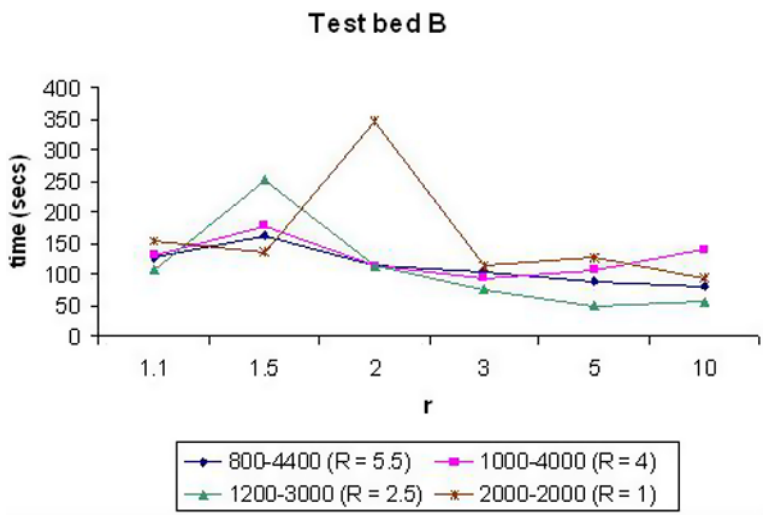

Fig. 5 Test-bed B, *CPU Time* (s) vs. r

In the implementation of the Lagrangean heuristic, the parameter φ has been halved after 65 subgradient iterations without increasing the lower bound. This choice is the nearest to our step size rule ($1.01^{65} \approx 2$) and in any case it behaves better than halving φ after 20–30 iterations. An upper bound is computed before starting the subgradient optimization, as described in Sect. 3.2, and then in every iteration where the lower bound improves. To improve the upper bound computation, the total capacity constraint (10) is added to the Lagrangean relaxation (Cornuejols et al. 1991).

The Lagrangean heuristic is stopped when the gap between the upper and lower bound is either less than 1%, or after 1000 iterations. From Table 18 we can see that our approach outperforms the standard Lagrangean heuristic both in terms of computational time and quality of the upper bound.

6 Conclusions

In this paper we have proposed an effective heuristic for the solution of the Capacitated Facility Location Problem (CFLP). It is based on a Lagrangean heuristic, made very efficient by subgradient deflection and by a new step size rule and on the selection of a core problem defined according to Lagrangean reduced costs and solved by a Branch-and-Cut algorithm.

The heuristic has been tested on a wide set of instances. On two of the three test beds considered, it is able to solve very large scale instances in very short computation times, thus allowing a significant enhancement in the state of the art of the problem.

The heuristic is not efficient if applied to instances (test bed C) characterized by fixed and *scaled* transportation costs of the same order of magnitude.

References

Aardal, K.: On the solution of one and two-level capacitated facility location problems by the cutting plane approach. Ph.D. Thesis, CORE, Louvain-la-Neuve (1992)

Aardal, K.: Capacitated facility location: separation algorithms and computational experience. Math. Program. **81**, 149–175 (1998)

Aardal, K., Pochet, Y., Wolsey, L.A.: Capacitated facility location: valid inequalities and facets. Math. Oper. Res. **20**, 562–582 (1995)

Anbil, R., Barahona, F.: The volume algorithm: producing primal solutions with a subgradient method. Math. Program. **87**(3), 385–399 (2000)

Balas, E., Zemel, E.: An algorithm for the zero-one knapsack problems. Oper. Res. **28**, 1130–1154 (1980)

Barahona, F., Chudak, F.A.: Near-optimal solution to large scale facility location problems. Internal Report, IBM Research Division, T.J. Watson Research Center, RC 21606 (1999)

Beasley, J.E.: An algorithm for solving large capacitated warehouse location problems. Eur. J. Oper. Res. **33**, 314–325 (1988)

Beasley, J.E.: Lagrangean heuristics for location problems. Eur. J. Oper. Res. **33**, 383–399 (1993)

Camerini, P.M., Fratta, L., Maffioli, F.: On improving relaxation methods by modified gradient techniques. Math. Program. Study **3**, 26–34 (1975)

Caprara, A., Fischetti, M., Toth, P., Vigo, D., Guida, P.L.: Algorithms for railway crew management. Math. Program. **79**, 125–141 (1997)

Chudak, F.A., Williamson, D.P.: Improved approximation algorithms for capacitated facility location problems. In: Proceedings of the 7th International IPCO Conference, June 1999 (1999)

Cornuejols, G., Sridharan, R., Thizy, J.M.: A comparison of heuristics and relaxations for the capacitated plant location problem. Eur. J. Oper. Res. **50**, 280–297 (1991)

Korupolu, M.R., Plaxton, C.G., Rajaraman, R.: Analysis of a local search heuristic for facility location problems. DIMACS Technical Report, 98–30 (1998)