Available online at www.sciencedirect.com

SCIENCE DIRECT®

Computers & Operations Research 32 (2005) 3075–3091

computers & operations research

www.elsevier.com/locate/cor

Solving two location models with few facilities by using a hybrid heuristic: a real health resources case

Joaquín A. Pacheco*, Silvia Casado

Department of Applied Economics, University of Burgos, Plaza Infanta Elena s/n Burgos 09001, Spain

Available online 10 June 2004

Abstract

We propose a metaheuristic procedure based on the scatter search approach for solving two location problems with few facilities ($p \leqslant 10$). The first problem is the well-known p-center problem. The second one is the *maximum set covering problem* (MSCP). This scatter search algorithm incorporates procedures based on different strategies, such as local search, GRASP, and path relinking. We first designed the algorithm for the p-center problem, and then modified it for the MSCP. The aim is to solve problems with real data provided by the Health Authorities of Burgos (northern Spain). Because the authorities have a limited budget, less than 10 facilities can considered in both problems. A series of computational experiments were also performed. The proposed algorithm gave similar results to the recently reported methods for the p-center problem but much faster. The quality of the solutions is also very good for the MSCP (less than 1% deviation from the lower bound). We show its application to the location of health resources with real data in the province of Burgos.
© 2004 Elsevier Ltd. All rights reserved.

Keywords: Location; p-center; GRASP; Path relinking; Local search; Scatter search

1. Introduction

The p-center problem is a well-known NP-hard discrete location problem [1]. It consists in locating p facilities, which are assigned to clients (each facility can serve different numbers of clients) in order to minimize the maximum distance between them and their closest facility.

Let $U = \{u_1, u_2, \ldots, u_m\}$ be a set of users and let $V = \{v_1, v_2, \ldots, v_n\}$ be a set of potential locations for facilities. Let us consider d_{ij} to be the distance between each user-location pair (u_i, v_j). The problem

* Corresponding author. Tel.: +34-947-259021; fax: +34-947-258956.
E-mail address: jpacheco@ubu.es (J.A. Pacheco).

is to find a subset $X \subseteq V$ of size p such that the following expression is minimized

$$\max_{i=1..m} \left\{ \min_{v_j \in X} d_{ij} \right\}.$$

An integer programming formulation of the problem is the following:

$$\begin{aligned}
&\text{Minimize} \quad z \\
&\text{subject to} \quad \sum_{j=1..n} x_{ij} = 1, \quad i = 1..m; && (1)\\
&x_{ij} \leqslant y_j, \quad i = 1..m; \; j = 1..n; && (2)\\
&\sum_{j=1..n} y_j = p; && (3)\\
&\sum_{j=1..n} d_{ij} x_{ij} \leqslant z, \quad i = 1..m; && (4)\\
&x_{ij}, y_j \in \{0, 1\}, \quad i = 1..m; \; j = 1..n; && (5)
\end{aligned}$$

where $y_j = 1$ means that a facility is located at v_j (and 0 otherwise); $x_{ij} = 1$ if user u_i is assigned to facility v_j (and 0 otherwise).

This model is used, for example, to locate fire stations, police stations or ambulances such that the distance from the facilities to their farthest allocated client is the minimum. However, in some instances, such as health services, rather than minimizing the distance of users to the facilities, the key issue is to maximize the potential number of users able to use the service within a reasonable time ('threshold time' or 'critical time'). For example, let us assume that in a particular region the National Health Service is interested in opening some centers for diabetic patients. If a patient suffers an insulin shock he/she has to be attended to in less than 20 min, otherwise damage may be permanent. In this case, it is important for most people with diabetes to be no further than 20 min from their closest center.

This involves designing a new model that can be expressed as a *maximum set covering problem* or MSCP (see [2]). Let us assume that $U = \{u_1, u_2, \ldots, u_m\}$ is now a set of locations within a certain region or area, each one with a given population q_i, $i = 1..m$, of potential users of a given service or facility (for example a center for diabetic patients). V is the location set where p facilities can be open; the matrix d_{ij} is a time matrix and the maximum time to attend to users should not exceed a period denoted by $t\_critical$. The objective is

$$\begin{aligned}
&\text{Minimize} \quad \sum_i q_i \cdot r_i \\
&\text{subject to:} \quad (3) \\
&\sum_{j/d_{ij} \leqslant t\_\text{critical}} y_j + r_i \geqslant 1, \quad i = 1..m; \\
&y_j \in \{0, 1\}, \quad j = 1..n; \\
&r_i \in \{0, 1\}, \quad i = 1..m; && (8)
\end{aligned}$$

where r_i has a value of 1 if the facility assigned to u_i is more than *t_critical* and 0 otherwise. Note that (8) could be replaced by

$$r_i \in (0, 1), \quad i = 1..m. \tag{8'}$$

Although the MSCP is a maximization problem, the objective function is minimized because it consists in minimizing the number of patients that are not covered.

This work is part of a project aimed at developing a system to find the best locations in which to place special health resources (geriatric, diabetic care units, etc.) in the provinces of Castilla and León, Spain. Because the authorities have a limited budget, less than 10 facilities are considered. These economic conditions prevent the opening of more than 10 units in each province. Two key factors are taken into account, i.e., the economic factor (determined by the number of units to be opened that has to be less than 10), and the social factor, determined by several objective functions according to the kind of unit or service. For example, in an emergency unit, the objective is to minimize the maximum distance a user has to travel to his/her closest center. On the other hand, in a diabetic care unit, as we pointed out earlier, the objective is to maximize the number of patients who can reach a unit within a given time.

Therefore, for each unit we are dealing with a 2-criteria problem: economic (number of units) and social factors. The goal is to develop a system able to provide solutions as close as possible to the efficiency curve. In this way, decision-makers can choose the most suitable ones from among a set. On the other hand, the number of units of each type to be added has to be small: never higher than 10 in each area. Thus, we opted for a system able to solve single-objective problems corresponding to different numbers of units to be added (from $p = 1$ to 10). For this reason, we developed two different algorithms for the two location models—p-center and 'maximum set cover'—which correspond to the two social objectives under consideration. Our aim is to make these algorithms especially efficient for low p values.

In a recent work, Mladenović et al. [3] adapted several of the more classic heuristics for the p-median problem to the p-center problem (see [4,5]), as well as several types of neighboring moves and structures (see [6,7]). They also suggested two algorithms based on the *variable neighborhood* and *tabu search* strategies. There exists at least one recent reference to the MSCP as formulated in the present work ([8], 1998).

In this paper, we propose a scatter search (SS) procedure [9–11] for the p-center problem. This algorithm incorporates different elements based on GRASP strategies [12–14], local search, and path relinking [15,16]. We test the efficiency of this algorithm for low values of p and compare it to other recent strategies. More specifically, we use the instances of the well-known library OR-Lib [17]. Similarly, the SS algorithm and its elements have been adapted to create another algorithm for the MSCP. In order to evaluate the efficiency of this new strategy—also for low values of p—several tests were done with different adaptations of these instances. The results obtained with our SS strategy were compared to the lower bounds obtained by executing CPLEX MIP 8.0. In addition, we analyzed a series of instances based on real data using both models. These data refer to the area of Burgos (northern Spain), and to estimates regarding the care of diabetics.

The paper is organized as follows: in Section 2 we describe the classic heuristics for the p-center problem that we use later, as well as more recent metaheuristics. Section 3 describes the scatter search algorithm proposed as well as its elements. Section 4 shows how the algorithm can be adapted for the MSCP. In Section 5 we show the computational results from both models with hypothetical instances. Section 6 shows results with real data and the solutions. Finally, Section 7 is devoted to the conclusions.

2. Main Heuristics for the p-center problem

Mladenović et al. [3] adapt three classical heuristics of the well-known heuristics of [18] for the p-median problem to the p-center problem. These three heuristics involve a constructive method (*Greedy*) and two improvement procedures (*alternate* and *interchange*). We describe them briefly by using a problem defined by m, n, d, and p (from now on, and for reasons of simplicity, U and V will be directly identified with the indices, i.e., $U = \{1, 2, \ldots, m\}$ and $V = \{1, 2, \ldots, n\}$), where m is the number of users, n is the number of potential locations for facilities, d is the matrix of distances (d_{ij}: the distance between each user and location) and p is the number of facilities that must be located. In addition, we denote X as the partial or complete solution at each moment (partial solution if there are less than p elements in X, and complete solution otherwise), i.e., locations (indexes) where the facilities are placed, and f as the value of the objective function corresponding to X. The heuristics can be described as follows:

```
Greedy procedure
Let $X = \emptyset$
While $|X| < p$ do
  • Determine the location $j^*$ that will provide the lowest value of $f$ if a facility is added.
  • Let $X = X \cup \{j^*\}$.
```

```
Alternate procedure
Repeat
  • For each facility $j$ of $X$, determine the subset of points $U^j \subset U$ that have $j$ as
    the closest facility.
  • Solve the 1-center problem for each subset $U^j$.
  • Let $X'$ be the set of solutions of these $p$ problems, and $f'$ its value.
  • If $f' < f$ let $X = X'$ and $f = f'$
until no more changes take place in $X$.
```

```
Interchange procedure
Repeat
  • For each $j \in V–X$ and $k \in X$ calculate the value of the objective function $v_{jk}$ if
    the facility moved from $k$ to $j$.
  • Calculate $v_{j^*k^*} = min\{v_{jk}/j \in V–X \text{ and } k \in X\}$.
  • If $v_{j^*k^*} < f$ then, let $X = X - \{k^*\}$, $X = X \cup \{j^*\}$ and $f = v_{j^*k^*}$
until no further improvement,
```

where v_{jk} is the value of the objective function if the facility moved from k to j. In these algorithms, as occurs in the one we describe below, there is a fundamental auxiliary variable, $c_{\mathrm{I}}(i)$, which indicates the location of the closest facility in the current solution for each user i, $i = 1..m$. Mladenović et al. [3] also make use in this last procedure of the following property: "let i^* be the user who defines the value of f; that is, $d_{i^*c_{\mathrm{I}}(i^*)} = f$, ('critical user'), then any move defined by $j \in V–X$ and $k \in X$, able to reduce the value of the objective function ($v_{jk} < f$), has to verify that $d_{i^*j} < d_{i^*c_I(i^*)}$". In this way, the search is reduced at each iteration and the procedure accelerates.

Mladenović et al. [3] also define two heuristics based on tabu search (TS) and variable neighborhood search (VNS).

In the tabu search procedure two tabu lists are used that correspond to the elements have just entered the solution (L_{in}) and those that have just left (L_{out}). Notice that the concept of critical client is used when a good path is found (improvement of the best solution). The TS procedure is briefly described below.

```
Tabu search
Read initial solution $X$, $f$, and let $X\_best = X$, $f\_best = f$, improve = TRUE
Initialize tabu lists, $L_{in} = \emptyset$, $L_{out} = \emptyset$
Repeat
  • Calculate the critical client $i^*$ corresponding to $X$.
  • Let $J = \{j \in V \backslash X \backslash L_{out} / d_{i^*j} < f\}$.
  • If ($J = \emptyset$) or (no improve) then $J = \{j \in V \backslash X \backslash L_{out}\}$.
  • $\forall k \in X \backslash L_{in}$ and $j \in J$ calculate $v_{jk}$.
  • Calculate $v_{j^*k^*} = min\{v_{jk} / j \in V \backslash X$ and $k \in X\}$.
  • Let $X = X - \{k^*\}$, $X = X \cup \{j^*\}$ and $f = v_{j^*k^*}$.
  • If $f < f\_best$, then let $f\_best = f$, $X\_best = X$, $improve = TRUE$;
    else $improve = FALSE$.
  • Update tabu lists: $L_{in}$ and $L_{out}$
until a stopping condition is reached.
```

VNS is a recent metaheuristic for solving optimization problems, whose basic idea is systematic change of neighborhood within a local search [19,20]. Two recent tutorials are Hansen and Mladenovic [21,22]. More information is available at http:vnsheuristic.ull.es.

The VNS procedure is as follows:

```
Variable neighbor search procedure
Read initial solution $X\_best$, $f\_best$
Repeat
  $k = 1$
  Repeat
  •  $X = X\_best$, $f = f\_best$
  •  $\forall j = 1..k$
       Calculate the critical client $i^*$ corresponding to $X$
       Take $L_{in} \in V \backslash X / d(i^*, L_{in}) < f$ randomly
       Take $L_{out} \in X$ randomly
       Let $X = X - \{L_{out}\}$, $X = X \cup \{L_{in}\}$ and update $f$
  •  Apply the procedure interchange to $X$ and $f$
  •  If $f < f\_best$ then $X\_best = X$, $f\_best = f$ and $k = 1$;
     otherwise $k = k + 1$
  until $k = kmax$
Until a stopping condition is reached.
```

In both algorithms the stopping condition is a series of iterations without improvement in f_best or a maximum computation time.

3. Scatter search algorithm

The solution approach we have developed is an adaptation of the scatter search methodology. SS is an instance of the so-called evolutionary methods, with the distinction (compared to other evolutionary methods) that its mechanism for searching is not based solely on randomization. More about the origins and multiple applications of scatter search can be found in [9–11,16].

SS is characterized by the use of a *Reference Set* (*RefSet*) of solutions. At each step reference solutions are combined to generate new solutions and update the current Reference Set according to some systematic rules.

For this problem, we have developed a version of scatter search that uses a static update of the Reference Set. The following pseudocode gives a general description of our scatter search implementation:

```
Static scatter search procedure
  1. Generate an initial set P of PSize solutions with a Diversification-Generation Method
  2. Improve these solutions by an Improvement Method
  3. With these solutions build an initial RefSet
  4. Repeat
     4.1. Obtain all subsets of pairs of solutions from RefSet
     4.2. Apply a Combination Method to these subsets and obtain new solutions
     4.3. Improve these solutions by the Improvement Method
     4.4. Update RefSet considering these news solutions
     until RefSet is stable (i.e. no new solutions have been included)
  5. If max_iter iterations (steps 1–4) elapse without improvement stop
     else return step 1
```

The size of P is denoted by *PSize* (step 1).

In order to build the initial Reference Set (step 3) we start by selecting the best b_1 solutions in P according to the objective function value. Then, in order to add the remaining b_2 elements to *RefSet*, we measure the “diversity” of a candidate solution x in relation to those elements already in *RefSet*:

$$Difmin(x, RefSet) = \min\{dif(x, x')/x' \in RefSet\};$$

where $dif(x, x') = |x - x'|$. We then select the candidate solution x that maximizes *Difmin*(x,*RefSet*).

In addition, $b = b_1 + b_2$ denotes the size of *RefSet*.

The updating of *RefSet* (step 4.4.) is carried out by taking into account only the quality of the solutions. In other words, a new trial (x) solution that improves the objective function of the worst reference solution (x^b) is included in *RefSet*. Since the Reference Set is ordered according to solution quality, the worst reference solution is always the last one (denoted by x^b). As improvement method we use the Alternate and Interchange procedures in this order, which has been chosen after performing different tests. We now provide descriptions of the diversification and combination methods.

3.1. Diversification method

Our diversification method is based on GRASP constructions. GRASP, or Greedy Randomized Adaptive Search Procedure, is a heuristic that usually applies a randomized greedy constructive heuristic. Most GRASP implementations also include a local search that is used to improve upon the solutions generated with the randomized greedy function. GRASP was originally proposed in the context of a set covering problem [12]. Details of the methodology and a survey of applications can be found in [13,14].

In this case, the greedy function Δ_j is, in each iteration, the value of the objective function if location j was added to the solution. The diversification method consists of the following steps.

```
Random-greedy procedure
Let $X = \emptyset$
While $|X| < p$ do
  • Calculate $\Delta_j \forall j \in V \backslash X$
  • Calculate $\Delta\_max = max\{\Delta_j / j \in V \backslash L\}$ and $\Delta\_min = min\{\Delta_j / j \in V \backslash L\}$
  • Define $L = \{j \in V \backslash L / \Delta_j \leqslant \alpha \Delta\_min + (1-\alpha) \cdot \Delta\_max\}$
  • Choose $j^* \in L$ randomly
  • Let $X = X \cup \{j^*\}$.
```

The α parameter ($0 \leqslant \alpha \leqslant 1$) controls the level of randomization for the greedy selections. Randomization decreases as the value of α increases. This controlled randomization results in a sampling procedure where the best solution found (referring to the constructive algorithm) is typically better than the one found by setting $\alpha = 1$. A judicious selection of the value of α provides a balance between diversification and solution quality.

The first time that the diversification method is employed (step 1), there is no history associated with the number of times each location j has belonged to solutions in *RefSet*. However, this information is valuable when the method is applied to re-start the process. The information is stored in the following array:

$$freq(j) = \text{number of times each location } j \in V \text{ has belonged to solutions in } RefSet.$$

The information accumulated in $freq(j)$ is used to modify the Δ_j values in the application of diversification method. The modified evaluation is

$$\Delta'_j = \Delta_j - \beta \Delta\_max \frac{freq(j)}{freq_{max}},$$

where $freq_{\max} = max\{freq(j) : \forall i\}$. The modified Δ'_j values are used to calculate Δ'_max and Δ'_min and execute the diversification method. When $\beta = 0$, this modified diversification method is the original diversification method. Large values of β encourage the selection of locations that have not frequently belonged to solutions in *RefSet*. The use of frequency information within a diversification method is inspired by [23,24].

3.2. Combination method

New solutions are generated from combining pairs of reference solutions (step 4.2). The number of solutions generated from each combination depends on the relative quality of the solutions being

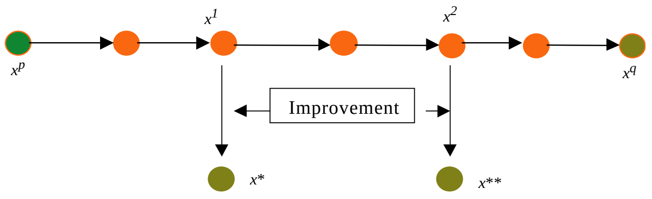

Fig. 1. Generation of new solutions by using path relinking.

combined. Let x^p and x^q be two reference solutions being combined, where $p<q$. Assume, as before, that the Reference Set is ordered in a way that x^1 is the best solution and x^b is the worst. Then, the number of solutions generated from each combination is:

$$3 \text{ if } p \leqslant b_1 \text{ and } q \leqslant b_1$$

$$2 \text{ if } p = b_1 \text{ and } q > b_1$$

$$1 \text{ if } p > b_1 \text{ and } q > b_1.$$

Each pair of reference solutions is combined to generate new solutions. The combination method is based on a strategy called path relinking, which was originally proposed in the context of tabu search [15] and has also been used in scatter search [16] as well as GRASP [25,26]. The underlying idea is that in the path between two good solutions other solutions of similar quality (and perhaps better) may be found. It consists in building a path to join the two solutions: an initiating solution and a guiding solution. This path contains a number of intermediate points (solutions). In our implementation, the intermediate solutions are chosen so as to be as equidistant as possible from each other and the extremes. The improvement method is then applied to these intermediate solutions. Fig. 1 depicts this idea.

From every pair of solutions of the Reference Set, in figure x^p and x^q, a path to join them is built. Solutions in these intermediate preselected positions within the path, in figure x^1 and x^2, are improved. In this way new solutions are generated, denoted as x^* and x^{**} in Fig. 1.

The path that joins x^p and x^q is built as follows: initially we set $x=x^p$. In the next iterations we add to x an element of $x^q \backslash x$ and then we remove from x an element of $x \backslash x^q$. In this way, the intermediate solution x in each iteration moves closer to x^q. At each iteration the best possible swap is selected, that is, the best pair of elements.

4. Adaptations for the MSCP

The SS algorithm for the MSCP shares the same basic structure and design as the SS algorithm described by the p-center. The same procedures were applied with small modifications. These modifications only refer to the way the objective function is calculated.

Note that in all the procedures we can call upon (alternate, interchange, random-greedy or path relinking) it is better to use the auxiliary variable $c_{\mathrm{I}}(i)$, which shows for each user i, $i=1..m$ the location of the closest facility in the current solution, whether this is partial or complete. In this way, the objective function f in the p-center problem is calculated as follows:

$$f = \max\{d_{ic_{\mathrm{I}}(i)} / i = 1..m\}$$

and in the case of MSCP:

$$f = \sum_{i/d_{ic_{\mathrm{I}}(i)} > t\_\mathrm{critical}} .$$

For example, in the Random–Greedy process, Δ_j values, i.e., the value of f if j was added to X, can be easily calculated with the following code for p-center problems:

```
• $\Delta_j = 0$
• For $i = 1..m$ let:
    $dist = \min\{d_{ij}, d_{ic_{\mathrm{I}}(i)}\}$
    $\Delta_j = \max\{\Delta_j, dist\}$;  (*)
```

For the MSCP we would only have to replace instruction (*) by

$$\text{If } dist > t\_\text{critical then } \Delta_j = \Delta_j + q_i.$$

In short, the adaptation only requires changing very few lines of code.

5. Computational results

In order to assess the efficiency of the strategy suggested for both problems, a series of tests were carried out which are described in Sections 5 and 6. These tests were done with a Pentium 3, 600 MHz. The implementation of the algorithms (the ones proposed in Mladenović et al. [3] and our scatter search) was done in Pascal, with the Delphi 5.0 compiler. In this section, we use artificial instances, whereas Section 6 includes results with real instances.

5.1. Tests for the p-center problem

We performed a final preliminary experiment to determine the best values for the parameters in the scatter search procedure. The tables for this experiment are not presented at this point. The parameter combination that yielded the best results was $Psize = 12$, $b_1 = b_2 = 3$; $\alpha = \beta = 0,8$ and $max\_iter = 5$.

Next, we compare the results of our scatter search algorithm (SS) with those obtained for the algorithms suggested by Mladenović et al. [3]: VNS, TS-II (tabu search with two tabu lists, L_{in} and L_{out}). To this end, we have used instances from the OR-Lib for the p-median problem corresponding to values $p \leqslant 10$. The reason is that this work is aimed at solving problems with real data. This real data comes from a province named Burgos located in the north of Spain. The health authorities in this area cannot consider more than 10 new units due to budget restrictions. In these examples $U = V$, which means that the facility locations match users. The results are given in Table 1.

Columns (T.B.VNS) and (T.B.TS) show the computational time (in seconds) until the best solution is found by VNS and TS, respectively. The total computational time for these strategies was limited to 400 s. The last column (T.Total) shows the total computational time (in seconds) used by SS. The second to last (T.Best) column shows the computational time (in seconds) until the best solution is found by SS. For each instance the strategy with the best result is shown in bold, or the fastest when there is tie.

In Table 1 we can observe that scatter search in all cases finds the best solution, and matches the previous best known solution reported in Mladenović et al. [3]. Scatter search is the best strategy in

Table 1
Results for the instances of OR-Lib, with $p \leqslant 10$ (p-center problem)

OR Lib file	n	p	VNS	T.B.VNS	TS-II	T.B.TS	SS	T.Best	T.Total
Pmed1	100	5	127	0.48	**127**	**0.08**	127	0.11	6.20
Pmed2	100	10	98	197.40	98	1.48	**98**	**1.26**	7.96
Pmed3	100	10	93	8.72	94	27.28	**93**	**0.49**	7.91
Pmed6	200	5	84	2.92	84	0.12	**84**	**0.06**	17.91
Pmed7	200	10	65	5.16	64	45.80	**64**	**16.31**	37.73
Pmed11	300	5	59	2.36	**59**	**0.76**	59	7.96	41.85
Pmed12	300	10	51	114.4	51	28.28	**51**	**4.29**	46.03
Pmed16	400	5	**47**	**0.72**	47	0.76	47	2.42	59.21
Pmed17	400	10	39	203.08	40	6.44	**39**	**10.49**	81.95
Pmed21	500	5	**40**	**1.56**	40	34.68	40	2.91	84.42
Pmed22	500	10	39	94.76	39	9.24	**38**	**81.12**	192.95
Pmed26	600	5	**38**	**3.88**	38	6.04	38	4.50	131.11
Pmed27	600	10	33	102.48	33	81.32	**32**	**78.60**	256.83
Pmed31	700	5	30	63.32	**30**	**1.80**	30	2.25	158.51
Pmed32	700	10	29	67.04	29	374.2	**29**	**21.42**	224.21
Pmed35	800	5	**30**	**3.0**	30	387.6	30	4.88	212.50
Pmed36	800	10	28	3.08	28	30.84	**27**	**26.47**	302.74
Pmed38	900	5	**29**	**7.40**	29	13.24	29	11.32	284.24
Pmed39	900	10	23	142.64	24	38.68	**23**	**35.54**	352.51

Table 2
Results for north of Spain instances with $p \leqslant 10$ (p-center problem)

	m	N	P	VNS	TB.VNS	TS	TB.TS	SS	T.Best	T.Total
Avila	248	156	5	36	4.12	**36**	**0.08**	**36**	**0.08**	139.84
	248	156	10	25	106.4	23	10.24	**23**	**3.72**	191.16
Leon	211	184	5	47	13.4	**47**	**1.44**	47	2.84	139.36
	211	184	10	33	4.64	33	9.16	**33**	**4.44**	172.92
Salamanca	362	150	5	45	0.8	45	17.48	**45**	**0.36**	226.96
	362	150	10	31	227.68	31	118.32	**31**	**5.68**	296.12
Segovia	209	119	5	31	35.4	**31**	**0.6**	31	1	111
	209	119	10	22	87.88	22	8	**22**	**2**	120.48

11 of 19 cases, (marked in bold in Table 1), VNS in 5 cases and TS in 3 cases. Therefore, we can conclude that SS finds solutions of same quality and performs faster than other strategies.

We have run more test problems to show that SS is faster than the other strategies proposed in Mladenović et al. [3] with few facilities ($p \leqslant 10$). The instances refer to other provinces in the north of Spain: Avila, León, Salamanca and Segovia. In these instances $U \neq V$. The values of m and n for each one and the results are shown in Table 2 .These results lead to the same conclusions derived from Table 1.

Table 3
Results for the instances of OR-Lib, with $p \leqslant 10$ (MSCP)

OR Lib file	n	P	CPLEX	SS				
			LB	Value	Pop.Out	LBdv	T.Best	Total T.
Pmed1	100	5	965	967	19.62	0.2	0.03	0.88
Pmed2	100	10	845	845	17.14	0	0.13	1.75
Pmed3	100	10	878	880	17.86	0.22	0.01	1.69
Pmed6	200	5	1399	1406	14.28	0.49	0.33	6.58
Pmed7	200	10	1274	1283	13.03	0.70	0.22	11.8
Pmed11	300	5	1780	1780	12.3	0	0.11	87.84
Pmed12	300	10	1474	1474	10.18	0	0.16	41.36
Pmed16	400	5	2251	2263	11.33	0.53	0.21	65.52
Pmed17	400	10	2599	2609	13.07	0.38	5.63	29.82
Pmed21	500	5	3289	3318	13.36	0.88	5.70	43.13
Pmed22	500	10	3316	3322	13.38	0.18	3.24	34.61
Pmed26	600	5	2982	2987	9.95	0.17	0.83	389.15
Pmed27	600	10	3919	3941	13.13	0.56	0.85	207.14
Pmed31	700	5	5536	5576	15.78	0.72	1.05	195.7
Pmed32	700	10	4901	4943	13.99	0.86	5.80	111.12
Pmed35	800	5	2897	2910	7.28	0.45	19.62	178.13
Pmed36	800	10	4657	4702	11.76	0.96	26.22	222.48
Pmed38	900	5	3046	3075	6.85	0.95	3.24	399.84
Pmed39	900	10	5739	5780	12.88	0.71	2.14	315.01

5.2. Tests for the maximum set covering problem

In order to measure the quality of solutions obtained by our SS algorithm, a lower bound for each problem is obtained by executing CPLEX MIP 8.0 for 2 h for each instance. Since there are no libraries available for this model, we have used the instances used in Section 5.1., defining $t\_critical$ in the following way: $t\_critical = Round(P.B.Kn/1.5)$; where $P.B.Kn$ is the best known solution for the p-center problem, and *Round* is the function that rounds to the closest integer. On the other hand, q_i is generated randomly for values ranging from 1 to 100, (these values are available to interested readers).

In this case, the scatter search procedure has used the same parameter values as for the p-center problem: $Psize = 12$, $b_1 = b_2 = 3$; $\alpha = \beta = 0.8$ and $max\_iter = 5$. Table 3 shows the following results: the lower bound obtained by CPLEX MIP 8.0, the value of the solution (*value*), the percentage of the population that is left out (*Pop.Out*); the percentage deviation from the best lower bound (*LBDev*), the computational time till the best solution is found, (*T.Best*) and the total computational time (*Total T.*) for the SS algorithm (the latter two in seconds).

The quality of scatter search solutions is acceptable because the average deviation with respect to the lower bound is (on average) less than 1%. In addition, note that only in one case does SS exceed 20 s computational time before finding the best solution (26.22 s for $n = 800$, $p = 10$).

In the same way as the p-center problem, we have run more test problems to reaffirm the conclusions obtained with the instances of Or-Lib with few facilities ($p \leqslant 10$). These instances are the same as in

Table 4
Results for north of Spain instances with $p \leqslant 10$ (MSCP)

	M	n	P	CPLEX	SS				
				LB	Value	Pop.Out	LBdv	T.Best	Total T.
Avila	248	156	5	3746	3750	32.76	0.1	0.12	120.04
	248	156	10	1469	1474	12.87	0.34	3.6	61.68
Leon	211	184	5	9741	9753	28.50	0.12	0.8	120.08
	211	184	10	5320	5320	15.54	0	7.28	103
Salamanca	362	150	5	6015	6027	24.90	0.2	0.2	120.04
	362	150	10	3941	3941	16.28	0	6.36	115.08
Segovia	209	119	5	2454	2464	23.82	0.4	0.16	120
	209	119	10	662	662	6.40	0	2.72	37.16

Table 2 (files with the description of these instances are available to interested readers). In this case $t\_critical = 15$. The results are shown in Table 4.

6. Results with real data

In this section we show the results obtained from the tests performed with actual problems using the objective function of both models. The data refer to the area of Burgos (northern Spain). The objective is to analyze the best locations for a series of diabetes units that are chosen from a set of potentially suitable towns.

Initially, 452 locations with at least one known case of diabetes in the town were taken into account within the area. In fact, the data are estimations as we were not provided with exact data for each town, although we believe these estimates are close enough to reality as these data were supplied by the Local Public Health Authorities (http:/www.jcyl.es). Among these towns a subset of 152 locations were considered suitable for the project (they already had some kind of facility that could serve this purpose).

A time matrix (in minutes) was created with the 452 source locations, and the 152 that could potentially host the diabetes services (destination). To calculate traveling times, we used road data provided by the Spanish Center of Geographical Information (CNIG), and we estimated different speeds depending on the kind of road (national, regional, local roads, etc.). The road network data was used to calculate the time matrix using Djikstra's algorithm. The data employed (source and destination towns, time matrix, diabetes cases per location, etc.) are available to readers upon request.

Fig. 2 shows a map with the locations in the Burgos area that included cases of diabetes (grey), and those that were suitable for hosting the diabetes service (black).

For both models we used three values of p, $p = 5$, 7, and 10. For the maximum set covering model we used two values of *t_critical*,

$$t\_critical = Round(p\text{-}center\ \ solution/1.5)$$

and

$$t\_critical = Round(p\text{-}center\ \ solution/2),$$

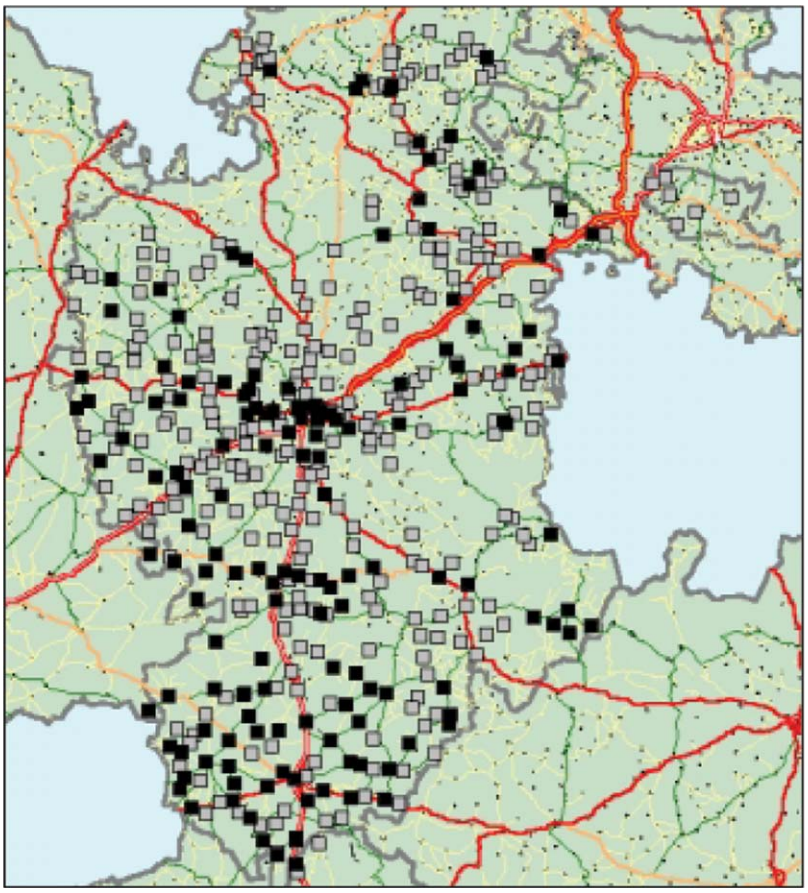

Fig. 2. Towns in the province of Burgos with cases of diabetes (grey), and towns that can host special diabetes units (black).

Table 5
Results for real instances for the p-center problem

P	VNS		SS	
↓	Value	Time to best	Value	Time to best
5	61	0.22	61	0.05
7	46	10.86	46	0.48
10	41	7.20	41	0.05

where *p-center solution* is the value of the best solution for the *p-center* problem. Therefore, we have three instances for the *p-center* problem, and six for the MSCP. We executed our SS algorithm and our implementation of the VNS algorithm for the p-center problem, and our SS algorithm and CPLEX for the 'maximum set covering problem'. A computational time of n (152) s was considered as the stopping condition for SS and VNS, and 2 h for CPLEX. The results (*value*) for both models as well as the

Table 6
Results for real instances for the MSCP

P	*T*_critical	CPLEX	SS		
↓		Lower bound	Value	LB dev	Time to best
5	41	119	121	1.68	0.22
	31	595	595	0	0.77
7	31	260	262	0.77	4.28
	23	768	768	0	0.17
10	27	204	206	0.98	0.6
	21	621	622	0.16	1.05

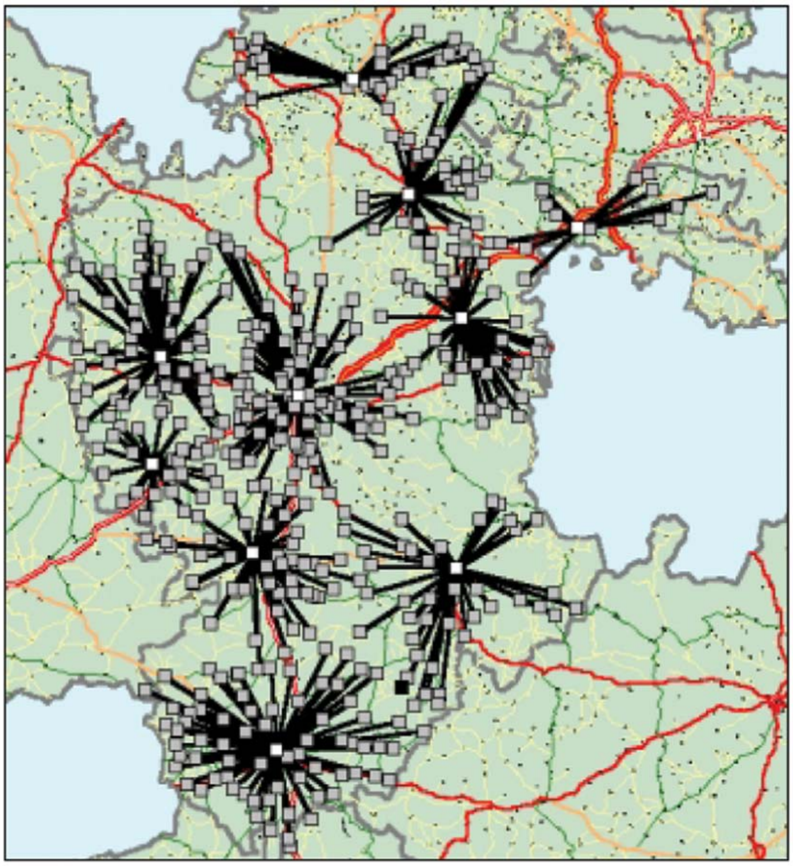
Fig. 3. Map showing (in white) the solutions provided by the SS algorithm for the *p*-center problem. The locations (in grey) are linked to the allocated facility by a line. The critical client is shown in black (maximum time to the closest facility).

computational time to obtain the best solution (*Time to Best*) by each algorithm are shown in Tables 5 and Table 6. In Table 6 the percentage deviation from the best lower bound (*LBDev*) is shown.

As is the case with the OR-Lib instances for the *p*-center problem, SS matches the solutions obtained by VNS faster. In addition, for the MSCP, the solution quality obtained by SS is perfectly acceptable

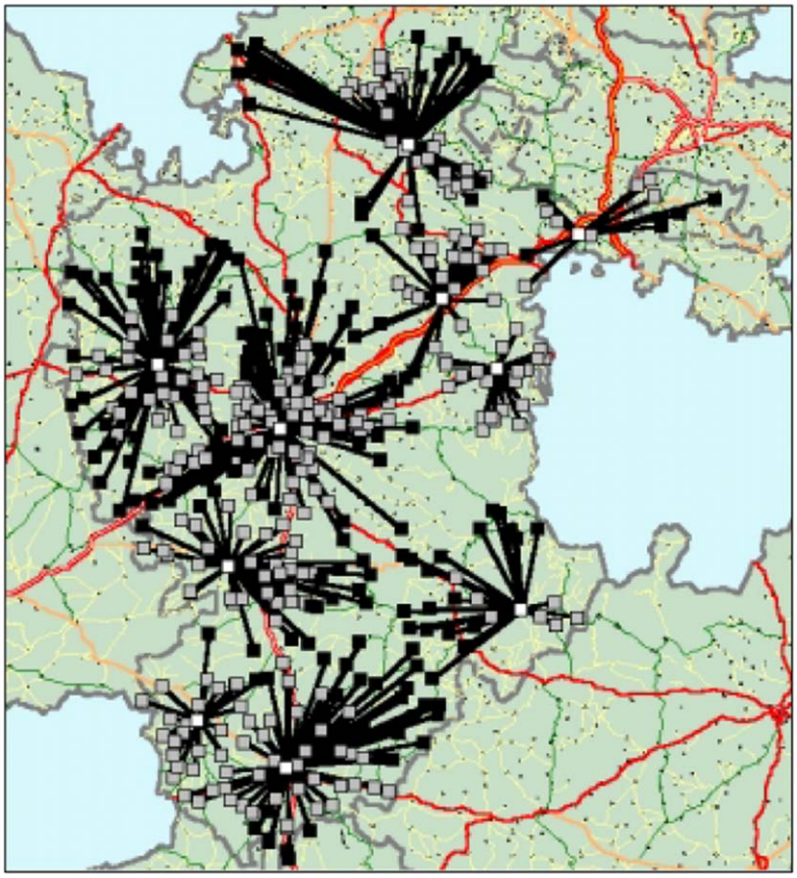

Fig. 4. Map showing (in white) the solutions given by the SS algorithm to the MSCP. The locations further away from its closest facility are in black ($t\_critical = 21$ min).

because the average deviation from the lower bound obtained with CPLEX is less than 1%. Fig. 3 shows a plan with the solutions obtained using SS $(p = 10)$ for the p-center problem.

Finally, Fig. 4 shows a map with the best SS solution for the MSCP problem ($p = 10$ and $t\_critical$=21).

7. Conclusions

Our research was motivated by the need to find solutions to the problem of allocating health resources throughout the area of Burgos, in the north of Spain. This is a rural area with a very dispersed population. Therefore, it was essential to design a method to provide good solutions with a low number of new facilities. This paper has dealt with two problems: minimizing the maximum distance between users and facilities (p-center), and minimizing the population further away from its closest available facility than a previously fixed time (‘maximum set covering’). Both problems were solved by a procedure based on the scatter search strategy and the results were analyzed using both actual and artificial data. Real data refer to the estimation of diabetes cases in the Burgos area. According to our results, our algorithm for the p-center problem provides similar quality solutions as other recently developed strategies, but in less computational time. With the MSCP, the quality of the solutions obtained is very high level (deviation from lower bound less than 1%).

Finally, this SS procedure has been used to solve other problems (with the necessary changes): clustering [27], logistics [28], labor scheduling [29] and school bus routing [30].

References

[1] Kariv O, Hakami SL. An algorithmic approach to network location problems. Part I: the p-center problem. SIAM Journal of Applied Mathematics 1979;37:513–38.

[2] Love RF, Morris JG, Wesolowsky GO. Facilities location: models and methods. Amsterdam: North-Holland; 1988.

[3] Mladenović N, Labbe M, Hansen P. Solving the p-center problem with tabu search and variable neigborhood search. Les Cahiers du GERAD, G-2000-35, 2001.

[4] Hansen P, Mladenović N. Variable neigborhood search for the p-Median. Location Science 1997;5:207–26.

[5] Whitaker R. A fast algorithm for the greedy interchange for large-scale clustering and median location problems. INFOR 1983;21:95–108.

[6] Mladenović N, Moreno JP, Moreno-Vega J. Tabu search in solving the p-facility location-allocation problems. Les Cahiers du GERAD, G-95-38, 1995.

[7] Mladenović N, Moreno JP, Moreno-Vega J. A chain-interchange heuristic method. Yugoslav. Journal Operations Research 1996;6(1):41–54.

[8] Resende MGC. Computing approximate solutions of the maximum covering problem using GRASP. Journal of Heuristics 1998;4:161–71.

[9] Glover F. A template for scatter search and path relinking. In: Hao J-K, Lutton E, Ronald E, Schoenauer M, Snyers D. editors. Artificial evolution, lecture notes in computer science, vol. 1363. Berlin: Springer; 1998. p. 13–54.

[10] Glover F, Laguna M, Martí R. Fundamentals of scatter search and path relinking. Control and Cybernetics 2000;39(3):653–84.

[11] Laguna M. Scatter search. In: Pardalos PM, Resende MGC, editors. Handbook of applied optimization. New York: Oxford University Press; 2002. p. 183–93.

[12] Feo TA, Resende MGC. A Probabilistic heuristic for a computationally difficult set covering problem. Operations Research Letters 1989;8:67–71.

[13] Feo TA, Resende MGC. Greedy randomized adaptive search procedures. Journal of Global Optimization 1995;2:1–27.

[14] Pitsoulis LS, Resende MGC. Greedy randomized adaptive search procedures. In: Pardalos PM, Resende MGC, editors. Handbook of applied optimization. Oxford: Oxford University Press; 2002. p. 168–82.

[15] Glover F, Laguna M. Tabu Search. Dordrecht: Kluwer Academic Publishers; 1997.

[16] Laguna M, Martí R. Scatter search. methodology and implementations. In: Dordrecht: Kluwer Academic Publishers; 2003 (312p.).

[17] Beasley JE. OR-Library: distributing test problems by electronic mail. Journal of the Operational Research Society 1990;41:1069–72.

[18] Mulvey JM, Beck MP. Solving capacitated clustering problems. European Journal Operations Research 1984;18(3): 339–48.

[19] Hansen P, Mladenović N. An introduction to variable neighborhood search. In: Voss S et al., editors. Metaheuristics advances and trends in local search paradigms for optimization. MIC-97. Dordrecht: Kluwer; 1998. p. 433–58.

[20] Hansen P, Mladenović N. First improvement may better than best improvement: an empirical study. Les Cahiers du GERARD. G-99-40, Montréal, Canada; October; 1999.

[21] Hansen P, Mladenović N. Variable neighbourhood search. Handbook of applied optimization, 2002.

[22] Hansen P, Mladenović N. Variable neighbourhood search. Handbook of Metaheuristics. Dordrecht: Kluwer; 2003, [Chapter 6].

[23] Campos V, Glover F, Laguna M, Martí R. An experimental evaluation of a scatter search for the linear ordering problem. Journal of Global Optimization 2001;21:397–414.

[24] Ribeiro CC, Ochoa E, Wernek RF. A Hybrid GRASP with Perturbations for the Steiner Problem in Graphs. INFORMS Journal on Computing 2002;14(3):228–46.
[25] Laguna M, Martí R. GRASP and path relinking for 2-layer straight line crossing minimization. INFORMS Journal on Computing 1999;11(1):44–52.
[26] Resende MGC, Ribeiro CC. A GRASP with path-relinking for permanent virtual circuit routing. Networks 2003;41: 104–14.
[27] Pacheco J. A scatter search approach for the minimum sum-of-squares clustering problem. Computers & Operations Research, accepted 2003.
[28] Delgado C, Laguna M, Pacheco J. Minimizing labor requirements in a periodic vehicle loading problem. University of Colorado at Boulder, 2003.
[29] Casado S, Laguna M, Pacheco J. A Heuristic to solve a large number of labor scheduling problems within a system for optimizing passenger flow at an airport. Working paper in Leeds School of business, University of Colorado at Boulder (see Manuel Laguna website: http://www-bus.Colorado.esu/Faculty/Laguna), 2003.
[30] Corberán A, Fernández E, Laguna M, Martí R. Heuristic solutions to the problem of routing school buses with multiple objectives. Journal of the Operational Research Society 2002;53(4):427–35.