Hybrid Heuristics for the Probabilistic Maximal Covering Location-Allocation Problem

Francisco de Assis Corrêa, Antonio Augusto Chaves, Luiz Antonio Nogueira Lorena

LAC – Computer and Applied Mathematics Laboratory
INPE – Brazilian Space Research Institute
12227-010 São José dos Campos – SP, Brazil.
e-mail address: lorena@lac.inpe.br fax number: +55-012-39456375 (L.A.N. Lorena).

Abstract

The Maximal Covering Location Problem (MCLP) maximizes the population that has a facility within a maximum travel distance or time. Numerous extensions have been proposed to enhance its applicability, like the probabilistic model for the maximum covering location-allocation with constraint in waiting time or queue length for congested systems, with one or more servers per service center. This paper presents one solution procedure for that probabilistic model, considering one server per center, using a Hybrid Heuristic known as Clustering Search (CS), that consists of detecting promising search areas based on clustering. The computational tests provide results for network instances with up to 818 vertices.

Keywords: Location problems, covering problems, congested systems, clustering search.

1. Introduction

The Maximal Covering Location Problem (MCLP) has been extensively studied in the literature since its formularization by [Church and ReVelle (1974)]. The main objective of the MCLP is to choose the location of facilities to maximize the population that has a facility within a maximum travel distance (or time). Thus, a population is considered covered if it is within a predefined service distance (or time) of at least one of the existing facilities. The MCLP does not require that all demand areas be covered, but offers service to the maximum population, considering the available resources.

Since its proposal, numerous extensions of the MCLP have been proposed to enhance its applicability, both in public and private sectors. Applications range from emergency services [Eaton et. al. (1986)] [Current and O'Kelly (1992)], hierarchical health [Moore and ReVelle (1982)], air pollution control [Hougland and Stephens (1976)], to congested systems [Marianov and Serra (1998)] [Marianov and Serra (2001)]. Solutions methods for the MCLP include linear programming relaxation

[Church and ReVelle (1974)], greedy heuristics [Daskin (1995)], Lagrangean Relaxation [Galvão and ReVelle (1996)], the Lagrangean/Surrogate heuristic [Lorena and Pereira (2002)], column generation [Pereira, Lorena and Senne (2007)]. Considerable revision of this subject can be found in [Chung (1986)], [Hale and Moberg (2003)], [Serra and Marianov (2004)] and [Galvão (2004)].

In many studies involving location problems, the distance (or time) between demand points and the facilities to which they have been allocated is the factor that represents the quality of the services that are given to the users. However, when service networks are projected, as in health systems or banking, the location of service centers has a strong influence on the congestion of each of them, and, consequently, the quality of services must be better defined and not only considering the travel distance or time. The centers must be located to allow the users to arrive at the center in an acceptable time, but it is desirable that the waiting time for service be no longer than a given time limit or that nobody stands in line with more than a predetermined number of other clients. These are important parameters in the measure of the desired quality [Marianov and Serra (1998)].

Congestion happens when a service center is not able to deal, simultaneously, with all the service requests that are made to it. Normally, the traditional models that deal with congestion include a capacity constraint, which forces the demand for service, normally constant in time and equal to an average, to be smaller than the maximum capacity of the center all the time. This is a deterministic approach to the problem, not considering the dynamic nature of the congestion. Depending on how the capacity constraint is developed, this means that the solution model produces idle servers, or results in a system that is not able to deal with all the demand [Marianov and Serra (1998) (2001)].

[Marianov and Serra (1998)] have proposed models based on the fact that the number of requests for services are not constant in time, but a stochastic process whose stochasticity of demand is explicitly considered in the capacity constraints. Instead of being limited to a maximum value, the authors define a minimum limit for the quality of the service reflected in the waiting time or the number of people waiting for service. The authors have developed the Queuing Maximal Covering Location-Allocation Model (QM-CLAM). Good reviews of the probabilistic models can be found in [Galvão (2004)] and [Brotcorne, Laporte and Semet (2003)].

[Correa and Lorena (2006)] have applied the Constructive Genetic Algorithm (CGA) in the QM-CLAM. The CGA works with a population formed by schemata (incomplete solutions) and structures (the complete solution), with the traditional operators: selection, recombination and mutation. The CGA differs from a classical GA in the way it evaluates the schemata, in its capacity to use heuristics to define the fitness evaluation function, and in its treatment of a dynamic population. A dynamic population is initially formed only by schemata, but may be enlarged after the use of recombination operators, or made smaller along the generations, guided by an evolutionary parameter. The dynamic population is built, generation after generation, by directly searching for well-adapted structures (a complete solution) and also for good schemata [(Lorena and Furtado (2001)] [Oliveira and Lorena (2004)] [Oliveira and Lorena (2005)].

The purpose of this paper is to examine the QM-CLAM with one server per service center and present a solution using a hybrid heuristic called Clustering Search (CS), which was proposed by [Oliveira and Lorena (2004) (2007)]. The CS consists of detecting promising areas of the search space, using an algorithm that generates solutions to be clustered. These promising areas may then be explored through local search methods as soon as they are discovered. The CS results are compared with those obtained by CGA and by the heuristic proposed by [Marianov and Serra (1998)].

The commercial solver CPLEX [ILOG (2006)] has been used to approximately solve the formulation for all problems, in order to validate the computational results of CS.

The remainder of the paper is organized as follows. Section 2 presents a mathematical formulation for QM-CLAM. Section 3 describes the heuristic that was used in this paper, and section 4 present the CS applied to QM-CLAM. Section 5 presents the computational results and section 6 concludes the paper.

2. QM-CLAM

The traditional Maximum Covering Location Problem (MCLP) proposed by [Church and ReVelle (1974)] cannot be used to deal with the congestion constraints, because there are no allocation variables. So, it is impossible to compute the requests for services that arrive at a center, and, consequently, to determine when congestion occurs. Thus, the MCLP has been rewritten as a p-median-like model, modified to accommodate the location and allocation variables. The objective is to maximize the covered population, considering a predefined number of service centers (p).

An integer linear programming formulation for the QM-CLAM is obtained by introducing the following variables. Let $y_j = 1$ if a center is located at a node j and $y_j = 0$ otherwise; $x_{ij} = 1$ if the users located at demand node i are allocated to a center located at j, and $x_{ij} = 0$ otherwise. We consider $i \in I$ and $j \in N_i$, such that I is a set of demand nodes, and N_i is either a set of candidate locations that are within a standard distance from node i, or the set of candidate locations which can be reached from node i within a certain standard time. Let a_i be the total population at demand node i. The formulation of the model is [Marianov and Serra (1998)]:

$$\text{v(QM-CLAM)} = \text{Max} \sum_{i,j} a_i x_{ij} \qquad \textbf{(1)}$$

Subject to

$$x_{ij} \leq y_j \qquad i \in I, \quad j \in N_i \qquad \textbf{(2)}$$

$$\sum_{j \in N_i} x_{ij} \leq 1 \qquad i \in I \qquad \textbf{(3)}$$

$$\begin{cases} \text{P(center } j \text{ has} \leq \text{b people in queue)} \geq \varphi & \textbf{(4)} \\ \text{or} \quad \text{P(waiting time at center } j \leq \tau) \geq \varphi & \textbf{(4a)} \end{cases}$$

$$\sum_{i \in I} y_i = p \qquad \textbf{(5)}$$

$$y_j, x_{ij} \in \{0,1\}, \quad i \in I, \quad j \in N_i \qquad \textbf{(6)}$$

The objective (1) maximizes the population allocated to a center. Constraint (2) defines that a demand point *i* can be allocated to a node *j* only if there is a center in *j*. Constraint (3) forces each demand node *i* to be allocated to at most one service center *j*. Constraints (4) ensure that each center has no more than *b* people on a line, with a probability of at least φ. Constraints (4a), make the total time spent by a user at a center *j* be shorter than, or equal to τ, with a probability of at least φ. Constraint (5) sets the number of centers to be located. Constraints (6) define the integrality requirements.

In order to write constraint (4), there is an assumption that requests for service at each demand node *i* appear according to a Poisson process with intensity f_i. The service requests at a center are the union of the requests for service of the demand nodes, and they can be described as another stochastic process, equal to the sum of several Poisson processes, with intensity ω_j:

$$\omega_j = \sum_{i \in I} f_i x_{ij} \qquad \textbf{(7)}$$

which means that, if the variable x_{ij} is one, node *i* is allocated to center *j* and the corresponding intensity f_i will be included in the computation of ω_j. The well-known results for a M/M/1 queuing system have been considered for each center and its allocated users [Larson and Odoni (1981)]. An exponentially distributed service time, with an average rate μ_j, has been considered in those models, where $\mu_j \geq \omega_j$; otherwise, the system does not reach the equilibrium.

The QM-CLAM can be formally stated as:

$$\text{v(QM-CLAM)} = \text{Max} \sum_{i,j} a_i x_{ij} \quad \textbf{(1)}$$

Subject to

$$x_{ij} \le y_j \qquad i \in I, \quad j \in N_i \quad \textbf{(2)}$$

$$\sum_{j \in N_i} x_{ij} \le 1 \qquad i \in I \quad \textbf{(3)}$$

$$\text{or} \begin{cases} \sum_{i \in I} f_i x_{ij} \le \mu_j \sqrt[b+2]{1-\varphi} \qquad j \in N_i & \textbf{(4)} \\ \sum_{i \in I} f_i x_{ij} \le \mu_j + \frac{1}{\tau} \ln(1-\varphi) \quad j \in N_i & \textbf{(4a)} \end{cases}$$

$$\sum_{i \in I} y_i = p \quad \textbf{(5)}$$

$$y_j, x_{ij} \in \{0,1\}, \quad i \in I, \quad j \in N_i \quad \textbf{(6)}$$

The complete calculations can be found in Marianov and Serra (1998). The QM-CLAM belongs to the NP-Complete class of problem [Pirkul and Schilling (1991)]. Even when using commercial solvers, it is not always possible to find the optimal solution in a reasonable computational time, due to its classification and to the problem size. Therefore, alternative methods are investigated. In the next Section a solution based on the Clustering Search will be presented.

3. Clustering Search Method

The Clustering Search (CS), proposed by [Oliveira and Lorena (2004) (2007)], employs clustering for detecting promising areas of the search space. The sooner these strategic areas can be identified, the sooner a more accurate search strategy can be applied. An area can be seen as a search subspace defined by a neighborhood relationship in metaheuristic coding space. In the CS, a clustering process is executed simultaneously to a metaheuristic, identifying groups of solutions that deserve special attention.

The CS attempts to locate promising search areas by framing them by clusters. A cluster can be defined as a tuple G = {c; r; s} where *c*, *r* and *s* are, respectively, the center and the radius of the area, and a search strategy associated to the cluster.

The center of the cluster is a solution that represents the cluster, identifying its location inside the search space. Initially, the centers of the clusters are obtained randomly; but progressively, they tend to fall along really promising points in the close subspace. The radius *r* establishes the maximum distance, starting from the center, for which a solution can be associated to the cluster. For example, in combinatorial optimization, *r* can be defined as the number of movements needed to change a solution into another one. The search strategy is a systematic search intensification, in which solutions of a cluster interact among themselves along the clustering process, generating new solutions.

The CS consists of four conceptually independent components with different attributions:

- search metaheuristc (SM);
- iterative clustering (IC);
- analyzer module (AM);
- local searcher (LS);

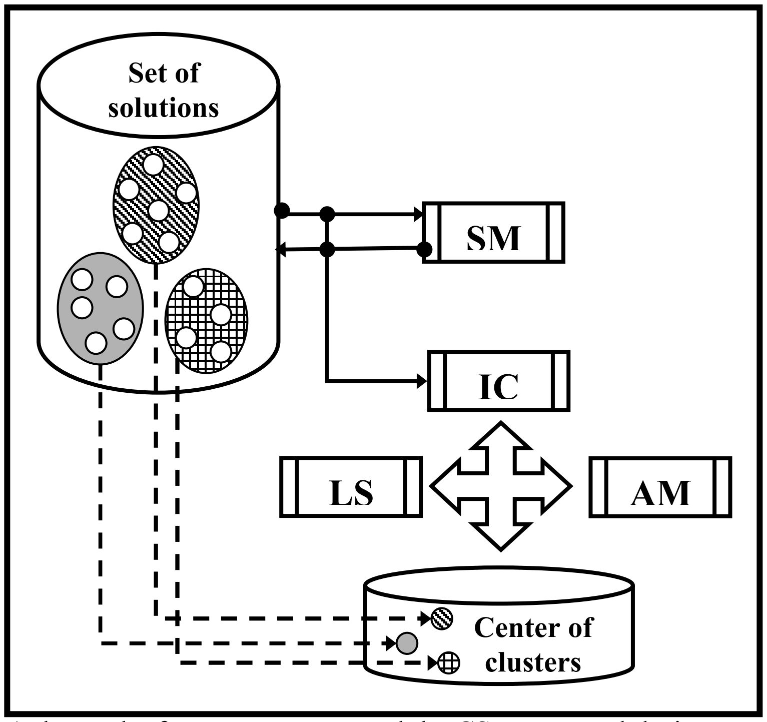

Figure 1 shows the four components and the CS conceptual design.

Figure 1. *CS Components*

The SM works as a full-time solution generator. The algorithm is executed independently of the remaining components and must be able to provide a continuous generation of solutions to the clustering process. Clusters are maintained, simultaneously, to represent these solutions.

The IC aims to gather similar solutions into groups, identifying a representative cluster center for them. To avoid extra computational effort, IC is designed as an online process, in which the clustering is progressively fed by solutions generated in each iteration of SM. A maximum number of clusters NC is an upper bound value that prevents an unlimited cluster creation. A distance metric must be defined, a priori, allowing a similarity measure for the clustering process.

The AM provides an analysis of each cluster, at regular intervals, indicating a probable promising cluster. A cluster density, δ_i, is a measure that indicates the activity level inside the cluster. For simplicity, δ_i counts the number of solutions generated by SM and allocated to the cluster i. Whenever δ_i reaches a certain threshold, indicating that some information template has become predominantly generated by SM, that information cluster must be better investigated to accelerate the convergence process on it.

Finally, the LS is a local search module that provides the exploitation of a supposed promising search area framed by a cluster. This process is executed each time AM finds a promising cluster. LS can be considered as the particular search strategy associated with the cluster, i.e., a problem-specific local search to be applied to the cluster.

4. CS for QM-CLAM

A version of CS for the QM-CLAM is presented in this section. A solution was represented through a computational structure that contains the located centers, the allocation of the demand points and the objective value. The allocation procedure tests all the service centers defined by the solution and chooses one to allocate each demand point that satisfies: a) the coverage; b) its capacity; c) the service center that will have the minimum accumulated demand.

The whole CS pseudo-code for the QM-CLAM is presented in Figure 2. The pseudo-code shows where the different components of the procedure fit.

The component SM, responsible for generating solutions to the clustering process, was the metaheuristic Greedy Randomized Adaptive Search Procedure (GRASP) [Feo and Resende (1995)]. The GRASP is basically composed of two phases: a construction phase, in which a feasible solution is generated, and a local search phase, in which the constructed solution is improved.

```
procedure CS
    // component SM
    while (number of iterations is not satisfied) do
        // construction phase of GRASP
        $s = \varnothing$
        while (solution not built) do
            compute candidate list (C)
            RCL = C * $\alpha$
            $e$ = select at random a value of RCL
            $s = s \cup \{e\}$
        end-while
        // local search phase of GRASP
        $s$ = Location-Allocation heuristic(s);

        // component IC
        calculate the distance of the solution GRASP ($s$) and the clusters
        insert the solution in the most similar cluster ($c_i$)
        apply the assimilation process – path-relinking(s, $c_i$)

        // component AM
        verify if the cluster can be considered promising. If so, the LS component
             is applied to it

        // component LS
        apply the location-allocation heuristic to the promising cluster
    end-while
end procedure
```

Figure 2. *CS pseudo-code*

The construction phase of GRASP uses the steps 1 to 5 of the greedy heuristic reported in [Marianov and Serra (1998)], but changes the way to locate a server in step 5. A candidate list (C) is built in step 1 with all candidate locations. A restrict candidate list (RCL) is built in step 5 with a percentage α of C (RCL = C*α), with the highest current total incoming call rate. One node is chosen randomly in the RCL. Steps 1 to 5 used in this phase are shown in Figure 3.

The local search phase of GRASP uses the location-allocation heuristic, proposed by [Lorena and Pereira (2002)]. The solutions obtained from the construction phase can be improved by searching for a new center in each covered area, swapping the current center by a allocated node of the same covered area, and changing the allocation solution, as shown in Figure 4. The algorithm is shown in Figure 5. That change may alter both allocation and the coverage of the QM-CLAM solution; so it is necessary to recalculate allocation.

The IC is the CS's core, working as a classifier, keeping in the system only relevant information, and guiding search intensification in the promising search areas. A maximum number of clusters (*NC*) is defined a priori. The i^{th} cluster has its own center c_i and a radius r, like the other clusters.

Heuristic

Step 1:

Make a list of the candidate locations j (candidate list C), ordered by a decreasing call rate (each demand node is a potential center location). Call this list D_j.

For each candidate location, compute the right-hand side of equation (4) or (4a), depending on the model being utilized. For both equations, the right-hand side is initially computed using all previously known values of μ, b and φ (equation 4) or μ, b and τ (equation 4a).

For each candidate location j, make a list of all demand nodes i within the standard distance, ordered by increasing distance to the candidate location. Call this list D_{ji}.

Step 2:

For each candidate location, make the current total incoming call rate equal zero ($\omega_{inc,\, j}$).

Step 3:

Starting with the first candidate location j on the list D_j, add to its incoming call rate $\omega_{inc,\, j}$, the call rate f_i of the first node on the list D_{ji}. Then, add the call rate f_i of the second node on the list D_{ji}, then the third, and so on, until the point where adding any extra demand node would exceed the limit value of calls $\omega_{inc,}$. Temporarily allocate all these demand nodes to a hypothetical service center at node j.

Step 4:

Repeat step 3 for all nodes in list D_j. Note that the same demand node could be temporarily allocated to several candidate locations.

Step 5:

Locate a service center on the node chosen randomly within a percentage α of the highest current total incoming call rates. Take all demand nodes allocated to it out of all the lists D_{ji} of all potential centers. Allocate them definitively to the located center.

Step 6:

Repeat steps 2 to 5 until all available centers are located.

End-Heuristic

Figure 3. *Heuristic used in the construction phase of GRASP*

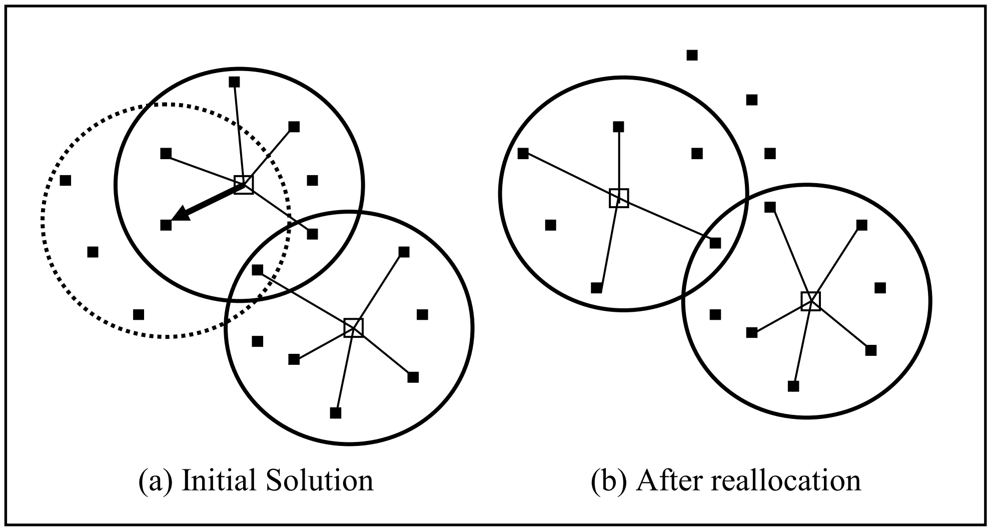

Figure 4. *Reallocation of nodes.*

```
// Algorithm for primal solutions improvement (Location-Allocation heuristic)
// Let $v_c$ be the best current objective value.
**procedure** Location-Allocation;
    **while** ($v_c$ increases)
        **for** k = 1…p
            interchange center and allocated non-center nodes in the coverage area
            $k$;
            calculate the correspondent value of $v$ of the best reallocation;
            **if** $v > v_c$
                update the center node for the coverage area k;
                $v_c = v$;
            **end-if**;
        **end-for**;
    **end-while**;
**end procedure**
```

Figure 5. *Primal solutions improvement*

Solutions generated by GRASP (SM) are passed to IC that attempts to group these solutions as known information in a cluster, chosen according to a *distance metric*. The solution activates the closest center c_i (cluster center that minimizes the distance metric), and a disturbance is applied to it. In this paper, the *metric distance* is the number of different located centers between the GRASP and the center of the cluster solutions. When there are a larger number of different located centers between the GRASP and the cluster solution center it increases the dissimilarity. The disturbance is an *assimilation process*, in which the center of the cluster is updated by the new generated solution. In this paper, this process is the path-relinking method [Glover (1996)], that generates several points (solutions) along the path that connects the solution generated by GRASP and the one in the center of the cluster. Since each point is evaluated by the objective function, the assimilation process itself is an intensification mechanism inside the clusters. The new center c_i is the best-evaluated solution obtained in the path.

Path-relinking starts from two elite solutions. The first is the solution that comes from the SM component ($s_{initial}$). The second is the closest cluster center c_i (s_{guide}). The procedure starts by computing the symmetric difference between the two solutions $\Delta(s_{initial}, s_{guide})$, i.e. the set of moves needed to reach s_{guide} from $s_{initial}$. A path of solutions is generated, linking $s_{initial}$ and s_{guide}. At each step, the procedure examines all moves $m \in \Delta(s_{initial}, s_{guide})$ from the current solution s and selects the one which results in the best cost solution, applying the best move (m^*) to solution s ($s \oplus m^*$). The set of available moves is updated. The procedure terminates when s_{guide} is reached, i.e. when $\Delta(s_{initial}, s_{guide}) = \emptyset$. The best solution s^* in this path is returned by the algorithm. In this paper, one move is to swap a median of the $s_{initial}$ by a median of the s_{guide}, changing the allocation solution. Figure 6 illustrates the pseudo-code of the path-relinking and Figure 7 shows an example for a 30-node network with five service centers.

```
**procedure** Path-relinking($s_{initial}, s_{guide}$)
    compute symmetric difference $\Delta(s_{initial}, s_{guide})$
    $f^* = \min \{f(s_{initial}), f(s_{guide})\}$
    $s^* = \text{argmin} \{f(s_{initial}), f(s_{guide})\}$
    $s = s_{initial}$
    **while** ($\Delta(s_{initial}, s_{guide}) \neq \emptyset$)
        $m^* = \text{argmin}\{f(s \oplus m): m \in \Delta(s_{initial}, s_{guide})\}$
        $\Delta(s \oplus m^*, s_{guide}) = \Delta(s, s_{guide}) \setminus \{m^*\}$
        $s = s \oplus m^*$
        **if** $f(s) < f^*$
            $f^* = f(s)$
            $s^* = s$
        **end-if**
    **end-while**
    **return** $s^*$
**end procedure**
```

***Figure 6**. Path-relinking algorithm*

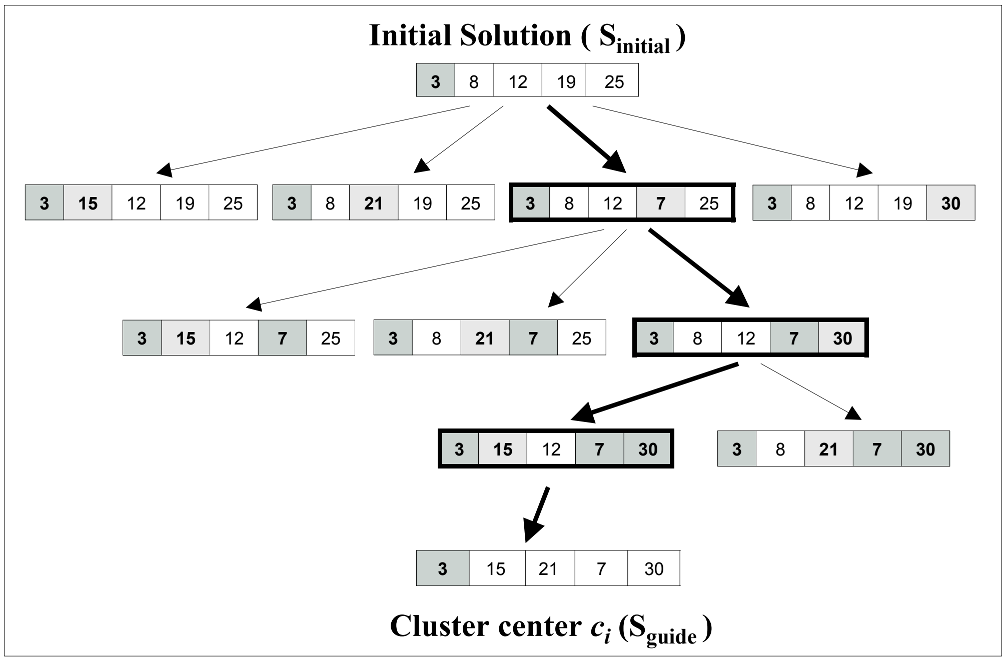

***Figure** 7. Path-relinking moves*

The AM component is executed whenever a solution is assigned to a cluster, verifying if the cluster can be considered promising. A cluster becomes promising when it reaches a density λ_i,

$$\lambda_i \geq PD.\frac{NS}{|C_t|} \tag{8}$$

where, NS is the number of solutions generated in the interval of analysis of the clusters, $|C_t|$ is the number of clusters in the iteration t, and *PD* is the desirable cluster density beyond the normal density, obtained if NS were equally divided to all clusters. The center of a promising cluster is improved through the LS. The AM also performs the *cooling* of all clusters, halving the λ_i value.

The component LS is activated when the AM discovers a promising cluster. The LS implementation uses the idea behind the local search phase of GRASP, the location-allocation heuristic [Lorena and Pereira (2002)], changing the way of searching for a new service center. Rather than swapping the current center by an allocated node of the same covered area, it swaps the current center by all node in the same covered area, changing the allocation solution, recalculating the coverage, and taking the best solution found as the new one. It permits searching for good solutions in different search spaces. After all, the LS applies the Swap2 heuristic (SW2) to all located centers, examining all possible swaps of two allocated and two non allocated demand points. The best one is performed and the process is repeated until no improvements occur. Therefore, the location-allocation heuristic searches for the best center location, and the Swap2 searches for the best allocation inside the best location found. After applying both heuristics, the cluster center is updated if the new solution is better than the previous one.

5. Computational Results

The CS was tested in the 30-node network provided by [Marianov and Serra (1998)] and in 324-node and 818- node networks obtained from a geographical data base of São José dos Campos, Brazil. These two networks were increased by fictitious population for each demand point and they are available at http://www.lac.inpe.br/~lorena/instancias.html. The 30-node network is shown in Table 1.

By varying the p, b, μ, φ e τ parameters, various problems have been created. The results from CS have been compared to the results obtained using the commercial solver CPLEX, version 10.0 [ILOG (2006)], from the CGA [Correa and Lorena (2006)] and from the heuristic proposed by Marianov and Serra (1998).

For the implementation of the QM-CLAM, the service centers are primary health care centers, with one physician at each center. Each demand point is also a potential center location (candidate location), and the distances are Euclidean. For the 30-node network, the following parameters were adopted: covered distance equaled to 1.5 miles; average service time ($1/\mu$) was set at 20 minutes; call rates were set at 0.015 times the node population for the constrained queue length and 0.006 times the node population for the constrained waiting time, all defined in [Marianov and Serra (1998)].

For the 324-node and 818-node networks, the following parameters were used: average service time ($1/\mu$) was set at 15 minutes; call rates were set at 0.01 times the node population for both constrained queue length and constrained waiting time. For the 324-node network, covered distance equaled 250 meters, and for the 818-node, equaled 750 meters.

The problems were codified in the following way: number of points, number of centers, constraint type (0 for the constrained queue length and 1 for the constrained waiting time), number of people in line or waiting time, and probability. For example: 324_20_0_2_95, means there are 324 points, 20 centers, a constrained queue length, with a maximum of two people in line, with the probability of at least 95%. The CS code was written in Object Pascal. The times in tables are shown in seconds and were determined in a Pentium IV 3 GHz computer, with 1Gb of RAM memory. The times for the attainment of the CPLEX solutions were limited to 3 hours (10800 seconds). The instances marked with one * define a stop in the execution due to an out of memory error.

Table 2 shows the results obtained for the 30-node network; Table 3 shows the results for the 324-node network and Table 4, for the 818-node network. The values of Gap CPLEX equal to zero mean that the optimum was reached.

The results for the heuristic defined in [Marianov and Serra (1998)] (MS_Heuristic) are shown in three columns: Solution, Time, and Deviation. The Dev column (Deviation) shows, in percentage, the relative error between the heuristic and CPLEX solutions. Its values are calculated by (CPLEX solution – MS_Heuristic solution)/(CPLEX solution).

The results for the CGA and CS reflect fifty executions of each problem and are shown in four columns: the best value found (Best column), the average value (Average column), the average time (Time column) and the deviation (Dev column), that reflects, in percentage, the relative error between the best solution for the CGA or CS and the CPLEX, and is calculated by (CPLEX solution – Best solution)/(CPLEX solution). Therefore, the negative values of the deviations indicate that the best solution for the CS was better than the CPLEX solution. The values in boldface show the best solutions found.

The following values of parameters for CS were adjusted through several executions and are also based in [Oliveira and Lorena (2004)]. The following parameters produced good results:

- number of solutions generated at each analysis of the clusters *NS* = 15;
- maximum number of clusters *NC* = 5;
- density pressure *PD* = 2;

In 75% of the cases, the full CS procedure found objective values that were better than those found by the GRASP alone (SM component of CS) for the 324-node network and in about 92% for the 818-node network. It shows that the other components of CS contribute substantially to improve the solution values.

As shown in Table 2, for the 30-node network, CS found the optimal value in about 65% of the problems, considering the average results, and in about 81%, considering the best values, with better average time than CGA and CPLEX.

For the 324-node network, CPLEX found better results in 100% of the tests. However, it had found serious problems in two instances due to out of memory error, and it did not find the optimal value for three problems in three hours of execution. The running times of the CS were very competitive compared to CPLEX, with good solution values. CS found better results than the heuristic by Marianov and Serra (1998) and CGA.

For the 818-node network, the running times of the CS were very competitive as compared to the CPLEX, with better solution values in about 30% of the problems. CPLEX did not find the optimal value in about 46% of the instances tested in three hours of execution, and in some of them there were gaps above 100%. CS found the same optimal values as CPLEX with better computational times. CS also obtained better results than the heuristic by Marianov and Serra (1998) and CGA.

Table 1. *30-node network*

Node	X	Y	Population	Node	X	Y	Population
1	3.2	3.1	710	**16**	3.0	5.1	110
2	2.9	3.2	620	**17**	1.9	4.7	100
3	2.7	3.6	560	**18**	1.7	3.3	100
4	2.9	2.9	390	**19**	2.2	4.0	90
5	3.2	2.9	350	**20**	2.5	1.4	90
6	2.6	2.5	210	**21**	2.9	1.2	90
7	2.4	3.3	200	**22**	2.4	4.8	80
8	3.0	3.5	190	**23**	1.7	4.2	80
9	2.9	2.7	170	**24**	6.0	2.6	80
10	2.9	2.1	170	**25**	1.9	2.1	80
11	3.3	2.8	160	**26**	1.0	3.2	70
12	1.7	5.3	150	**27**	3.4	5.6	60
13	3.4	3.0	140	**28**	1.2	4.7	60
14	2.5	6.0	120	**29**	1.9	3.8	60
15	2.1	2.8	120	**30**	2.7	4.1	60

6. Conclusions

This paper has presented a solution for the Probabilistic Maximal Covering Location-Allocation Problem using Clustering Search (CS). The CS is a new method that has been applied with success in some combinatorial optimization problems, such as the pattern sequencing problem [(Oliveira and Lorena (2004)] and prize collecting traveling salesman problem [Chaves and Lorena (2005].

The idea of the CS is to avoid applying a local search heuristic to all solutions generated by a metaheuristic, which can make the search process impracticable since it is time consuming, mainly when the heuristic has a high computational cost. The CS detects the promising regions in the search space during solution generation process, i.e., to detect promising regions becomes an interesting alternative in order to prevent the indiscriminate application of such heuristics.

This paper reports results of different methods applied to QM-CLAM. CS got better results than others heuristics (CGA and the heuristic by Marianov and Serra) and it founds good values comparing to CPLEX. CS has two advantages over CPLEX: execution time, and the cost of a commercial solver.

The results show that the CS approach is competitive for the resolution of this problem in reasonable computational times. For some instances of 30-node and 818-node networks, the optimal values were found. Therefore, these results validate the CS application to the QM-CLAM.

Further studies can be done which analyze others metaheuristics to generate solutions for the clustering process of CS, such as the Ant Colony System, Tabu Search and Genetic Algorithm, and which implement new local search heuristics for the QM-CLAM. Moreover, larger instances of this model can be generated and solved.

Table 2. *Results for the 30-node network*

Problem	CPLEX			CGA				MS_Heuristic			GRASP			CS			
	Sol.	Gap	Time	Best	Average	Time	Dev.	Sol.	Time	Dev.	Best	Average	Time	Best	Average	Time	Dev.
30_2_0_0_85	**3700**	0	0.22	**3700**	**3700**	0.31	0.00	**3700**	0.00	0.00	**3700**	3700.0	0.007	**3700**	3700.0	0.009	0.00
30_3_0_0_85	**5390**	0	0.12	**5390**	**5390**	0.51	0.00	5210	0.00	3.34	**5390**	5390.0	0.019	**5390**	5390.0	0.025	0.00
30_2_0_1_85	**5100**	0	3.31	5090	5090	0.36	0.20	4630	0.00	9.22	5090	5090.0	0.016	5090	5090.0	0.018	0.20
30_3_0_1_85	**5390**	0	0.08	**5390**	**5390**	0.48	0.00	5210	0.00	3.34	**5390**	5390.0	0.021	**5390**	5390.0	0.024	0.00
30_2_0_2_85	**5210**	0	0.20	**5210**	**5210**	0.38	0.00	4780	0.00	8.25	**5210**	5210.0	0.015	**5210**	5210.0	0.016	0.00
30_3_0_2_85	**5390**	0	0.12	**5390**	**5390**	0.47	0.00	5210	0.00	3.34	**5390**	5390.0	0.022	**5390**	5390.0	0.023	0.00
30_5_0_0_95	**5330**	0.38	10800	**5330**	5323	0.80	0.00	5210	0.00	2.25	**5330**	5312.8	0.032	**5330**	5317.6	0.041	0.00
30_6_0_0_95	**5410**	1.11	10800	**5410**	5392	0.84	0.00	5390	0.00	0.37	5390	5390.0	0.029	**5410**	5391.0	0.037	0.00
30_3_0_1_95	**5270**	0	116.00	5240	5240	0.48	0.57	5080	0.00	3.61	5240	5239.8	0.022	5240	5240.0	0.024	0.57
30_4_0_1_95	**5390**	0	3.77	**5390**	**5390**	0.54	0.00	5260	0.00	2.41	**5390**	5390.0	0.022	**5390**	5390.0	0.026	0.00
30_2_0_2_95	**4520**	0	2.11	**4520**	4513	0.30	0.00	4470	0.00	1.11	**4520**	4520.0	0.013	**4520**	4520.0	0.015	0.00
30_3_0_2_95	**5390**	0	0.12	**5390**	**5390**	0.50	0.00	5230	0.00	2.97	**5390**	5390.0	0.023	**5390**	5390.0	0.027	0.00
30_4_1_48_90	**1920**	0	0.41	**1920**	**1920**	0.66	0.00	1890	0.00	1.56	**1920**	1920.0	0.011	**1920**	1920.0	0.014	0.00
30_5_1_48_90	**2400**	0	0.41	2390	2390	0.74	0.42	2280	0.00	5.00	**2400**	2398.8	0.013	**2400**	2398.8	0.019	0.00
30_3_1_49_90	**2160**	0	0.31	**2160**	**2160**	0.53	0.00	**2160**	0.00	0.00	**2160**	2160.0	0.007	**2160**	2160.0	0.009	0.00
30_4_1_49_90	**2880**	0	0.41	**2880**	2877	0.59	0.00	2870	0.00	0.35	**2880**	2880.0	0.008	**2880**	2880.0	0.013	0.00
30_5_1_50_90	**4700**	0	4.09	**4700**	**4700**	0.73	0.00	4670	0.00	0.64	**4700**	4700.0	0.018	**4700**	4700.0	0.028	0.00
30_6_1_50_90	**5390**	0	350.52	**5390**	**5390**	0.97	0.00	5060	0.00	6.12	**5390**	5390.0	0.027	**5390**	5390.0	0.038	0.00
30_5_1_40_85	**3050**	0	3.66	3020	3001	0.74	0.98	2910	0.00	4.59	**3050**	3033.2	0.016	**3050**	3033.2	0.021	0.00
30_6_1_40_85	**3610**	0	31.62	**3610**	**3610**	0.91	0.00	3480	0.00	3.60	**3610**	3606.8	0.021	**3610**	3608.8	0.031	0.00
30_7_1_40_85	**4060**	0	16.89	**4060**	**4060**	1.03	0.00	3860	0.00	4.93	**4060**	4060.0	0.028	**4060**	4060.0	0.039	0.00
30_6_1_41_85	**5330**	0.19	10800	5300	5274	1.00	0.56	5120	0.00	3.94	5270	5270.0	0.028	**5330**	5286.4	0.038	0.00
30_7_1_41_85	**5410**	0	43.56	5390	5390	0.88	0.37	5300	0.00	2.03	5390	5390.0	0.023	5390	5390.0	0.035	0.37
30_8_1_41_85	**5470**	0	0.05	**5470**	**5470**	0.95	0.00	5390	0.00	1.46	**5470**	5470.0	0.019	**5470**	5470.0	0.032	0.00
30_4_1_42_85	**4600**	0	0.75	**4600**	**4600**	0.61	0.00	4550	0.00	1.09	**4600**	4600.0	0.016	**4600**	4600.0	0.022	0.00
30_5_1_42_85	**5390**	0	6.50	**5390**	**5390**	0.77	0.00	5210	0.00	3.34	**5390**	5390.0	0.027	**5390**	5390.0	0.035	0.00

Table 3. *Results for the 324-node network*

Problem	CPLEX			CGA				MS_Heuristic			GRASP			CS			
	Solution	Gap	Time	Best	Average	Time	Dev.	Solution	Time	Dev.	Best	Average	Time	Best	Average	Time	Dev.
324_10_0_0_95	**21460**	0	67.4	21431	21373.0	8.63	0.14	21386	0.14	0.34	21446	21441.2	2.05	21455	21447.2	3.16	0.023
324_10_0_1_95	**35360**	0	242.1	35342	35304.0	7.79	0.05	35250	0.16	0.31	35339	35336.5	2.06	35359	35354.6	3.17	0.003
324_10_0_2_95	**45390**	0	305.3	45347	45245.0	7.81	0.09	45300	0.20	0.20	45341	45334.6	2.01	45374	45354.6	3.10	0.035
324_10_0_0_85	**37180**	0	481.0	37145	37069.0	8.10	0.09	37081	0.22	0.27	37157	37155.8	2.24	37173	37163.2	3.46	0.019
324_10_0_1_85	**51000**	0	297.3	50880	50711.0	7.69	0.24	50750	0.20	0.49	50948	50946.3	2.30	50948	50946.3	3.41	0.102
324_10_0_2_85	**59740**	0	961.1	59624	59437.0	7.62	0.19	59598	0.20	0.24	59693	59688.5	2.24	59693	59688.5	3.33	0.079
324_10_1_40_85	**27700**	0	292.1	27675	27602.0	8.54	0.09	27583	0.17	0.42	27670	27666.6	2.18	27698	27692.1	3.37	0.007
324_10_1_41_85	**29360**	0	216.5	29324	29260.0	8.27	0.12	29288	0.14	0.25	29335	29330.9	2.23	29351	29341.9	3.52	0.031
324_10_1_42_85	**30950**	0	1074.1	30932	30895.0	8.34	0.06	30902	0.17	0.16	30928	30920.8	2.21	30948	30943.3	3.33	0.006
324_10_1_48_90	**26920**	0	421.6	26883	26835.0	8.35	0.14	26855	0.14	0.24	26899	26896.6	2.18	26917	26910.6	3.50	0.011
324_10_1_49_90	**28330**	0	359.1	28280	28221.0	8.28	0.18	28206	0.25	0.44	28295	28285.0	2.24	28318	28310.3	3.43	0.042
324_10_1_50_90	**29680**	0	529.9	29641	29593.0	8.26	0.13	29638	0.22	0.14	29658	29656.1	2.24	29672	29665.5	3.36	0.027
324_20_0_0_95	**42920**	0.021*	9672.3	42577	42318.0	24.30	0.80	42714	0.73	0.48	42813	42792.2	4.24	42840	42804.9	9.37	0.186
324_20_0_1_95	**70720**	0	9911.9	70471	70308.0	23.40	0.35	70368	0.74	0.50	70561	70529.8	4.43	70656	70628.2	9.06	0.090
324_20_0_2_95	**90778**	0.003	10800.0	89970	89355.0	24.15	0.89	90424	0.81	0.39	90550	90518.9	4.64	90556	90521.2	9.40	0.245
324_20_0_0_85	**74315**	0.061*	4791.9	73407	73001.0	23.69	1.22	73981	0.83	0.45	74165	74106.0	4.80	74165	74106.7	9.71	0.202
324_20_0_1_85	**101928**	0.071	10800.0	99576	98353.0	24.69	2.31	100628	1.02	1.28	101374	101177.4	4.88	101374	101177.4	9.07	0.544
324_20_0_2_85	**119445**	0.030	10800.0	116639	115235.0	23.33	2.35	118451	0.77	0.83	118771	118613.6	4.86	118771	118613.6	8.99	0.564
324_20_1_40_85	**55397**	0.005	10800.0	54804	54414.0	23.92	1.07	55006	0.84	0.71	55193	55147.4	4.63	55306	55226.7	9.65	0.164
324_20_1_41_85	**58720**	0	828.9	58009	57571.0	24.70	1.21	58577	1.02	0.24	58602	58582.6	4.72	58604	58583.4	10.33	0.198
324_20_1_42_85	**61900**	0.002	10800.0	61545	61266.0	23.81	0.57	61637	0.88	0.42	61746	61715.1	4.68	61847	61822.6	9.74	0.086
324_20_1_48_90	**53839**	0.002	10800.0	53300	52958.0	24.32	1.00	53377	0.63	0.86	53647	53607.5	4.54	53793	53689.5	9.82	0.085
324_20_1_49_90	**56651**	0.016	10800.0	56216	55813.0	24.15	0.77	56180	1.34	0.83	56321	56254.3	4.59	56532	56429.9	9.55	0.210
324_20_1_50_90	**59357**	0.005	10800.0	58941	58577.0	26.03	0.70	59119	0.94	0.40	59253	59237.0	4.74	59285	59242.6	10.08	0.121

Table 4. *Results for the 818-node network*

Problem	CPLEX			CGA				MS_Heuristic			GRASP			CS			
	Solution	Gap	Time	Best	Average	Time	Dev.	Solution	Time	Dev.	Best	Average	Time	Best	Average	Time	Dev.
818_10_0_0_95	**21460**	0	2957.7	21455	21449.32	43.65	0.02	21429	4.94	0.14	21459	21458.3	6.30	**21460**	21459.9	11.23	0.000
818_10_0_1_95	**35360**	0	3404.1	35356	35346.02	50.45	0.01	35339	13.05	0.06	**35360**	35360.0	6.65	**35360**	35360.0	11.54	0.000
818_10_0_2_95	**45390**	0	3320.1	45387	45377.4	49.37	0.01	45375	12.09	0.03	45389	45388.2	6.97	**45390**	45390.0	12.17	0.000
818_10_1_48_90	**26920**	0	3004.2	26915	26901	48.35	0.02	26907	11.38	0.05	26918	26918.0	6.10	**26920**	26920.0	11.32	0.000
818_10_1_49_90	**28330**	0	3303.5	28320	28297.98	47.78	0.04	28309	8.63	0.07	28329	28328.6	6.67	**28330**	28330.0	12.23	0.000
818_10_1_50_90	**29680**	0	2780.8	29678	29673.76	47.81	0.01	29661	18.28	0.06	**29680**	29679.4	6.42	**29680**	29680.0	11.33	0.000
818_20_0_0_95	**42920**	0	4579.9	42870	42817.66	76.20	0.12	42793	25.67	0.30	42903	42900.0	14.51	**42920**	42918.9	54.84	0.000
818_20_0_1_95	**70720**	0	6392.9	70653	70557.06	80.42	0.09	70644	37.03	0.11	70717	70715.1	16.90	**70720**	70719.2	62.91	0.000
818_20_0_2_95	**90780**	0	7098.1	90747	90672.9	84.99	0.04	90730	75.19	0.06	90772	90769.5	19.18	**90780**	90779.6	66.80	0.000
818_20_0_0_85	**74360**	0	6993.7	74341	74279.42	90.04	0.03	74313	75.05	0.06	74353	74352.1	16.27	**74360**	74359.8	66.81	0.000
818_20_0_1_85	100989	435.00	10800.0	101955	101898.82	89.19	-0.96	101933	76.06	-0.93	101996	101991.9	19.24	**102000**	101998.9	67.36	-1.001
818_20_0_2_85	119405	352.48	10800.0	119445	119306.4	89.18	-0.03	119397	105.48	0.01	119468	119466.0	21.26	**119480**	119477.6	74.36	-0.063
818_20_1_40_85	55398	0.004	10800.0	55341	55235.68	88.33	0.10	55325	66.11	0.13	55396	55395.3	15.93	**55400**	55399.0	61.04	-0.004
818_20_1_41_85	58719	0.002	10800.0	58706	58677.58	87.62	0.02	58637	69.63	0.14	58711	58709.2	15.24	**58720**	58719.9	57.31	-0.002
818_20_1_42_85	61818	774.00	10800.0	61839	61719.7	90.43	-0.03	61814	71.81	0.01	61894	61892.2	16.16	**61900**	61898.8	58.71	-0.133
818_20_1_48_90	**53840**	0	5565.0	53814	53762.18	87.87	0.05	53728	22.41	0.21	53827	53825.7	15.35	**53840**	53839.6	59.56	0.000
818_20_1_49_90	**56660**	0	5254.6	56605	56465.32	88.71	0.10	56602	34.34	0.10	56653	56651.6	16.11	**56660**	56660.0	67.42	0.000
818_20_1_50_90	**59360**	0	4919.3	59336	59279.8	87.04	0.04	59269	39.84	0.15	59352	59350.2	15.02	**59360**	59359.9	58.48	0.000
818_50_0_0_85	185876	0.013	10800.0	184428	184153.6	177.34	0.78	185426	756.72	0.24	185779	185755.8	50.62	**185880**	185775.6	364.20	-0.002
818_50_0_1_85	251029	115.23	10800.0	253438	252884.6	171.86	-0.96	254509	730.20	-1.39	254860	254836.6	58.37	**254985**	254905.0	387.37	-1.576
818_50_0_2_85	292912	84.45	10800.0	296763	296182.8	171.74	-1.31	298217	757.73	-1.81	298547	298511.2	63.58	**298582**	298517.6	392.29	-1.936
818_50_1_48_90	**134598**	0.001	10800.0	134079	133999.4	177.67	0.39	134088	596.11	0.38	134474	134444.5	43.99	**134598**	134561.6	335.00	0.000
818_50_1_49_90	**141648**	0.001	10800.0	140439	140143.4	174.96	0.85	141281	571.17	0.26	141548	141527.5	46.32	141586	141532.8	356.80	0.044
818_50_1_50_90	**148398**	0.001	10800.0	147202	147123.8	173.96	0.81	147931	667.67	0.31	148280	148253.1	46.62	148383	148321.9	337.4	0.010

References

Brotcorne L., Laporte G. and Semet F. (2003). Ambulance Location and relocation models. European Journal of Operational Research, 147, pp. 451-463.

Chaves A. A. and Lorena L. A. N. (2005). Hybrid algorithms with detection of promising areas for the prize collecting traveling salesman problem. Fifth international conference on hybrid intelligent systems (HIS'05), pp. 49-54.

Chung C.H. (1986). Recent applications of the Maximal Covering Location Problem (MCLP) model. Journal of the Operational Research Society. 37, pp. 735-746.

Church R.L. and ReVelle C. (1974). Maximal covering location problem. Papers of the Regional Science Association, 32, pp. 101-118.

Corrêa F.A. and Lorena L.A.N. (2006). Using the Constructive Genetic Algorithm for Solving the Probabilistic Maximal Covering Location-Allocation Problem. I Workshop on Computational Intelligence/SBRN. Available at http://www.lac.inpe.br/~lorena/correa/Correa_Lorena_Wci_2006.pdf.

Current J. R. and O'Kelly M. (1981). Locating emergency warning sirens. Decision Sciences, 23, pp. 221-234.

Daskin M. S. (1995). Network and discrete location: models, algorithms and applications. John Wiley & Sons, New York.

Eaton D., Hector M., Sanchez V., Latingua R. and Morgan J. (1986). Determining ambulance deployment in Santo Domingo, Dominican Republic. Journal of the Operational Research Society, 37, pp. 113-126.

Feo T. and Resende M. (1995). Greedy randomized adaptive search procedures. Journal of Global Optimization, 6, pp. 109–133.

Galvão R.D. (2004). Uncapacitated facility location problems: contributions. Pesquisa Operacional, 24: 7-38.

Galvão R. D. and ReVelle C. S. (1996). A Lagrangean heuristic for the maximal covering location problem. European Journal of Operational Research, 88, pp. 114-123.

Glover F. (1996). Tabu search and adaptive memory programming: Advances, applications and challenges. Interfaces in Computer Science and Operations Research, pp. 1–75.

Hale T. S. and Moberg C. R. (2003). Location science review. Annals of Operations Research, 123, pp. 21-35.

Hougland E. S. and Stephens N. T. (1976). Air pollulant monitor sitting by analytical techniques. Journal of the Air Pollution Control Association, 26, pp. 52-53.

ILOG CPLEX 10.0: User's Manual, France, 2006.

Larson R. C. and Odoni A. R. (1981). Urban operations research, Prentice Hall, Englewood Cliffs, N.J.

Lorena L. A. N. and Furtado J. C. (2001). Constructive genetic algorithm for clustering problems. Evolutionary Computation, 9(3), pp. 309-327.

Lorena L. A. N. and Pereira M. A. (2002). A lagrangean/surrogate heuristic for the maximal covering location problem using Hillsman´s edition, International Journal of Industrial Engineering, 9, pp. 57-67.

Marianov V. and Serra D. (1998). Probabilistic maximal covering location-allocation models for congested systems. Journal of Regional Science, 38(3): 401-424.

Marianov V. and Serra D. (2001). Hierarchical location-allocation models for congested systems. European Journal of Operational Research, 135: 195-208.

Moore G. C. and ReVelle C.S. (1982). The Hierarchical Service Location Problem, Management Science, 28 (7), pp. 775-780.

Oliveira A. C. M. and Lorena L. A. N. (2004). Detecting-promising areas by evolutionary clustering search. Bazzan, A. L. C. and Labidi, S. (Eds.) Springer Lecture Notes in Artificial Intelligence Series vol. 3171, pp. 385-394.

Oliveira A. C. M. and Lorena L. A. N. (2005). Population training heuristics. Gottlieb, J. and Raidl, G. (Eds.) Springer Lecture Notes in Computer Science Series Vol. 3448, pp. 166-176.

Oliveira A. C. M. and Lorena L. A. N. (2007). Hybrid Evolutionary Algorithms and Clustering Search. Crina Grosan, Ajith Abraham and Hisao Ishibuchi (Eds.) Springer SCI Series, vol. 75, pp. 81–102.

Pereira M. A., Lorena L. A. N. and Senne E. L. F. (2007). A column generation approach for the maximal covering location problem. International Transactions in Operations Research, v. 14, p. 349-364.

Pirkul H. and Schilling D. A. (1991). The maximal covering location problem with capacities on total workload. Management Science, 37(2), pp. 233-248.

Serra D. and Marianov V. (2004). New trends in public facility location modeling. Universitat Pompeu Fabra Economics and Business Working Paper 755. Available at http://www.econ.upf.edu/docs/papers/downloads/755.pdf.