Clustering Search Algorithm for the Capacitated Centred Clustering Problem

Antonio Augusto Chaves, Luiz Antonio Nogueira Lorena*

LAC – Laboratory of Computing and Applied Mathematics, INPE – National Institute for Space Research

12227-010 São José dos Campos – SP, Brazil.

Abstract

The Capacitated Centred Clustering Problem (CCCP) consists in partitioning a set of *n* points into *p* disjoint clusters with a known capacity. Each cluster is specified by a centroid. The objective is to minimize the total dissimilarity within each cluster, such that a given capacity limit of the cluster is not exceeded. This paper presents a solution procedure for the CCCP, using the hybrid metaheuristic Clustering Search (CS), whose main idea is to identify promising areas of the search space by generating solutions through a metaheuristic and clustering them into groups that are then further explored with local search heuristics. Computational results in test problems of the literature show that the CS found a significant number of new best known solutions in reasonable computational times.

Keywords: Clustering Problems, Clustering Search Algorithm, Hybrid Metaheuristics.

* Corresponding author.
e-mail address: lorena@lac.inpe.br fax number: +55-012-39456375 (L.A.N. Lorena).

1. Introduction

The location of facilities is a central problem for strategic decisions. Many applications have been explored in areas such as telecommunication, industrial transportation and distribution, with applications on location of off-shore platforms for oil exploration, garbage collection zones, and others. One of the most well-known facility-location problems is the *p*-Median Problem. This problem consists of locating p facilities in a given space (e.g., Euclidean space) which satisfy n demand points in such a way that the total sum of distances between each demand point and its nearest facility is minimized [5].

Another classical location problem is the Capacitated p-Median Problem (CPMP), which have various applications in many practical situations. It can be described as follows: given a set of n points (customers), each of them with a known demand, the problem consists of finding p medians and assign each point to exactly one median such that the total distance of assigned points to their corresponding medians is minimized, and the capacity limit on the medians may not be exceeded [16].

This paper presents a hybrid heuristic approach to solve the Capacitated Centred Clustering Problem (CCCP). The CCCP is a generalization of the CPMP, which can be viewed as the problem of defining a set of p clusters with limited capacity and minimum dissimilarity between the formed clusters, where each cluster has a centroid located at the geometric centre of its points and covers all the demands of a set of n points. The main difference to the CPMP, is that clusters are centred at the "average" of their points' co-ordinates, where for the CPMP, the clusters are centred by their medians.

The following formulation is proposed by Negreiros and Palhano [10]:

$$(CCCP) \quad \min \quad \sum_{i \in I} \sum_{j \in J} \left\| a_i - \bar{x}_j \right\|^2 y_{ij} \tag{1}$$

Subject to,

$$\sum_{j \in J} y_{ij} = 1 \qquad \forall\, i \in I \tag{2}$$

$$\sum_{i \in I} y_{ij} = n_j \qquad \forall j \in J \tag{3}$$

$$\sum_{i \in I} a_i y_{ij} \leq n_j \bar{x}_j \qquad \forall j \in J \tag{4}$$

$$\sum_{i \in I} q_i y_{ij} \leq Q_j \tag{5}$$

$$\bar{x}_j \in \Re^l,\ n_j \in N,\ y_{ij} \in \{0, 1\} \qquad \forall i \in I, \forall j \in J \tag{6}$$

Where,

- I is the set of demand points;
- J is the set of clusters, with $|J| = p$;
- a_i is the geometric position of the point i;
- $\bar{x}_j$ is the geometric position of the centroid of a cluster j;
- $y_{ij} = 1$, if the point i is assigned to cluster j, and $y_{ij} = 0$ otherwise;
- n_j is the number of points in the cluster j;
- q_i is the demand of the point i;
- Q_j is the capacity of each cluster j;

In the above model, the objective function (1) minimizes the total distance between each point and the centroid of the cluster that it is allocated. Notice that the geometric position of the centroid depends on the points that compose the cluster, so, the position of the centroid is an unknown parameter a priori. Constraints (2) impose that each point is allocated to exactly one cluster. Constraints (3) give the number of points in each cluster. Constraints (4) locate the centroid of each cluster at its geometric center. Constraint (5) imposes that a total cluster

capacity must be respected. Constraint (6) defines the decision variables, and the upper limits to the number of individuals per group.

The CCCP is known to be NP-hard [10]. Besides, the non-linearity of the objective function gets hard the application of exact methods to solve the CCCP. So, the problem is usually solved through heuristic approaches.

In this paper, the CCCP is solved using a hybrid metaheuristic known as Clustering Search (CS). The CS, proposed by Oliveira and Lorena [14, 15], consists in detecting promising areas (clusters) of the search space using a metaheuristic that generates solutions to be clustered. These promising areas should be explored through local search methods as soon as they are discovered. The metaheuristic Simulated Annealing [11] is used to generate the solutions for the clustering process.

The remainder of the paper is organized as follows. Section 2 reviews previous works about CCCP and similar problems. Section 3 describes the hybrid metaheuristic Clustering Search (CS) that was used in this paper and section 4 present the CS algorithm applied to CCCP. Section 5 presents the computational results and section 6 concludes the paper.

2. Literature Review

The CCCP was recently introduced by Negreiros e Palhano [10]. The authors presented a two-phase polynomial heuristic algorithm, where the first phase uses the Forgy algorithm [8] to build an initial solution oriented by a *log*-polynomial algorithm using structured geometric balanced *q*-trees. The second phase is a refinement of the Variable Neighborhood Search (VNS).

The CCCP is not intensively studied like other classical location problems. The CPMP have various heuristics and metaheuristics proposed in the literature. Osman and Christofides [16] propose a Simulated Annealing and Tabu Search method. Lorena and Senne [13] explore

local search heuristics based on location-allocation procedures and Lorena and Senne [12] use column generation to CPMP. Diaz and Fernández [6] examine a hybrid scatter search and path-relinking method and Scheuerer and Wendolsky [18] a scatter search method. Recently, Fleszar and Hindi [7] propose a variable neighborhood search heuristic and Chaves, Correa and Lorena [4] present a CS algorithm to solve the CPMP.

Another similar problem is the capacitated multisource Weber problem, also referred to as the capacitated continuous location-allocation problem [19]. The continuous problem requires the generation of a given number m of capacitated facilities in the continuous space and the allocation of n customers or fixed points to each one. The facility can be located anywhere in the continuous space. The aim is to satisfy the demand of the customers and minimize the total transportation (or service) cost. Brimberg et al. [2] presents a survey on continuous location-allocation problem, giving an overview of exact solution methods and approximated methods (heuristics). Recently, Aras, Orbay and Altinel [1] propose three heuristic methods based on a new mixed integer linear programming formulation of the problem.

3. Clustering Search algorithm

The Clustering Search (CS) algorithm generalizes the Evolutionary Clustering Search (ECS), proposed by Oliveira and Lorena [14, 15], that employs clustering for detecting promising areas of the search space. It is particularly interesting to find out such areas as soon as possible to change the search strategy over them. An area can be seen as a search subspace defined by a neighborhood relationship in the metaheuristic coding space. In the ECS, a clustering process is executed simultaneously to an evolutionary algorithm, identifying groups of individuals that deserve special attention. In the CS, the evolutionary algorithm is substituted by distinct metaheuristics, such as Simulated Annealing, GRASP, Tabu Search and others.

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

The center of the cluster c is a solution that represents the cluster, identifying its location inside the search space. Initially, the centers 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. The search strategy β 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:

- a search metaheuristic (SM);
- an iterative clustering (IC);
- an analyzer module (AM);
- a local search module (LS).

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

The search metaheuristic (SM) component 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 simultaneously maintained to represent these solutions. This entire process works like an infinite loop, in which solutions are generated along the iterations.

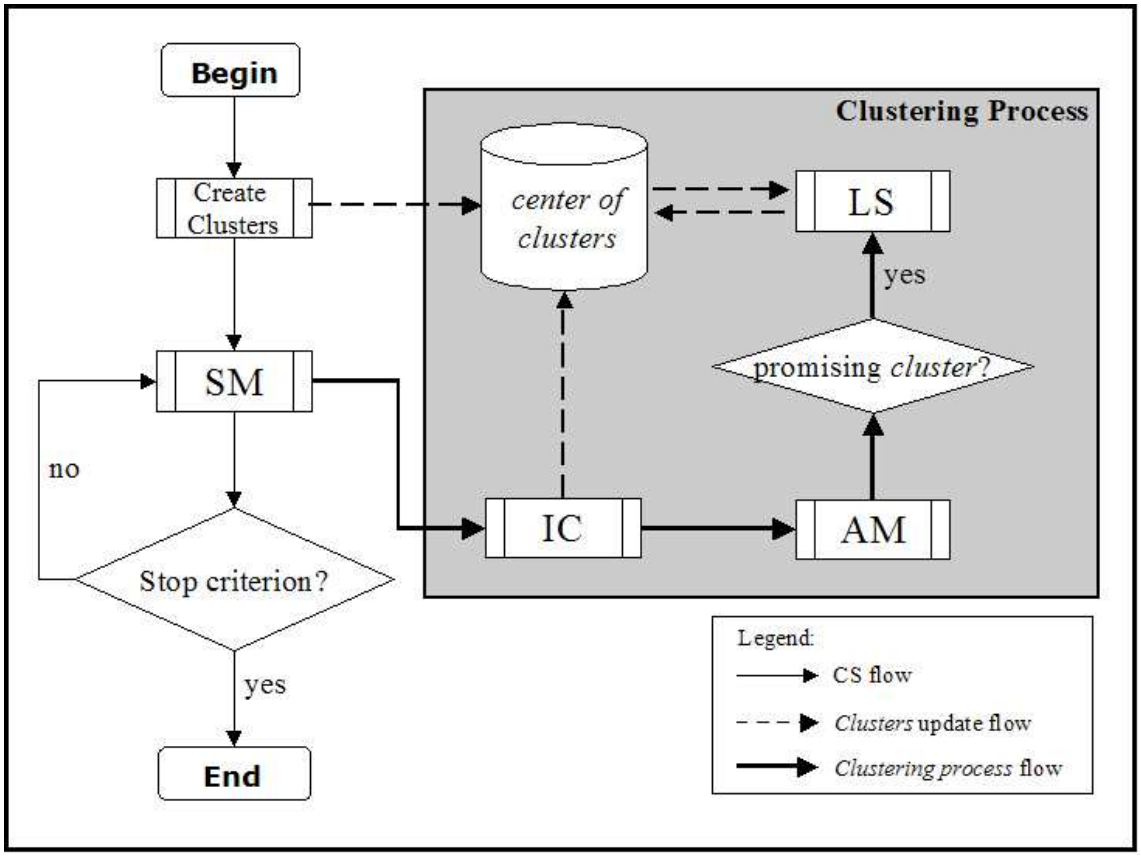

Fig. 1. Diagram for the CS algorithm.

The iterative clustering (IC) component 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 by SM. A maximum number of clusters *NC* is an upper bound value that prevents creation of an unlimited number of clusters. A distance metric must be defined, a priori, allowing a similarity measure for the clustering process.

The analyzer module (AM) component provides an analysis of each cluster, at regular intervals, indicating a probable promising cluster. A cluster density, δ, 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 local search (LS) component is a local search module that provides the exploitation of a supposed promising search area framed by the cluster. This process is executed each time AM finds a promising cluster and the local search is applied on the center of the cluster. LS can be considered as the search strategy β associated with the cluster, i.e., a problem-specific local search to be applied into the cluster.

4. CS algorithm for CCCP

The CS approach for the CCCP uses the computational structure of the CPMP proposed by Chaves, Correa and Lorena [3], which is represented by two vectors: the first vector represents the chosen medians and the second vector represents the point/median allocation. The Figure 2 shows an example of one solution with 10 points and 3 medians.

Medians	3	5	7							
Point	1	2	3	4	5	6	7	8	9	10
Median	5	3	3	7	5	7	7	5	3	7

Fig. 2. An example of representation.

The evaluation of the objective function for the CCCP has a high computational cost as the centroids are unknown a priori. The CS avoids calculating this objective function every time; otherwise the computational times will become impracticable. So, in this paper we used the same CS approach developed by CPMP [3] and only the LS component find the centroids and calculate the objective function of the CCCP with information collected from the CPMP solution.

The SM component, responsible for generating solutions to clustering process, was the Simulated Annealing (SA) metaheuristic [11], which is capable to generate a large number of different solutions for this process. The application details of the SA and the other components of the CS are now described, clarifying this approach.

4.1 Simulated Annealing

The component SM generates solutions to clustering process with a Simulated Annealing (SA) algorithm [11]. The SA needs to evaluate the objective function of all generated neighbor and has a high computational cost for direct application to CCCP. We then use in this paper the SA to solve the corresponding CPMP and the LS components to solve the CCCP.

The SA algorithm starts from a random initial solution which is obtained choosing randomly the medians and assigning the points to the closer median that not exceed the cluster capacity. The next step follows the traditional SA algorithm schema. Given a temperature *T*, the algorithm randomly selects one of the moves to a neighborhood and computes the variation of the objective function. If it improves the current solution the move is accepted, otherwise there is a probability of acceptance that is lower in low temperatures.

Four different moves have been defined to compose distinct neighborhoods from a solution *s*, named N^1, N^2, N^3 and N^4. N^1 is obtained swapping the allocation of two points of different medians. N^2 is obtained swapping a median with an assigned point to it. N^3 is obtained dropping a point allocated to a median and allocating it to another median. Finally N^4 is obtained swapping a median with any other non median point.

The control parameters of the procedure are the rate of cooling or decrement factor α, the number of iterations for a fixed temperature (*SAmax*) and the initial temperature *To*. In this paper, we use $\alpha = 0.95$, *SAmax* = 1000 and *To* = 1000000.

4.2 The Clustering Process

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 = 20) is defined a priori. The i^{th} cluster has its own center c_i and a radius r, like the other clusters.

Solutions generated by SA 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), causing some kind of disturbance on it. In this paper, the metric distance was the number of points assigned to different medians between the SA and the center of the cluster solutions. The dissimilarity increases when there are a large number of different medians between the SA and the cluster solution center.

The disturbance is an assimilation process, in which the center of the cluster is updated by the newly generated solution. In this paper, this process is done by the path-relinking method [9] that generates several points (solutions) along the path that connects the solution generated by the SA and the one in the cluster center. The assimilation process is useful to apply intensification and diversification strategies inside the clusters. Again, in path-relinking we preferred to solve the CPMP because of the high computational cost for calculate the objective function of the CCCP. The new center c_i is the best-evaluated solution obtained in the path.

Path-relinking starts from two solutions. The first is the closest cluster center c_i ($s_{initial}$). The second is the solution that comes from the SM component (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 $\Delta(s_{initial}, s_{guide}) = \varnothing$. 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}, and to change the allocation of the points regarding the new median. Figure 3 illustrates the pseudo-code of the path-relinking and Figure 4 shows an example for a CCCP instance.

```
procedure Path-relinking(s_initial , s_guide)
1.  compute symmetric difference Δ(s_initial , s_guide)
2.  f* = ∞
3.  s = s_initial
4.  while (Δ(s_initial , s_guide) ≠∅)
5.        analyze all moves m ∈ Δ(s_initial , s_guide)
6.        find the best move m*
7.        s = s ⊕ m*
8.        update the available moves Δ(s⊕m*, s_guide) = Δ(s, s_guide)\{m*}
9.        if f(s) < f*
10.             f* = f(s)
11.             s* = s
12.       end-if
13. end-while
14. return s*
end procedure
```

Fig. 3. Path-relinking algorithm

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 certain density δ_i,

$$\delta_i \geq PD.\frac{NS}{|Clus|} \tag{6}$$

where, *NS* is the number of solutions generated in the interval of analysis of the clusters, |*Clus*| is the number of clusters, 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.

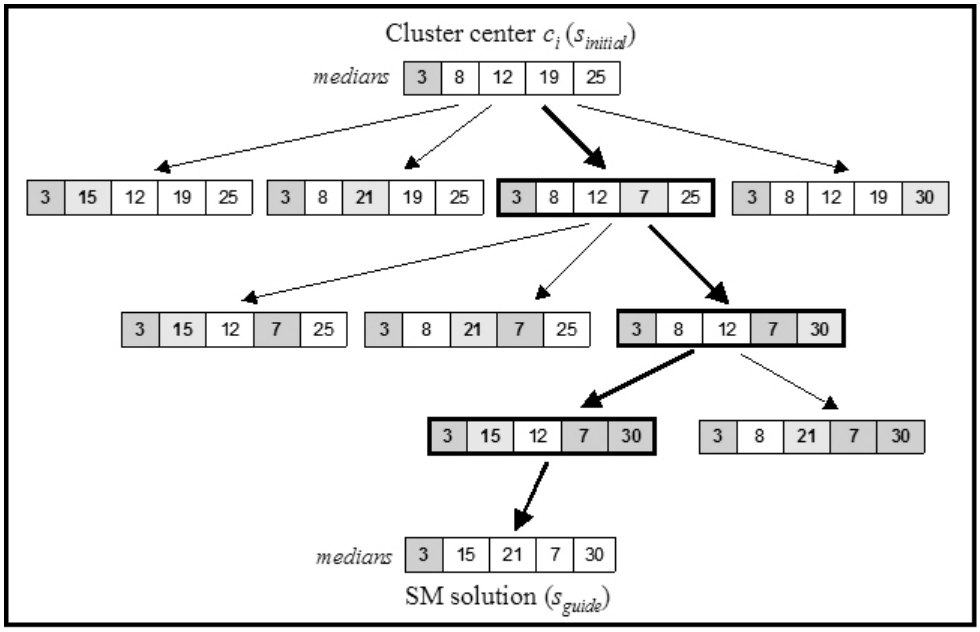

Fig. 4. Path-relinking moves

The LS component is activated when the AM discover a promising cluster. The LS implementation uses the Location-Allocation heuristic [13], which seeks to improve the center of the promising cluster. This heuristic is applied in the solution for the CPMP and for each new generated solution is calculated the coordinates of the centroids and the value of the objective function of CCCP. The choice of the Location-Allocation heuristic was based on the observation that the solution in the cluster center of CS have *p* medians and their allocated points, and, this solution can be improved by searching for a new median, swapping the current median by a non-median point assigned to it, and reallocating the points. We consider two steps for reallocating the points. The first one is to examine the points that were allocated

to the current median and reallocate to the closest one. The second step is to examine the points assigned to the others medians and calculates the *saving* of moving of them to the new one, and if it improves the solution the point is reallocated to the new median. If the solution is improved, the process can be repeated until no more improvements occur.

The whole CS algorithm pseudo-code is presented in Figure 5.

```
procedure CS
1.  Create initial clusters of the CS randomly
2.  Create initial solution(sol)
3.  { SM component }
4.  IterT = 0; α = 0,95;   T = T0
5.  while (T > 0.0001)
6.      while (IterT < SAmax)
7.          IterT = IterT + 1
8.          Generate at random sol' ∈ N^k(sol)
9.          Δ = f(sol') – f(sol)
10.         if (Δ < 0)
11.             sol = sol'
12.         else
13.             let x ∈ [0,1]
14.             if (x < e^(-Δ/T))
15.                 sol = sol'
16.     end-while
17.     T = T * α
18.     IterT = 0
19.     { IC component }
20.     calculate the distance of the solution SA (sol) and the clusters of the CS
21.     insert the solution in the most similar cluster (ci)
22.     apply the assimilation process – path-relinking(sol, ci)
23.     { AM component }
24.     verify if the cluster can be considered promising. If so, the LS component
        is applied to it
25.     { LS component }
26.     apply the local heuristic to the promising cluster
27. end-while
end-procedure
```

Fig. 5. Clustering Search (CS) algorithm

5. Computational Results

The CS was coded in C++ and the computational tests executed on a 3 GHz Pentium 4. Four problem sets are used in this tests: two sets composed of real data collected at the São José dos Campos city introduced by Lorena and Senne [12], that contains 6 problem instances named "*SJC*" and 5 problem instances named "*p3038*", and two sets of instances introduced by Negreiros and Palhano [10], that contains 7 problem instances named "*TA*" and 7 problem instances named "*Doni*", which are based in customers of a food distributor that operates in the metropolitan area of the Fortaleza city. Those instances and the best results of the CS for the CCCP are available at http://www.lac.inpe.br/~lorena/instancias.html.

The parameters' values for the CS approach were adjusted through several executions. The following parameters obtained the best results:

- number of solutions generated at each analysis of the clusters $NS = 200$;
- maximum number of clusters $NC = 20$;
- density pressure $PD = 2.5$;

Tables 1-4 give the results for the CCCP. The entries in the tables are:

- number of demand points (*n*);
- number of clusters (*p*);
- the best solutions (*best_sol*) found by the VNS in Negreiros e Palhano [10];
- the best solution (*best_sol*), the Deviation (*dev*), that reflects the relative error of the average solution and the best solution found by the CS algorithm, and it is calculated by (average solution of CS – best solution of CS) / (best solution of CS) * 100, the average running time to find the best solution during the CS execution (*best_time*) in seconds and the average total running time of the CS execution (*Time*) in seconds; and

– the best solution (*best_sol*), the Deviation (*dev*), calculated by (average solution of SA – best solution of SA) / (best solution of SA) * 100, and the average total running time of the SA execution (*Time*).

The best solutions found (*best_sol*), the deviations (*dev*) and the averages running times were considered to compare the approaches. The values in boldface show the best objective function values for each instance, the best CS results are followed by an asterisk (*) and the best VNS [10] results are followed by two asterisks (**). Each instance has been run for 10 replications. The comparison of computation times between CS and Negreiros and Palhano's algorithm was not possible as they were tested on different machines and also to the fact that this information is not completely clear in [10].

Table 1 gives the results for the *TA* instances. One can see that the CS approach finds the best-known solutions for all instances. For instances with 50, 70, 90 and 100 points, it finds a solution better than the previously best known. The CS algorithm achieves very good solutions within much shorter computation times. The SA, without the clustering process, has worse results than CS in quality of solutions. The CS algorithm was very robust producing zero deviations.

Table 1

CCCP: *TA* instances. Times in seconds

		VNS [10]	**CS**				**SA**		
n	***p***	***best_sol***	***best_sol***	***dev***	***best_time***	***Time***	***best_sol***	***dev***	***Time***
25	*5*	**1251.44**	**1251.44**	0.00	0.30	2.86	1273.46	0.00	2.43
50	*5*	4476.12	**4474.52***	0.00	1.06	5.12	4478.15	0.84	3.62
60	*5*	**5356.58**	**5356.58**	0.00	0.64	6.13	5370.05	0.17	3.98
70	*5*	6241.55	**6240.67***	0.00	0.57	7.38	6267.89	0.00	4.73
80	*7*	**5730.28**	**5730.28**	0.00	4.00	10.04	5780.55	0.07	5.88
90	*4*	9103.21	**9069.85***	0.00	0.67	9.61	9069.85	0.00	5.33
100	*6*	8122.67	**8102.04***	0.00	3.81	12.68	8153.64	0.03	6.75

Table 2 reports the results of computational experiments with the *SJC* instances. In this case, the CS algorithm again has better results in all tests, with new best known solutions for all instances. The running times of the CS algorithm were very competitive and the results of the CS were better than the SA without the clustering process.

Table 2

CCCP: *SJC* instances. Times in seconds

		VNS [10]	**CS**				**SA**		
n	***p***	***best_sol***	***best_sol***	***dev***	***best_time***	***Time***	***best_sol***	***dev***	***Time***
100	*10*	17696.53	**17359.75***	0.02	7.10	14.66	17363.47	0.31	7.99
200	*15*	33423.84	**33181.65***	0.00	15.70	49.51	33458.40	0.39	17.05
300	*25*	47985.29	**45366.35***	0.06	78.17	144.40	46847.61	1.14	35.62
300	*30*	-	**40695.46***	0.07	100.79	184.24	41450.63	1.57	39.99
402	*30*	66689.96	**61944.85***	0.13	106.23	262.59	64981.66	1.69	54.21
402	*40*	-	**52214.55***	0.14	229.84	441.13	53735.96	2.06	67.66

The results of Table 3 refer to *p3038* instances. The results for these instances were much better than the solutions found in [10], albeit the large processing times. The large number of clusters ($p \geq 600$) is the probable cause of the high computational cost, as for those instances the path-relinking is responsible for more than 40% of the running time of CS. The results of the SA were worse than the CS, but they are close of the results found in [10].

Table 3

CCCP: *p3038* instances. Times in seconds

		VNS [10]	**CS**				**SA**		
n	***p***	***best_sol***	***best_sol***	***dev***	***best_time***	***Time***	***best_sol***	***dev***	***Time***
3038	*600*	192024.83	**129194.11***	0.59	27634.66	38116.25	194541.04	1.37	1263.08
3038	*700*	176731.07	**117295.47***	0.87	18607.20	43231.23	177148.48	2.24	4055.09
3038	*800*	184502.38	**109532.61***	1.25	20256.49	52908.44	169045.19	1.31	1676.25
3038	*900*	176781.51	**102458.93***	1.87	30307.64	61171.45	163751.56	0.45	1869.50
3038	*1000*	159139.89	**97771.67***	2.97	36588.77	68466.73	153564.41	2.94	1946.65

Table 4 shows the results obtained for the *Doni* instances. In this case, the CS algorithm fails to find the best-known solution for 2 instances, but the CS found 5 new best known solutions. The running times of the CS algorithm were very competitive and the results of the CS were better than the SA.

Table 4

CCCP: *Doni* instances. Times in seconds

		VNS [10]	**CS**				**SA**		
n	***p***	***best_sol***	***best_sol***	***dev***	***best_time***	***Time***	***best_sol***	***dev***	***Time***
1000	*6*	**3021.41****	3022.26	0.03	133.26	468.22	3138.67	2.75	38.00
2000	*6*	**6080.70****	6372.81	0.01	461.47	879.80	6985.30	4.22	98.95
3000	*8*	8769.05	**8446.08***	0.12	890.94	2176.12	9653.27	7.02	167.65
4000	*10*	11516.14	**10854.48***	0.81	3547.81	8367.95	13328.16	3.72	253.97
5000	*12*	11635.18	**11134.94***	0.26	4209.36	11486.72	13920.49	3.01	366.65
10000	*23*	18443.50	**15928.38***	1.80	5122.72	17851.72	29102.49	1.56	505.57
13221	*30*	23478.79	**20291.52***	0.64	7929.99	19511.34	29484.66	10.46	861.44

6. Conclusions

This paper has presented a solution for the Capacitated Centred Clustering Problem (CCCP) using the Clustering Search (CS) algorithm that uses the concept of hybrid metaheuristics, combining metaheuristics with a local search in a clustering process. The CS has been applied with success in some combinatorial optimization problems, such as the pattern sequencing problem [14], the prize collecting traveling salesman problem [3], the capacitated p-median problem [4], flowshop scheduling [17], and others.

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 because of time consumption, mainly when the heuristic has a high computational cost. The CS detects the promising regions in the search space during the solution generation process and applies the

local search heuristics only in these regions, i.e., to detect promising regions becomes an interesting alternative preventing the indiscriminate application of such heuristics.

This paper reports results of different classes of instances to the CCCP found by the CS, Simulated Annealing (SA) without the clustering process and the VNS proposed by [10]. The CS got better results than SA and VNS [10] in nearly all instances producing several new best solutions for those instances. The results also show that the CS approach is competitive for solving the CCCP in reasonable computational times.

One important difference between the CS and the VNS [10] is the fact of the VNS directly solves the CCCP whilst the CS solves the CPMP and calculates the objective function of the CCCP only when the LS component is applied, which decreased the CS computational times.

Further studies can be done analyzing others metaheuristics to generate solutions for the clustering process of the CS, such as the Ant Colony System, Tabu Search and Genetic Algorithm, and by implementing new local search heuristics for the CCCP.

Acknowledgments

The authors acknowledge the useful comments and suggestions of an anonymous referee and the CNPq by a partial research support.

References

1. Aras N, Orbay M, Altinel IK. Efficient heuristics for the rectilinear distance capacitated multi-facilityWeber problem. Journal of the Operational Research Society, 2008; 59: 64 – 79.
2. Brimberg J, Hansen P, Mladenovic N, Salhi S. A Survey of Solution Methods for the Continuous Location-Allocation Problem. International Journal of Operations Research, to appear

3. Chaves AA, Lorena LAN. Hybrid algorithms with detection of promising areas for the prize collecting traveling salesman problem. International Conference on Hybrid Intelligent Systems, IEEE Computer Society, 2005; 49 – 54.
4. Chaves AA, Correa FA, Lorena LAN. Clustering Search Heuristic for the Capacitated p-median Problem Springer Advances in Software Computing Series, 2007; 44: 136 – 143.
5. Christofides N, Beasley JE. A tree search algorithm for the p-median problem, European Journal of Operational Research, 1981; 10: 196 – 204.
6. Diaz JA, Fernandez E. Hybrid scatter search and path relinking for the capacitated p-median problem. European Journal of Operational Research, 2006; 169: 570 – 585.
7. Fleszar K, Hindi DS. An effective VNS for the capacitated p-median problem. European Journal of Operational Research, 2007, doi:10.1016/j.ejor.2006.12.055.
8. Forgy EW. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. Biometrics, 1965; 21(3): 768.
9. Glover F. Tabu search and adaptive memory programming: Advances, applications and challenges. Interfaces in Computer Science and Operations Research , 1996; 1–75.
10. Negreiros MJN, Palhano AWC. The Capacitated Centred Clustering Problem. Computers and Operations Research, 2006; 33 (6): 1639 – 1663.
11. Kirkpatrick S, Gelatt CD, Vecchi MP. Optimization by Simulated Annealing, Science, 1983; 220 (4598): 671 – 680.
12. Lorena LAN, Senne ELF. A column generation approach to capacitated p-median problems. Computers and Operational Research, 1994; 31: 863 – 876.
13. Lorena LAN, Senne ELF. Local search heuristics for capacitated p-median problems, Networks and Spatial Economics, 2003; 3 (4): 407 – 419.

14. Oliveira ACM, Lorena LAN. Detecting promising areas by evolutionary clustering search.. Advances in Artificial Intelligence. Springer Lecture Notes in Artificial Intelligence Series, 2004; 385 – 394.
15. Oliveira ACM, Lorena LAN. Hybrid evolutionary algorithms and clustering search. In Grosan, C., Abraham, A., Ishibuchi, H., eds.: Hybrid Evolutionary Systems - Studies in Computational Intelligence. Springer SCI Series, 2007; 81 – 102.
16. Osman IH, Christofides N. Capacitated clustering problems by hybrid simulated annealing and tabu search. International Transactions in Operational Research, 1994; 1 (3): 317 – 336.
17. Ribeiro G, Nagano MS, Lorena LAN. Hybrid Evolutionary Algorithm for Flowtime Minimisation in No-Wait Flowshop Scheduling. Springer-Verlag Berlin Heidelberg, 2007; 1099 – 1109.
18. Scheuerer S, Wendolsky R. A scatter search heuristic for the capacitated clustering problem. European Journal of Operational Research, 2006; 169: 533 – 547.
19. Wesolowsky G. The Weber problem: history and perspectives. Location Science, 1993; 1: 5 – 23.