Clustering Search Heuristic for the Capacitated *p*-Median Problem

Antonio Augusto Chaves[1], Francisco de Assis Correa[1], Luiz Antonio N. Lorena[1]

[1] LAC – Laboratory of Computing and Applied Mathematics, INPE – National Institute for Space Research, 12227-010 São José dos Campos – SP, Brazil
{chaves, lorena}@lac.inpe.br; fcorrea@directnet.com.br

Abstract. In this paper we present a hybrid heuristic for the capacitated p-median problem (CPMP). This problem considers a set of *n* points, each of them with a known demand, the objective 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 a capacity limit on the medians may not be exceeded. The purpose of this paper is to present a new hybrid heuristic to solve the CPMP, called Clustering Search (CS), which consists in detecting promising search areas based on clustering. Computational results show that the CS found the best known solutions in all most instances.

Keywords: *p*-Median Problem, Hybrid heuristics, Clustering Search.

1 Introduction

This paper presents a new hybrid heuristic to solve the Capacitated *p*-Median Problem (CPMP). The CPMP is a classical location problem with 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 (centers) 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.

The CPMP also appears under the names Capacitated Clustering Problem, Capacitated Warehouse Location Problem, Sum-of-Stars Clustering Problem and others. Various heuristics and metaheuristics have been proposed for these problems, which are known to be NP-hard [4]. Osman and Christofides [11] propose a simulated annealing and tabu search method. Maniezzo *et al.* [7] present a bionomic algorithm to solve this problem. Lorena and Senne [6] explore local search heuristics based on location-allocation procedures and Lorena and Senne [5] use column generation to CPMP. Diaz and Fernández [2] examine a hybrid scatter search and path-relinking method and Scheuerer and Wendolsky [12] a scatter search method. Recently, Fleszar and Hindi [3] propose a variable neighborhood search heuristic and Osman and Ahmadi [10] investigate a guide construction search metaheuristic based on a periodic local search procedure or a greedy random adaptive construction search procedure (GRASP) to solve the CPMP.

The CPMP considered in this paper is modeled as the following binary integer programming problem:

$$z = \min \sum_{i \in N} \sum_{j \in N} d_{ij} x_{ij} \quad (1)$$

$$\text{subject to} \quad \sum_{j \in N} x_{ij} = 1, \quad \forall i \in N \quad (2)$$

$$\sum_{j \in N} x_{jj} = p \quad (3)$$

$$\sum_{i \in N} q_i x_{ij} \leq Q x_{jj}, \quad \forall j \in N \quad (4)$$

$$x_{ij} \in \{0,1\}, \quad \forall i \in N, \forall j \in N \quad (5)$$

where:

- $N = \{1,...,n\}$ is the index set of points to allocate and also of possible medians, where p medians will be located;
- q_i is the demand of each point and Q the capacity of each possible median;
- d_{ij} is a distance matrix;
- x_{ij} is the allocation matrix, with $x_{ij} = 1$ if point i is allocated to median j, and $x_{ij} = 0$, otherwise; $x_{jj} = 1$ if median j is selected and $x_{jj} = 0$, otherwise.

The objective of the CPMP is expressed in (1). Constraints (2) impose that each point is allocated to exactly one median. Constraint (3) set the number of medians to be located. Constraint (4) imposes that a total median capacity must be respected, and (5) provides the integer conditions.

In this paper, we propose a hybrid heuristic for the CPMP, known by Clustering Search (CS). The CS, proposed by Oliveira and Lorena [9], consists in detecting promising areas 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 remainder of the paper is organized as follows. Section 2 describes the basic ideas and components of CS. Section 3 present the CS applied to CPMP. Section 4 presents the computational results and conclusions are presented in Section 5.

2 Clustering Search

The Clustering Search (CS) generalizes the Evolutionary Clustering Search (ECS) proposed by Oliveira and Lorena [9] 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.

In the ECS, a clustering process is executed simultaneously to an evolutionary algorithm, identifying groups of individuals that deserve special interest. In the CS,

the evolutionary algorithm was substituted by distinct metaheuristics, such as Tabu Search, Variable Neighborhood Search or Simulated Annealing.

The CS attempts to locate promising search areas by framing them by clusters. A cluster can be defined as a tuple $\mathcal{G} = \{c; r; s\}$ 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 c is a solution that represents the cluster, identifying the location of the cluster inside of the search space. The radius r establishes the maximum distance, starting from the center, that a solution can be associated to the cluster. The search strategy s is a systematic search intensification, in which solutions of a cluster interact among themselves along the clustering process, generating new solutions.

The CS is hybrid metaheuristic that consists of four conceptually independent components with different attributions: a search metaheuristic (SM); an iterative clustering (IC); an analyzer module (AM); and a local searcher (LS).

The SM component works as a full-time solution generator. The algorithm is executed independently of the remaining components and must to be able provide the continuous generation of solutions directly to the clustering process. Simultaneously, clusters are kept to represent these solutions. This entire process works like an infinite loop, in which solutions are generated along the iterations.

The IC component aims to gather similar solutions into groups, keeping 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 component provides an analysis of each cluster, in 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, such cluster must be better investigated to accelerate the convergence process on it.

At last, the LS component is a local search module that provides the exploitation of a supposed promising search area, framed by cluster. This process happens after AM finds a promising cluster and the local search is applied on the cluster center. LS can be considered as the particular search strategy s associated with the cluster, i.e., a problem-specific local search to be employed into the cluster.

3 CS for CPMP

A version of CS for CPMP is presented in this section. The application details are now described, clarifying this approach. The component SM, responsible for generating solutions to clustering process, was a Simulated Annealing (SA) metaheuristic [13], which is capable to generate a large number of different solutions for this process. The others components of CS are also explained in the following.

3.1 Simulated Annealing

In the component SM, the metaheuristic used to generate solutions to clustering process is based on the Simulated Annealing (SA) [13]. The algorithm starts from a random initial solution which is obtained choosing randomly the medians and assigned the points to the closer median that not exceed the capacity of it. The next step follows the traditional simulated annealing 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 kinds of neighborhood, named N^1, N^2, N^3 and N^4, from a solution s. N^1 is obtained by swapping the allocation of two points of different medians. N^2 is obtained by swapping a median with an assigned point to it. N^3 is obtained by dropping a point of a median and add in other median. And N^4 is obtained by swapping a median with any other point that is not a median.

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

3.2 The CS application

The IC is the CS's core, working as a classifier, keeping in the system only relevant information, and driving the search intensification in the promising search areas. Initially, all clusters are created randomly ($NC = 20$), the i^{th} cluster has its own center c_i, and a radius r that is identical to the other clusters.

Solutions generated by SA are passed to IC that attempts to group as known solution, according to a distance metric. If the solution is considered sufficiently new, it is kept as a center in a new cluster. Otherwise, redundant solution activates the closest center c_i (cluster center that minimizes the distance metric), causing some kind of perturbation on it. In this paper, the metric distance was the number of points assigned to different medians in the solutions of the SA and the cluster center, and a larger distance imply in more dissimilarity.

Perturbation means an assimilation process, in which the cluster center is updated by the new generated solution. Here, we used the path-relinking method [8], that generates several points (solutions) taken in the path connecting the solution generated by SA and the cluster center. The assimilation process itself is an intensification mechanism inside the clusters. The new center c_i' is the best evaluated solution sampled in the path.

Path-relinking starts from two solutions. The first is the solution that comes from the SA (*initial*). The second is the closest cluster center c_i (*guide*). Starting from the first to the second solution, paths are generated and explored in the search for better solutions. To generate paths, moves are selected by changing one median of the *initial* by one from the *guide*, changing the allocation solution. The best solution in one move is defined as the *new initial*.

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

$$\delta_i \geq PD.\frac{NS}{|Clus|} \quad (6)$$

where, *NS* is the number of solutions generated in the interval of analysis of the clusters, |*Clus*| is the number of cluster, and *PD* is the desirable cluster density beyond the normal density, obtained if *NS* was equally divided to all clusters. In this paper, we use *NS* = 100 and *PD* = 2.

The component LS is activated when the AM discover a promising cluster. The LS uses the Location-Allocation heuristic [6], which seeks to improve the center of the promising cluster. This heuristic is based on the observation that the cluster center 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. 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 calculate the *saving* of moving them to the new one, if it improves the solution the point is reallocated to the new median. If the solution is improved, the process can be repeat until no more improvements occur.

The whole CS pseudo-code is presented in following.

```
Procedure CS
  Create Initial Solution (s)
  Create Initial Clusters
  IterT = 0
  T = To
  while (T > 0.0001)
    while (IterT < SAmax)
      IterT = IterT + 1
      Generate at random s'∈ N^k(s)
      Δ = f(s') – f(s)
      if (Δ <0)
        s = s'
      else
        Let x ∈ [0,1]
        if (x < e^(-Δ/T))
          s = s'
      end-if
      component IC (s')
      component AM (active cluster)
      if (active cluster is promising) then
        component LS (cluster center)
    end-while
    T = T x α
    IterT = 0
  end-while
end-CS.
```

4 Computational Results

The CS was coded in C++ and the computational tests executed on a Pentium IV 3.02 GHz. Two problem sets are used in this tests: a classical set introduced by Osman and Cristofides [11], that contains 10 problems of size $n = 50$ and $p = 5$, and 10 problems of size $n = 100$ and $p = 10$ (these problems are named of *p1* to *p20*), and a set of real data collected at the São José dos Campos city introduced by Lorena and Senne [5], that contains 6 problem instances named "*sjc*". Those instances can be downloaded from or through the OR-Library.

Table 1 gives the obtained results for each instance, comparing the performance of the CS and the SA without the clustering process. Column *best* gives the values of the best known solutions found in literature. The best solutions found by the approaches (*sol**), the time to obtain the best solution (*Time**) in seconds, the average of solutions (*sol*), and the total time (*Time*) in seconds, used to compare the CS and SA. Each instance has been run 10 times. The improvement of the CS in relation to SA is reported in terms of the relative percentage deviation (RPD).

Table 1. Results of the CS

		CS				SA			
ID	best	sol*	Time*	sol	Time	sol*	sol	Time	RPD
p1	713	713	0.02	713.00	2.24	713	721.60	1.56	1.21
p2	740	740	0.01	740.00	2.34	740	740.00	1.59	0.00
p3	751	751	0.08	751.00	2.29	751	751.60	1.46	0.08
p4	651	651	0.03	651.00	2.24	651	651.20	1.52	0.03
p5	664	664	0.03	664.00	2.35	664	664.40	1.51	0.06
p6	778	778	0.01	778.00	2.35	778	778.00	1.57	0.00
p7	787	787	0.01	787.00	2.34	787	788.60	1.55	0.20
p8	820	820	0.08	820.00	2.38	821	825.80	1.60	0.71
p9	715	715	0.02	715.00	2.36	715	717.80	1.46	0.39
p10	829	829	2.09	829.00	2.36	829	833.80	1.58	0.58
p11	1006	1006	7.97	1006.00	35.24	1007	1015.40	10.11	0.93
p12	966	966	0.05	966.00	49.39	968	970.20	12.74	0.43
p13	1026	1026	0.06	1026.00	41.05	1026	1028.80	10.00	0.27
p14	982	982	19.80	982.80	39.14	985	997.20	10.35	1.47
p15	1091	1091	1.30	1091.00	44.98	1092	1102.60	12.15	1.06
p16	954	954	15.94	954.00	36.03	957	960.40	10.37	0.67
p17	1034	1034	19.50	1034.20	48.18	1037	1043.00	12.24	0.85
p18	1043	1043	48.62	1044.60	46.39	1045	1048.20	11.41	0.34
p19	1031	1031	10.17	1031.80	38.56	1034	1042.80	10.09	1.07
p20	1005	1005	5.41	1005.40	48.77	1009	1015.20	12.45	0.97
sjc1	17288.99	17288.99	0.23	17288.99	22.80	17288.99	17343.96	10.43	0.32
sjc2	33270.94	33270.94	13.25	33275.43	120.40	33372.98	33491.75	25.70	0.65
sjc3a	45335.16	45335.16	405.50	45337.34	859.09	46746.68	47110.93	52.51	3.91
sjc3b	40635.90	40635.90	1626.52	40643.67	1649.41	41551.34	41888.14	54.01	3.06
sjc4a	61925.51	61928.72	938.45	62017.51	2601.81	63710.71	64574.92	77.19	4.10
sjc4b	52469.96	52531.27	1402.25	52540.67	7233.59	53789.61	54716.58	92.05	4.14

CS found the best known solutions for most of the instances. Except for *sjc4a* and *sjc4b*. The running times of the CS were very competitive, found better solutions in small times. The CS is very robust, producing average solutions close to the best known ones. The SA, without the clustering process, has worse results than the CS in quality of solutions, but the times are smaller. All SA solutions were improved by CS.

Table 2 present a comparison of the CS solutions with the results of two heuristics that have the best performing in the literature, the first is a scatter search with path-relinking (SS-PR) [12] and the second is a variable neighborhood search (VNS) [3]. Note that CS was very competitive to the SS-PR and VNS, failing in find the best known solution only for two instances. For the *sjc* instances, the total time of CS was smaller than the others heuristics in thee instances.

Table 2. Comparison of the results

		SS-PR		VNS		CS	
ID	best	sol*	Time	sol*	Time	sol*	Time
p1	713	713	6	713	0.17	713	2.24
p2	740	740	6	740	0.05	740	2.34
p3	751	751	6	751	0.19	751	2.29
p4	651	651	6	651	0.11	651	2.24
p5	664	664	6	664	0.27	664	2.35
p6	778	778	6	778	0.11	778	2.35
p7	787	787	6	787	0.31	787	2.34
p8	820	820	6	820	0.92	820	2.38
p9	715	715	6	715	0.13	715	2.36
p10	829	829	6	829	0.75	829	2.36
p11	1006	1006	60	1006	7.91	1006	35.24
p12	966	966	60	966	4.81	966	49.39
p13	1026	1026	60	1026	2.17	1026	41.05
p14	982	982	60	982	10.33	982	39.14
p15	1091	1091	60	1091	10.23	1091	44.98
p16	954	954	60	954	4.20	954	36.03
p17	1034	1034	60	1034	5.50	1034	48.18
p18	1043	1043	60	1043	9.06	1043	46.39
p19	1031	1031	60	1031	8.64	1031	38.56
p20	1005	1005	60	1005	27.34	1005	48.77
sic1	17288.99	17288.99	60	17288.99	50.50	17288.99	22.72
sjc2	33270.94	33293.40	600	33270.94	44.08	33270.94	112.81
sjc3a	45335.16	45338.02	2307	45335.16	8580.30	45335.16	940.75
sjc3b	40635.90	40635.90	2308	40635.90	2292.86	40635.90	1887.97
sjc4a	61925.51	61925.52	6109	61925.51	4221.47	61928.72	2885.11
sjc4b	52469.96	52531.46	6106	52469.96	3471.44	52531.27	7626.33

5 Conclusions

This paper has presented a solution for the Capacitated *p*-Median Problem (CPMP) using Clustering Search (CS). The CS is a new method that has been applied with success in some combinatorial optimization problems, such as pattern sequencing problem [9] and prize collecting traveling salesman problem [1]. The results show that the CS approach is competitive for the resolution of this problem in reasonable computational times. For the first set of instances considered in computational tests, the optimal values have been found, and for instances of the second set the best values known have been found in most of cases. Therefore, these results validate the CS application to the CPMP, and this approach has an additional advantage: does not use any commercial solver.

Further works can be done by analyzing other metaheuristics to generate solutions for the clustering process of CS, such as Ant Colony System, Tabu Search and Genetic Algorithm, and by implementing new local search heuristics for the CPMP. Besides that, bigger instances of this problem can be generated and solved.

References

1. Chaves, A.A. and Lorena, L.A.N.: Hybrid algorithms with detection of promising areas for the prize collecting traveling salesman problem. Fifth international conference on hybrid intelligent systems (2005) 49-54
2. Diaz, J.A. and Fernandez E.: Hybrid scatter search and path relinking for the capacitated p-median problem. European Journal of Operational Research, Vol. 169 (2006) 570 – 585
3. Fleszar, K. and Hindi, D.S.: An effective VNS for the capacitated p-median problem. European Journal of Operational Research (2007), doi:10.1016/j.ejor.2006.12.055
4. Garey, M.R., Johnson, D.S.: Computers and intractability: A guide to the theory of np-completeness, W.H. Freeman and Co. (1979)
5. Lorena L.A.N. and Senne E.L.F.: A column generation approach to capacitated p-median problems. Computers and Operational Research, Vol. 31 (1994) 863–876
6. Lorena L.A.N. and Senne E.L.F.: Local search heuristics for capacitated p-median problems, Networks and Spatial Economics, 3 (4) (2003) 407–419
7. Maniezzo V, Mingozzi A and Baldacci R.: A bionomic approach to the capacitated p-median problem. J Heuristics, 4 (1998) 263–280
8. Laguna, M. and Martí, R.: Scatter Search—Methodology and Implementations in C, Kluwer Academic Publishers, Boston (2003)
9. Oliveira, A.C.M. and Lorena, L.A.N.: Detecting promising areas by evolutionary clustering search.. Advances in Artificial Intelligence. Springer Lecture Notes in Artificial Intelligence Series (2004) 385–394
10. Osman, I.H., Ahmadi, S.: Guided construction search metaheuristics for the capacitated p-median problem with single source constraint, Journal of the Operational Research Society, (2006) 1–15
11. Osman, I.H., Christofides, N.: Capacitated clustering problems by hybrid simulated annealing and tabu search. Intern. Trans. in Operational Research, Vol.1 (3) (1994) 317–336
12. Scheuerer S. and Wendolsky R.: A scatter search heuristic for the capacitated clustering problem. European Journal of Operational Research, 169 (2006) 533–547.
13. S. Kirkpatrick and C. D. Gelatt and M. P. Vecchi, Optimization by Simulated Annealing, Science, Vol. 220, Number 4598 (1983) 671-680.