A Tabu Search Algorithm for the Capacitated Centred Clustering Problem

Albert Einstein Fernandes Muritiba[1], **Marcos Negreiros**[2], **Hedley Luna Gois Oriá**[2], **Michael Ferreira de Souza**[1]

[1] Universidade Federal do Ceará (UFC)
Departamento de Estatística e Matemática Aplicada (DEMA)

[2]Universidade Estadual do Ceará (UECE)
Mestrado Profissional em Computação Aplicada
MPCOMP/UECE-IFCE-UFRJ
Av. Paranjana, 1700 – Campus do Itaperi
CEP: 60740-000 – Fortaleza – CE – Brazil

einstein@ufc.br, negreiro@graphvs.com.br, hedleygois@gmail.com, michael@ufc.br

Resumo. *O problema de agrupamento capacitado em centro geométrico PACCG consiste em particionar um conjunto de n pontos no $\Re^2$ em p grupos disjuntos com capacidade limitada. A cada ponto está associado um valor de demanda e o objetivo é minimizar a soma das distâncias euclideanas entre os pontos e os seus respectivos centros. Neste trabalho, nós consideramos o PACCG e sua variante, o gCCCP, que não estabelece a priori o número de grupos p e estabelece um custo fixo F de abertura dos grupos. Nós propomos um conjunto efetivo de estratégias que se combinam à metaheurística clássica Busca Tabu, as quais em conjunto atingem melhores soluções que as até aqui publicadas pela literatura.*

Palavras Chaves*: Agrupamento, Agrupamento Min-Sum, PACCG, Metaheuristica.*
Área principal : MH

Abstract. *The capacitated centered clustering problem CCCP consists in partitioning a set of n points in $\Re^2$ in p disjoint clusters within a given capacity. Each point has an associated demand and the objective is to minimize the sum of the Euclidean distances between the points and the their respective clusters centroids. In this work, we address the CCCP and also its variant, the g-CCCP, which unleashes the number of clusters p and establishes the opening cost of clusters F. We propose effective strategies that combined with the classical Tabu Search metaheuristic outperform the recent methods published.*
Keywords*: Clustering, Min-Sum-Square Clustering, CCCP, Metaheuristics.*
Main area: MH

1 Introduction

Clustering is a very well known problem related to the process of assigning individuals to a number of disjoint partitions. The clustering problems may be classified in many ways, one of them is the min-sum of squares clustering. Its objective is to minimize a function of least-square distance between individuals to the geometric center of their partitions, in such a way that it is known previously the number of clusters to be built. In this type of clustering process, the solutions are composed of hyper-spherical clusters. The unconstrained min-sum-square clustering is largely studied and has a number of exact and approximative methods that can find very close to optimal solutions, [7], [3], [15], [16], [2].

Recently, a version of the problem was proposed by Negreiros & Palhano (2006), considering to the euclidean plane, a constrained process of doing min-sum clustering of a set of individuals, [8]. Once clustering can be done in many different ways, this new problem searches for solutions where there are limits on the capacity of the clusters or even their size (maximum number of individuals per cluster). This new problem is also NP-Hard, and introduces interesting research topics on exploring combinatorial optimization methods to solve it.

The related literature explored a number of applications of the CCCP in the dry food distribution logistics, designing zones for urban garbage collection, territorial design of salesmen regions, dengue desease control, [8], [10].

Two problems were introduced, the p-CCCP and the generalized CCCP or gCCCP. In the first version, the capacitated constrained min-sum clustering is bounded by a number of groups, where in the second generalized version there is no limit on the number of groups, but it is added to the objective function a fixed cost to open a new cluster. To evaluate the problems, the authors prepared a set of test instances extracted from diferent sources.

Effort is beeing done for some researchers in the direction of solving the CCCP. Negreiros & Palhano (2006) introduced the problem and proposed a binpacking greedy constructive heuristics and VNS based methods, to improve the solutions to a set of selected instances from many origins of real applications and special configuration tests. They evaluated instances for both versions of the problem, [12].

In the direction of the p-CCCP, new constructive heuristics were proposed by Palhano, Negreiros & Laporte (2008), the authors used spanning trees and Delaunay diagrams to perform the methods called *wave* and *fireworks*. The methods showed to be of better quality in the sense of doing good solutions then previous, the spatial clustering techniques reveal to be more effective for the set of instances evaluated. A new result was obtained for just one instance from garbage collection, [12]. The first column generation to the p-CCCP schema was proposed Pereira & Senne (2008), whitch results overpass mostly the previous works for a selected number of instances, [13]. Chaves & Lorena (2009) proposed a Cluster Search metaheuristic to the p-CCCP, combining simulated annealing and Cluster Search metaheuristic previously developed by Oliveira & Lorena (2007). Their results improved mostly 25 of the instances selected from the seminal work, [4], [11]. Chaves & Lorena (2011), most recently introduced a new combined procedure, by using Genetic Algorithms with local search VND heuristic to find promised clustering regions, and then apply Cluster Search metaheuristic. They found 16 new upper bounds for the 25 selected instances from the literature, [5].

Stefanello & Muller (2009) explored a new p-CCCP formulation, by introducing in the objective function a manhattan distance between the individuals and the center of the clusters. The formulation evaluated to the problem showed that for the selected instances (**sjc**), the results obtained were very far away (>100%) from the previous upper bounds published, [14].

The gCCCP was only investigated by Negreiros & Palhano (2006), and lately by Negreiros & Batista (2010), [8], [9]. The late work explored a B&B combinatorial procedure to the problem. They evaluated new instances from the max-cut problem literature, and proved optimality to instances up to 30 vertices. They also evaluated a set-partitioning approach using TS for the problem, by evaluating all the possible clusters of the instances. For all testes the authors achieved very close results (<0.2%) from the proved optimal instances.

In this work, section 2 review the mathematical formulation for both versions of the CCCP, in section 3 we show a constructive method. In section 4 we define a neighboorhood to p-CCCP and gCCCP. In section 5 we describe out the Tabu Search heuristic for the CCCP. In section 6 we evaluate the instances extracted from the literature to the p-CCCP and to the gCCCP, and compare the results obtained with the ones found in the literature.

2 Problem Description

Lets suppose that the CCCP can be represented by using the following set of parameters and variables:

l - is the dimension of the space ($l = 2$, in our case);
I - is the set of individuals;
J - is the set of clusters centers;
$|J|$ - is the cardinality of the set J, or a fixed number of clusters ($= p$);
p - is the number of clusters;
a_i - is a vector of l dimension with the coordenates of the individual i;
q_i - is the demand of an individual i;
Q - is the maximum capacity of a cluster;
n_j - is the number of individuals in cluster j;

$$\bar{x}_j = \begin{cases} 1, \text{is a vector of dimension } l \text{ representing the coordenates of the } \textit{cluster } j \\ 0, \text{otherwise} \end{cases}$$

$$y_{ij} = \begin{cases} 1, \text{if an individual } i \text{ is assigned to the cluster } j \\ 0, \text{otherwise} \end{cases}$$

$$(\mathbf{p-CCCP}) Minimize \sum_{i \in I} \sum_{j \in J} ||a_i - \bar{x}_j||\, y_{ij} \quad (1)$$

$$\text{such that}: \sum_{j \in J} y_{ij} = 1, \forall i \in I \quad (2)$$

$$\sum_{i \in I} y_{ij} \leq n_j, \forall j \in J \quad (3)$$

$$\sum_{i \in I} a_i\, y_{ij} \leq n_j \bar{x}_j, \forall j \in J \quad (4)$$

$$\sum_{i \in I} q_i\, y_{ij} = Q, \forall j \in J \quad (5)$$

$$\bar{x}_j \in \Re^l, n_j \in \aleph, y_{ij} \in \{0,1\},\ \forall i \in I, \forall j \in J \quad (6)$$

The objective function 1 wants to minimize the dissimilarity between *clusters*. The constraint 2 assign one individual to just one cluster. The constraint 3 consider a maximum number of individuals per *cluster*. The constraint 4 defines the center of the clusters. The constraint 5 limits the assigned individuals to the cluster maximum capacity. The constraint 4 refer to the decision variables of the problem.

The p-CCCP is non-linear. It is NP-Hard, once its unconstrained version is also NP-Hard, [7]. Its major difficulty is in fact in the knapsack constraint, althougth it is also non trivial if we just consider the constraints to form the center of the clusters.

Consider the previous model used to define p-CCCP, and suppose that the CCCP can be represented by also using the following set of parameters and variables:

F - fixed cost to open a cluster;

$$z_j = \begin{cases} 1, \text{If } \textit{cluster } j \text{ is opened} \\ 0, \text{otherwise} \end{cases}$$

The new formulation can be expressed as:

$$(\mathbf{gCCCP}) Minimize \sum_{j \in J} F\, z_j + \sum_{i \in I} \sum_{j \in J} ||a_i - \bar{x}_j||\, y_{ij} \tag{7}$$

$$\text{such that} : \sum_{j \in J} y_{ij} = 1, \forall i \in I \tag{8}$$

$$\sum_{j \in J} z_j \geq 1 \tag{9}$$

$$\sum_{i \in I} y_{ij} \leq n_j, \forall j \in J \tag{10}$$

$$\sum_{i \in I} a_i\, y_{ij} \leq n_j \bar{x}_j, \forall j \in J \tag{11}$$

$$\sum_{i \in I} q_i\, y_{ij} = Q\, z_j, \forall j \in J \tag{12}$$

$$\bar{x}_j \in \Re^l, n_j \in \aleph, z_j \in \{0,1\}, y_{ij} \in \{0,1\}, \ \forall i \in I, \forall j \in J \tag{13}$$

The difference between both problems are in: the objective function 7, that wants to minimize the dissimilarity between clusters while also minimizing the fixed cost of opening a new cluster, and a new constraint 10, which says that there is a necessity of opening at least a cluster to attend all the individuals.

3 Constructive heuristic

In order to provide a good starting solution to our Tabu Search procedure (TS), we propose a randomized best-fit constructive method for both problems. The p-RBFC and g-RBFC are versions of our constructive heuristic for the CCCP and g-CCCP respectively.

Both methods randomize the list of vertices, they use a random multistart greedy strategy to distribute the vertices through the clusters and them apply our local search procedure (LS, see section 4) to improve resulting solutions. This process is repeated few times, which by previous experiments we found to be 10 the best number to be used here

Algorithm 1 p - Randomized Best-Fit Constructive Heuristic.

Procedure RBFC
V - set of vertex
$C, best$ - set of clusters
$M()$ - the center of a cluster
$w()$ - the demand of vertex or the actual demand of a cluster.

1: **for** $k \leftarrow 1$ **to** $StartCount$ **do**
2: reset(C)
3: randomize(V)
4: **for** $p \leftarrow 1$ **to** P **do**
5: $C_p \leftarrow random(V)$
6: **end for**
7: **for** $i \leftarrow P + 1$ **to** $|V|$ **do**
8: $j \leftarrow argmin_{j \in C}\{||v_i - M(C_i)|| : w(v_i) + w(C_i) \leq Q\}$
9: *insert* v_i *into* C_j
10: **end for**
11: **if** $C < best$ **then**
12: $best \leftarrow C$
13: **end if**
14: **end for**
15: **return** best

in our implementation. Then, the best found solution is used to start the Tabu Search procedure.

The algorithm 1 shows the p-RBFC used to obtain an initial solution to the CCCP. In this algorithm we have p as a given number of clusters. These clusters are initially filled with the p first vertices of a randomized vector V, steps 2-6. Then, a best-fit fashion strategy places the remainders vertices in the nearest feasible cluster.

The algorithm 2, in other hand, treat the gCCCP. The g-RBFC version works placing the vertex v_i from the randomized vector V into the nearest feasible cluster. A new cluster is opened whenever any feasible cluster could not be found. For this case, we consider an infeasible cluster, also, the cluster C_j whose the distance between its current centroid $M(C_j)$ and the vertex v_i is greater or equal to the opening cost F.

The computational performance of the constructive heuristics above reported are discussed in section 6.

4 Local Search Movements

The core of our method is the local search (LS). It scans efficiently the neighborhood of a feasible solution. The LS procedure searches for improving movements committing them as soon as they are found. Our affords aimed a simple but efficient local search.

Three types of movements compose our LS. They are called *transfer, swap* and *wave* movements.

Transfer: An individual is transfered from a cluster to another. In this movement, an individual i is removed from a cluster A and placed in a different cluster B with enough free capacity ($Q - w(B) \geq w(i)$) whenever the global cost is improved.

In opposition to CCCP, the gCCCP transfer version permits to create or destruct a cluster by, respectively, transferring a vertex to an empty cluster or removing a vertex from a cluster composed of a single individual. Algorithm 3 shows this procedure.

Algorithm 2 g - Randomized Best-Fit Constructive Heuristic.

Procedure g-RBFC
V - set of vertex
$C, best$ - set of clusters
$M()$ - the center of a cluster.
$w()$ - the demand of vertex or the actual demand of a cluster.

for $k \leftarrow 1$ **to** $StartCount$ **do**
  reset(C)
  randomize(V)
  $P \leftarrow 0$
  for $i \leftarrow 0$ **to** $|V|$ **do**
    $j \leftarrow argmin_{j \in C}\{||v_i - M(C_j)|| : w(v_i) + w(C_j) \leq Q\}$
    if $j \neq$ **then**
      insert v_i *into* C_j
    else
      $P \leftarrow P + 1$
      insert v_i *into* C_P
    end if
  end for
  if $C < best$ **then**
    $best \leftarrow C$
  end if
end for
return best

Algorithm 3 Transfer movement.

Procedure Transfer(A,B)
A, B - clusters in a current solution C
$Z(X)$ - cluster X cost
$w()$ - the demand of vertex or the actual demand of a cluster.

for all $a \in A$ **do**
  if $w(B) + w(a) \leq Q$ **and** $Z(A - \{a\}) + Z(B \cup \{a\}) < Z(A) + Z(B)$ **then**
    $A \leftarrow A - \{a\}$
    $B \leftarrow B \cup \{a\}$
    $A_{flag} \leftarrow$ **true**
    $B_{flag} \leftarrow$ **true**
  end if
end for

Swap: The individual i from the cluster A is exchanged by the individual j from the cluster B. This movement takes two individuals, i from a cluster A and j from a different cluster B, and places point i in B and j in A. The necessary condition to not exceed the clusters capacity are: $Q - w(A) + w(i) \geq w(j)$ and $Q - w(B) + w(j) \geq w(i)$. Note that the transfer does not dominate the swap movement since the condition needed to swap may not imply in the necessary conditions ($Q - w(B) \geq w(i)$ and $Q - w(A) \geq w(j)$). Algorithm 4 shows the procedure.

The Swap movement can become computationally expansive due to the great amount of cluster costs recalculations. To speed up the cost verification we established a function called *guess*, that can provide an approximative value for the resulting clusters objective function. The *guess* function calculate a delta (δ) cost by computing the equation 16:

Algorithm 4 Swap movement.

Procedure Swap(A,B)
A, B - clusters in a current solution C
$Z(X)$ - cluster X cost
$w()$ - the demand of vertex or the actual demand of a cluster.

for all $a \in A$ **do**
  for all $b \in B$ **do**
    $Ok := w(A) + w(b) - w(a) \leq Q$
    $Ok := Ok$ **and** $w(B) + w(a) - w(b) \leq Q$
    $Ok := Ok$ **and** $Z((A \cup \{b\}) - \{a\}) + Z((B \cup \{a\}) - \{b\}) < Z(A) + Z(B)$
    if Ok **then**
      $A \leftarrow (A \cup \{b\}) - \{a\}$
      $B \leftarrow (B \cup \{a\}) - \{b\}$
      $A_{flag} \leftarrow$ **true**
      $B_{flag} \leftarrow$ **true**
    end if
  end for
end for

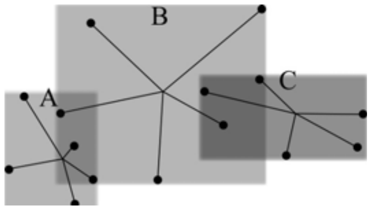

Figure 1. Overlapping clusters, as overlapping boxes

$$M(A) = \frac{\sum_{a_i \in A} a_i}{|A|} \tag{14}$$

$$\tilde{M}(A,i,j) = \frac{(M(A) * |A| - a_i + a_j)}{|A|} \tag{15}$$

$$\delta = \|\tilde{M}(A,i,j) - a_j\| - \|M(A) - a_i\| + \|\tilde{M}(B,j,i) - a_i\| - \|M(B) - a_j\| \tag{16}$$

If δ is less then a small ϵ, we may compute the exact value of the cost of the movement. Note that the equation 14, which computes the clusters centroids, can be stored on the cluster data structure, thus the whole calculation can be performed in $O(1)$.

In addition, we only consider a *swap* movement between a pair of clusters if the clusters boxes intercept each other. As shown in figure 1, we define as the box of a cluster the rectangle formed by the minimums and maximums (x, y) coordinates of its vertex. For this case, the procedure will not attempt to swap vertices between clusters A and C. More over, only vertex belonging to the common area can be swapped. This filter represents an important speed up for instances where the number of vertex in a single cluster is significant.

Wave: In this movement, an individual i^k is removed from the cluster A^k and be inserted in another cluster A^{k+1}, whenever there would be an improvement in the global cost, even if the cluster A^{k+1} overflows its capacity. In this case, the point i^{k+1}, that maximizes the distance to i_k, is removed from A^{k+1} and is inserted in another cluster $A^{k+2} \neq A^{k+1}$. The process is recursively repeated for a given maximum value to k (30 in

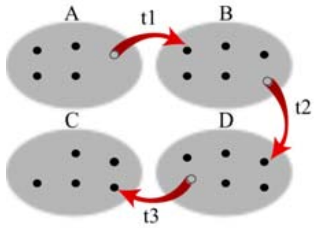

Figure 2. An example of the Wave movement between vertices of clusters

our experiments). We also set a maximum number of calls as the number of vertex in the problem, forcing the procedure to be $O(n)$. The algorithm 5 shows the procedure.

The figure 2 represents a wave movement. For sake of simplicity, we assume in the example that the vertices have demand equal to one and the capacity of each cluster is 5. The cluster A, B and D have initially 5 vertices each one, while cluster C has 4. There is no improving transfer moves between clusters A and C, B and C or D and C. However, the unfeasible transfer $t1$ of a vertex from A to B results in a cost improvement. The execution of $t1$ implies in the need of $t2$, a transfer move that will make cluster B be feasible. Note that $t2$ is not necessarily an improving transfer, it can be as worst as the accumulate improvement permits. The $t2$ transfer makes cluster D infeasible and the $t3$ transfer becomes necessary. The $t3$ transfers a vertex to the cluster C, which had 4 vertices. With all clusters feasible and some improvement, the $wave$ movement returns successfully.

The LS procedure combines the movements above to provide a fast intensification. As shown in the algorithm 6, the LS takes use of a cluster flag that is set to *false* when the search process starts and set *true* if the cluster is altered by any movement. Then we just check movements involving at least one flagged cluster, avoiding redundant computations.

5 Tabu Search

Our tabu search algorithm (TS) is a classical TS procedure, as proposed originally by Glover (1989), it consists in applying a local search method up to reach a local optimum, then a spoil movement is forced and a rule is stored in the tabu list in order to avoid the immediate return to the previous local optimum, [6].

The tabu method is shown in the algorithm 7. The command in line 9 consists in searching for the movement that will less spoil the current solution, commit it, and store the role in the tabu list to avoid doing this movement.

6 Computational Results

6.1 Benchmark instances

The benchmark instances are those available in the literature related to the CCCP. There are seven in the group of *TA*, six in the group *sjc*, five in the group *p3038* and, at last, seven in the group *doni*. All of these instances can be obtained from OR-library, [1].

Table 1 shows some statistics of the instances. The columns indicates the name of the instance - *Name*, number of individuals - *n*, number of clusters - *clusters p*, the capacity

Name	n	p	Q	w_Avg	w_Dev
TA25	25	5	6	1	0
TA50	50	5	11	1	0
TA60	60	5	13	1	0
TA70	70	5	17	1	0
TA80	80	7	12	1	0
TA90	90	4	23	1	0
TA100	100	6	17	1	0
SJC1	100	10	720	58.07	51.86
SJC2	200	15	840	46.34	37.15
SJC3a	300	25	740	37.51	29.58
SJC3b	300	30	740	37.51	29.58
SJC4a	402	30	840	39.76	32.87
SJC4b	402	40	840	39.76	32.87
p3038_600	3038	600	321	50.85	24.75
p3038_700	3038	700	273	50.33	25
p3038_800	3038	800	238	50.26	25
p3038_900	3038	900	216	51.29	25.34
p3038_1000	3038	1000	191	50.44	24.87
doni1	1000	6	200	1	0
doni2	2000	6	400	1	0
doni3	3000	8	400	1	0
doni4	4000	10	400	1	0
doni5	5000	12	450	1	0
doni6	10000	23	450	1	0
doni7	13000	30	450	1	0

Table 1. Characteristics of the CCCP benchmark instances

of each cluster - Q, the average demand of each vertex - (**q_Avg**) and the standard deviation of the demand of each vertex, (**q_Dev**).

The codes were done in C++(4.5.2) under Ubuntu Linux 11.04. The hardware used was an Intel Core 2 Quad Q9550 CPU 2.83Ghz (per core, 4 cores), 4 GBytes of RAM. The CPU times were measured in seconds, the codes were done to use just one core, in the sequential form.

To proceed with the tests, we created a priori 100 initial solutions to the tabu search, also limited the process in 48h, where 24h were given to the constructive phase and 24h to the refinement phase. The tabu search was imposed to be limited to 4000 iterations.

6.2 Results for CCCP

In the table 2, the columns show respectively the name of the instance (**Name**), the initial solution cost (**Start-sol**), the needed CPU time to obtain the initial solution (**Start-time**),the tenure used (**tenure**), the solution cost found (**Sol**), the CPU time needed to find the best solution (**Best-time**), the maximum number of iteration without current solution cost improvement (**#Ite**), the global time (**time**) to process the method, and the percentual improvement gap obtained between the starting and final solution achieved (**GAP**). The times on the table 2 are reported in seconds, rounded to the nearest integer value.

The columns **p_min** and **pMinCost** are respectively the minimum number of cluster that our constructive heuristic was still able to produce an initial solution, and the cost of this solution. The columns shows that our method found no difficulties to produce initial solutions to the bechmark instances, and averagely the starting solution is 0.59% far from the best solution obtained by the classical TS.

Name	#Starts	Start-sol	Start-time	Tenure	#Ite	Sol	Best-time	time	GAP%
TA25	10	1256.62	0	7	1000	1251.45	0	0	0.41
TA50	10	4474.52	0	7	1000	4474.52	0	0	0.00
TA60	10	5370.05	0	7	1000	5356.58	0	0	0.25
TA70	10	6241.56	0	7	1000	6240.67	0	0	0.01
TA80	10	5730.28	0	7	1000	5730.28	0	0	0.00
TA90	10	9069.85	0	7	1000	9069.85	0	1	0.00
TA100	10	8116.71	0	7	1000	8102.04	0	0	0.18
SJC1	10	17588.62	0	59	1000	17359.75	0	1	1.32
SJC2	10	33637.60	0	59	1000	33181.65	0	3	1.37
SJC3a	10	45923.61	0	149	5000	45356.35	2	24	1.25
SJC3b	100	41008.12	1	59	5000	40661.94	4	21	0.85
SJC4a	100	62737.21	3	149	10000	61993.66	4	133	1.20
SJC4b	100	53006.33	3	59	10000	52202.48	94	153	1.54
p3038_600	10	127947.34	34	101	1000	126567.31	435	810	1.09
p3038_700	10	115893.45	38	101	1000	115168.49	600	1022	0.63
p3038_800	10	105860.22	68	101	1000	105352.33	1405	2411	0.48
p3038_900	10	98191.75	73	101	1000	97319.54	898	1650	0.90
p3038_1000	10	90328.40	88	101	1000	89896.55	499	1017	0.48
doni1	20	3052.33	3	101	1000	3025.12	13	40	0.90
doni2	20	6393.10	36	101	1000	6384.84	45	142	0.13
doni3	20	8345.57	105	101	1000	8343.49	627	1032	0.02
doni4	10	10814.29	79	101	500	10777.64	969	1450	0.34
doni5	10	11115.25	114	101	500	11114.67	175	437	0.01
doni6	3	15736.19	245	101	500	15610.46	2972	5476	0.81
doni7	3	18595.48	535	101	300	18484.13	32074	36878	0.60

Table 2. Computational results for p-CCCP benchmark instances

6.3 Results for g-CCCP

For the run with g-CCCP approach, we performed three runs for each instance, changing their opening cost. We here estabilish open cost parameters to the benchmark instances for future algorithm evaluations.

Table 3 shows the obtained values for each run. Each row display the instance's name (**Name**), the applied opening-cost (Opening), the g-RBFC procedure iteration (**#Starts**), the initial solution cost (**Start-cost**), the number of point on the starting solution (**Start-p**), the needed CPU time to run g-RBFC procedure (**Start-t**), the tenure used (**tenure**), the maximum number of iteration without current solution cost improvement (**#Ite**), the found solution cost (**Sol-cost**), the number of cluster in the found solution (**Sol-p**), the CPU time needed to find the best solution (**Best-t**) and the global time (**Time**).

Note that the columns **Start-cost** and **Sol-cost** show only the costs referent to the sum of distances between the cluster's points and its centroid. The opening cost are taken only in the column **Sol**.

6.4 Comparisons of Performance

The table 4 compares the computational results for the CCCP with the results obtained from the literature.

In table 4 we compare the results obtained using the tabu search method proposed with the existing reported results to the CCCP, from [8], [13],[12],[4], [5].

Note that in 20/25 runs, we obtained the best known solutions where 12 are unheard solutions for [13]. Our average CPU time is smaller than the ones reported in [4] and [5] whose have comparable machines and compiler. Our executions were always performed faster than the ones reported by [5] (even our machine clock being a bit slower). The exceptions are the **Doni3** and **Doni7** runs.

In terms of percentage gap from best known solution, our method have average of 0.44% against 1.06 (2.4 times) of [5], 1.87 (4.25 times) of [4] whose have run all the instances.

Specifically, for the p3038 instances' class, our results was far better than the previous literature ones. Our solutions cost are averagely 2.13% better than [5] ones while out CPU times were roughly 10 times faster. It can be also noted that even our starting solutions were superior in cost. This could happen because our methodology scan better the neighbourhood space between clusters. These particular instances have a big number of clusters with few number of points. The neighbour between clusters is very wide. Probably there is still room for future improvements.

7 Conclusions

We proposed a simple Tabu search scheme, as well as its movements and local search procedure, that shows itself competitive for all benchmark instances with the state of art methods to the CCCP. The robustness of our approach can be reinforced by the fact that we obtained the best known solutions on 80% of runs on a such heterogeneous set of benchmark instances classes.

We extended our method to embrace the gCCCP case, and we reported some computational results with the new correspondents opening costs parameters. The resulting method can be improved by the addiction of mechanisms that permit easier change on the number of opened clusters, that would be the focus of future studies.

In our experiments we found a noticiable improvement in p3038 instances, which have a wide neighbour between clusters with few individuals. These instances still open room for future improvements.

Algorithm 5 Wave movement.

Procedure Wave(A,l,ṽ)
A - cluster in a current solution Cur
l - level of recursion
ṽ - last propagated vertex
$Z(X)$ - cluster X's cost
δ - global change in the current solution cost
$M(X)$ - cluster X's geometric center
$w()$ - the demand of vertex or the actual demand of a cluster.

if $l >$ MAXLEVEL **then**
  return
end if
if $l = 0$ **then**
  $v \leftarrow \arg\max_{i \in A}\{||a_i - M(A)||)\}$
else
  $v \leftarrow \arg\max_{i \in A}\{||a_i - M(A)||)\} : w(A) - w(i) \leq Q$ and $i \neq \tilde{v}$
end if
$\delta \leftarrow \delta + Z(A - \{v\}) - Z(A)$
$A \leftarrow A - \{v\}$
$auxFA \leftarrow A_{flag}$
$A_{flag} \leftarrow$ **true**
for all $B \in S : B \neq A$ **do**
  if $\delta + Z(B \cup \{v\}) - Z(B) < 0$ **then**
    $\delta \leftarrow \delta + Z(B \cup \{v\}) - Z(B)$
    $B \leftarrow B \cup \{v\}$
    $auxFB \leftarrow B_{flag}$
    $B_{flag} \leftarrow$ **true**
    if $w(v) + w(B) \leq Q$ **then**
      return
    else
      call **Wave**$(B, l + 1, v)$
      if $\delta < 0$ **then**
        return
      end if
    end if
    $\delta \leftarrow \delta + Z(B - \{v\}) - Z(B)$
    $B \leftarrow B - \{v\}$
    $B_{flag} \leftarrow auxFB$
  end if
end for
$\delta \leftarrow \delta + Z(A \cup \{v\}) - Z(A)$
$A \leftarrow A \cup \{v\}$
$A_{flag} \leftarrow auxFA$

Algorithm 6 Local Search.

Procedure LocalSearch
repeat
  for all $A \in C$ **do**
    for all $B \in C : B \neq A$ **and** $(A_{flag}$ **or** $B_{flag})$ **do**
      call **Swap**(A,B)
      call **Transfer**(A,B)
    end for
    call **Wave**(A,0,ϕ)
  end for
until No movement has been committed

Algorithm 7 Tabu Search.

Procedure Tabu Search

call Randomized Best-Fit Constructive Heuristic.
$count \leftarrow 0$
repeat
    call LocalSearch(C).
    if $C < best$ **then**
        $best \leftarrow C$
        $count \leftarrow 0$
    end if
    Apply Tabu movement.
    $count \leftarrow count + 1$
until $count = MAXITE$

Name	Opening	#Starts	Start-cost	Start-p	Start-t	Tenure	#Ite	Sol-cost	Sol-p	Sol	Best-t	Time
	100	10	1035.53	6	0	7	1000	1035.53	6	1635.53	0	0
TA25	300	10	1251.44	5	0	7	1000	1251.44	5	2751.44	0	0
	600	10	1251.44	5	0	7	1000	1251.44	5	4251.44	0	0
	100	10	1486.3	17	0	7	1000	1486.3	17	3186.30	0	0
TA50	300	10	3550.88	7	1	7	1000	3546.34	7	5646.34	1	1
	600	10	4474.51	5	0	7	1000	4474.51	5	7474.51	0	0
	100	10	2005.05	19	0	7	1000	1869.14	20	3869.14	0	0
TA60	300	10	4191.95	7	0	7	1000	4191.95	7	6291.95	0	0
	600	10	5356.58	5	0	7	1000	5356.58	5	8356.58	0	0
	100	10	2350.61	18	0	7	1000	2350.61	18	4150.61	0	0
TA70	300	10	2350.61	18	0	7	1000	2350.61	18	7750.61	1	1
	600	10	2350.61	18	0	7	1000	2350.61	18	13150.61	0	0
	100	10	2668.46	20	0	7	1000	2550.33	21	4650.33	0	0
TA80	300	10	4668.28	9	0	7	1000	4668.28	9	7368.28	0	0
	600	10	5740.58	7	0	7	1000	5730.28	7	9930.28	1	1
	100	10	2945.16	22	0	7	1000	2787.2	23	5087.20	0	0
TA90	300	10	6315.97	7	0	7	1000	6315.97	7	8415.97	0	0
	600	10	7791.95	5	0	7	1000	7791.95	5	10791.95	0	0
	100	10	3509.34	20	0	7	1000	3443.36	20	5443.36	0	1
TA100	300	10	6407.02	8	0	7	1000	6407.02	8	8807.02	0	0
	600	10	8126.82	6	0	7	1000	8115.7	6	11715.70	0	1
	1500	10	17486.93	10	0	59	1000	17359.75	10	32359.75	0	1
SJC1	2000	10	19437.46	9	0	59	1000	18543.66	10	38543.66	0	0
	2500	10	19437.46	9	0	59	1000	18642.08	10	43642.08	0	1
	1500	10	36587.82	13	0	59	1000	36232.14	13	55732.14	0	1
SJC2	2000	10	38477.38	12	0	59	1000	38477.38	12	62477.38	0	1
	2500	10	38298.01	12	0	59	1000	38298.01	12	68298.01	0	1
	1500	10	57676.79	18	0	149	5000	56718.29	18	83718.29	17	33
SJC3a	2000	10	59731.05	17	0	149	5000	59354.39	17	93354.39	4	19
	2500	10	60069.24	17	0	149	5000	59414.76	17	101914.76	8	25
	1500	10	57676.79	18	0	59	5000	56679.32	18	83679.32	2	13
SJC3b	2000	10	59731.05	17	0	59	5000	59139.17	17	93139.17	5	17
	2500	10	60069.24	17	0	59	5000	59076.84	17	101576.84	4	15
	1500	100	76945.61	22	4	149	10000	76942.85	22	109942.85	4	53
SJC4a	2000	100	77474.25	22	4	149	10000	74549.24	23	120549.24	112	160
	2500	100	80112.55	21	4	149	10000	79948.54	21	132448.54	20	70
	1500	100	76945.62	22	3	59	10000	76845.05	22	109845.05	6	44
SJC4b	2000	100	77474.25	22	3	59	10000	76763.06	22	120763.06	29	67
	2500	100	80112.55	21	4	59	10000	79723.06	21	132223.06	42	82
	500	10	144451.32	524	52	101	1000	143923.65	524	405923.65	571	959
p3038_600	750	10	148713.46	513	93	101	1000	147761.9	513	532511.90	724	1236
	1000	10	152771.65	506	169	101	1000	151369.25	506	657369.25	1338	2247
	500	10	133692.45	607	81	101	1000	132359.21	608	436359.21	2726	4101
p3038_700	750	10	138113.43	596	194	101	1000	136379.91	596	583379.91	2231	3444
	1000	10	42891.08	588	281	101	1000	141745.64	590	731745.64	408	3903
	500	10	127302.21	692	196	101	1000	125832.87	694	472832.87	1717	3679
p3038_800	750	10	131598.25	681	420	101	1000	129379.49	682	640879.49	4509	7728
	1000	10	132556.97	678	608	101	1000	131198.77	678	809198.77	10498	15090
	500	10	117166.41	785	282	101	1000	115352	786	508352.00	5920	9266
p3038_900	750	10	124198.15	769	671	101	1000	122462.02	769	699212.02	9210	15888
	1000	10	127827.03	762	1165	101	1000	125612.71	762	887612.71	14310	24279
	500	10	110962.98	871	544	101	1000	109228.96	871	544728.96	15496	20689
p3038_1000	750	10	116480.70	857	875	101	1000	113881.12	857	756631.12	8614	14773
	1000	10	121952.01	849	1282	101	1000	118697.67	849	967697.67	13511	26520
	10	20	2083.46	13	3	101	1000	2067.37	13	2197.37	28	53
doni1	20	20	2587.31	9	5	101	1000	2586.57	9	2766.57	29	57
	40	20	3042.15	6	3	101	1000	3024.99	6	3264.99	30	56
	10	20	4349.28	12	13	101	1000	4347.62	12	4467.62	14	107
doni2	20	20	4999.67	10	18	101	1000	4992.42	10	5192.42	65	178
	40	20	6428.1	6	19	101	1000	6400.4	6	6640.40	97	198
	10	20	5241.03	16	35	101	1000	5159.83	16	5319.83	43	209
doni3	20	20	6431.9	13	72	101	1000	6412.79	13	6672.79	105	301
	40	20	8344.92	8	173	101	1000	8343.65	8	8663.65	340	739
	10	10	6449.84	19	29	101	500	6432.03	19	6622.03	41	174
doni4	20	10	7383.55	14	55	101	500	7381.69	14	7661.69	61	207
	40	10	10111.6	11	66	101	500	10103.01	11	10543.01	296	517
	10	10	7705.22	19	47	101	500	7705.12	19	7895.12	88	300
doni5	20	10	9147.51	16	81	101	500	9135.49	16	9455.49	529	763
	40	10	11117.64	12	152	101	500	11112.05	12	11592.05	339	614
	10	3	12112.15	28	107	101	500	12001.98	28	12281.98	797	1679
doni6	20	3	13605.61	25	321	101	500	13582.54	25	14082.54	1672	2591
	40	3	15574.27	23	325	101	500	15519.26	23	16439.26	5017	6753
	10	3	13527.13	38	278	101	300	13393.5	38	13773.50	672	1714
doni7	20	3	15919.46	32	488	101	300	15739.82	33	16399.82	6177	10175
	40	3	18698.44	30	597	101	300	18435.28	30	19635.28	65923	70728

Table 3. Computational results with new open cost parameters for g-CCCP benchmark instances

Instance	Best-know	TS				Chaves & Lorena (2011)				Chaves & Lorena (2010)		Pereira & Senne (2008)		Palhano et al (2008)		Negreiros & Palhano (2006)	
		Sol	Gap	Best-time	time	Best-Sol	Gap	Best-time	time	Best-Sol	Gap	Best-Sol	Gap	Best-Sol	Gap	Best-Sol	Gap
TA25	1251,44	1251,44	0,00	0,00	0	1251,44	0,00	0,68	2	1251,44	0,00	1280,49	2,32	–	–	1251,44	0,00
TA50	4474,52	4474,52	0,00	0	0	4474,52	0,00	0,99	6	4474,52	0,00	4474,52	0,00	–	–	4476,12	0,04
TA60	5356,58	5356,58	0,00	0	0	5356,58	0,00	1,05	9	5356,58	0,00	5357,34	0,01	–	–	5356,58	0,00
TA70	6240,67	6240,67	0,00	0,00	0	6240,67	0,00	0,77	9	6240,67	0,00	6240,67	0,00	–	–	6241,55	0,01
TA80	5515,46	5730,28	3,89	0	0	5730,28	3,89	2,59	23	5730,28	3,89	5515,46	0,00	–	–	5730,28	3,89
TA90	8899,05	9069,85	1,92	0	1	9069,85	1,92	1,26	22	9069,85	1,92	8899,05	0,00	–	–	9103,21	2,29
TA100	8102,04	8102,04	0,00	0	0	8102,04	0,00	11,24	49	8102,04	0,00	8168,36	0,82	–	–	8122,67	0,25
[1ex] SJC1	17359,75	17359,75	0,00	0	1	17359,75	0,00	8,17	39	17359,75	0,00	17375,36	0,09	20341,34	17,18	17696,53	1,94
SJC2	33181,65	33181,65	0,00	0	3	33181,65	0,00	40,37	179	33181,65	0,00	33357,75	0,53	35211,99	6,12	33423,84	0,73
SJC3a	45356,35	45356,35	0,00	2	24	45358,23	0,00	509,66	1207	45366,35	0,02	45379,69	0,05	50590,49	11,54	47985,29	5,80
SJC3b	40661,94	40661,94	0,00	4	21	40661,94	0,00	771,83	1434	40695,46	0,08	41185,18	1,29	–	–	–	–
SJC4a	61931,60	61993,66	0,10	4	133	61931,60	0,00	1092,97	3026	61944,85	0,02	61969,06	0,06	69283,05	11,87	66689,96	7,68
SJC4b	52202,48	52202,48	0,00	94	153	52227,60	0,05	1965,82	3995	52214,55	0,02	52989,44	1,51	–	–	–	–
[1ex] p3038_600	126567,31	126567,31	0,00	435	810	128419,95	1,46	6137,67	9737	129194,11	2,08	–	–	135481,99	7,04	192024,83	51,72
p3038_700	115168,49	115168,49	0,00	600	1022	116325,05	1,00	6848,52	11658	117295,47	1,85	–	–	123698,76	7,41	176731,07	53,45
p3038_800	105352,33	105352,33	0,00	1405	2411	107764,69	2,29	8335,36	13195	109532,61	3,97	–	–	117705,48	11,73	184502,38	75,13
p3038_900	97319,54	97319,54	0,00	898	1650	99968,15	2,72	11726,17	15342	102458,93	5,28	–	–	111033,27	14,09	176781,51	81,65
p3038_1000	89896,55	89896,55	0,00	499	1017	92706,38	3,13	10747,13	17128	97771,67	8,76	–	–	110049,78	22,42	159139,89	77,03
[1ex] doni1	3021,41	3025,12	0,12	13	40	3027,63	0,21	86,38	127	3022,26	0,03	–	–	3234,58	7,06	3021,41	0,00
doni2	6080,70	6384,84	5,00	45	142	6373,26	4,81	309,77	687	6372,81	4,80	–	–	6692,71	10,06	6080,70	0,00
doni3	8343,49	8343,49	0,00	627	1032	8438,96	1,14	624,12	928	8446,08	1,23	–	–	9797,12	17,42	8769,05	5,10
doni4	10777,64	10777,64	0,00	969	1450	10952,27	1,62	1069,07	2390	10854,48	0,71	–	–	11594,07	7,58	11516,14	6,85
doni5	11114,67	11114,67	0,00	175	437	11209,99	0,86	2175,04	3624	11134,94	0,18	–	–	11827,69	6,42	11635,18	4,68
doni6	15610,46	15610,46	0,00	2972	5476	15722,67	0,72	6174,83	10317	15928,38	2,04	–	–	–	–	18443,50	18,15
doni7	18484,13	18484,13	0,00	32074	36878	18596,74	0,61	15860,55	26914	20291,52	9,78	–	–	–	–	23478,79	27,02
AVG	35930,81	35961,03	0,44	2908,00	4363,75	36418,08	1,06	2980,08	4882	36931,65	1,87	22476,34	0,51	58324,45	11,28	51226,17	18,41

Table 4. Comparison of results obtained between published works for p-CCCP benchmark instances

References

[1] *Orlibrary*, (2012), http://people.brunel.ac.uk/ mastjjb/jeb/info.html.

[2] D. ALOISE, P. HANSEN, and L. LIBERTI, *An improved column generation algorithm for minimum sum-of-squares clustering*, Mathematical Programming Section A **131** (2012), 195–220.

[3] A.M. BAGIROV, *Modified global k-means algorithm for minimum sum-of-squares clustering problems*, Pattern Recognition **41** (2009), 3192–3199.

[4] A. A. CHAVES and A. LORENA, *Clustering search algorithm for the capacitated centred clustering problem*, Computers and Operations Research **37** (2010), 552–558.

[5] ______, *Hybrid evolutionary algorithm for the capacitated centered clustering problem*, Expert Systems with Applications **38** (2011), 5013–5018.

[6] F. GLOVER, *Tabu search - part 1*, ORSA Journal on Computing **1(3)** (1989), 190–206.

[7] P. HANSEN and JAUMARD, *Cluster analysis and mathematical programming*, Mathematical Programming (1997), 191–215.

[8] M.J. NEGREIROS and A.W.C.PALHANO, *The capacitated centred clustering problem*, Computers and Operations Research **33** (2006), 1639–1663.

[9] M.J. NEGREIROS, P.L.F. BATISTA, and A.W.C. PALHANO, *The capacitated centred clustering problem*, Annals of the ALIO/INFORMS (2010), A New Anytime B&B Approach for the Capacitated Centred Clustering Problem.

[10] M.J. NEGREIROS, A.E. XAVIER, A.F.S. XAVIER, N. MACULAN, P. MICHELON, J.W.O. LIMA, and L.O.M. ANDRADE, *Optimization models, statistical and dss tools for prevention and combat of dengue disease*, vol. Chapter 7, INTECH, 2011.

[11] A. C. OLIVEIRA and A. LORENA, *Hybrid evolucionary algorithms and clustering search*, In C. Grosan, A. Abraham, H. Ishibush (Eds.), Hybrid Evolucionary Systems - Studies in computational intelligence (2007), 81–102.

[12] A.W.C. PALHANO, M.J. NEGREIROS, and G. LAPORTE, *A constrained k-median procedure for the capacitated centred clustering problem*, Anales del XIV CLAIO, *cd rom* (2008).

[13] M. PEREIRA and E. SENNE, *A column generation method for the capacitated centred clustering problem*, Annals of VI ALIO/EURO (2008), 1–6.

[14] F. STEFANELLO and F.M. MULLER, *Um estudo sobre problemas de agrupamento capacitado*, Anais do XLI SBPO in CD-ROM (2009), 2819–2828.

[15] A.E. XAVIER, *The hyperbolic smoothing clustering method*, Pattern Recognition **43** (2010), 731–737.

[16] ______, *Solving the min-sum-of-squares clustering problem by hyperbolic smoothing and partition into boundary and gravitational regions*, Pattern Recognition **44** (2011), 70–77.