Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computers & Operations Research 33 (2006) 1639–1663
computers & operations research
www.elsevier.com/locate/cor

The capacitated centred clustering problem

Marcos Negreiros[a,*], Augusto Palhano[b]

[a] *Universidade Estadual do Ceará, Mestrado Profissional em Computação-UECE/CEFET, Av. Paranjana, 1700-Campus do Itaperi, Fortaleza-Ceará-CEP: 60740-000, Brazil*
[b] *GRAPHVS CONS. COM. & REP. LTDA, Av. Santos Dumont, 2789 lj 12 (Aldeota), Fortaleza-Ceará-CEP: 60150-161, Brazil*

Available online 7 January 2005

Abstract

The capacitated centred clustering problem (CCCP) consists of defining a set of clusters with limited capacity and maximum proper similarity per cluster. Each cluster is composed of individuals from whom we can compute a centre value and hence, determine a similarity measure. The clusters must cover the demands of their individuals. This problem can be applied to the design of garbage collection zones, defining salesmen areas, etc. In this work, we present two variations (*p*-CCCP and Generic CCCP) of this problem and their mathematical programming formulations. We first focus our attention on the Generic CCCP, and then we create a new procedure for *p*-CCCP. These problems being NP-HARD, we propose a two-phase polynomial heuristic algorithm. The first phase is a constructive phase for which we will propose two variants: the first technique uses known cluster procedures oriented by a log-polynomial geometric tree search, the other one uses unconstrained to constrained clustering. The second phase is a refinement of the variable neighbourhood search (VNS). We also show the results we have obtained for tests from the CCP literature, the design of garbage collection zones, and salesmen areas distribution using the approach implemented for the SISROT® system.
© 2004 Elsevier Ltd. All rights reserved.

Keywords: Clustering; Capacitated clustering problem; VNS metaheuristic

1. Introduction

Clustering analysis involves the grouping of data entities (points) in a way that maximises the homogeneity of points within a group and, at the same time, the heterogeneity of points between groups [1–4].

* Corresponding author.
E-mail addresses: negreiro@uece.br (M. Negreiros), augustopalhano@graphvs.com.br (A. Palhano).

0305-0548/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2004.11.011

Capacitated clustering problems (CCP) have arisen when Mulvey and Beck proposed the first model for this problem. The main idea is based on a problem where we want to find capacitated clusters (each cluster with a given capacity) centred by a median of its individuals (objects or customers) that minimises an objective function that is described by the sum of the total dissimilarity between each individual and its median [5]. The original formulation of the problem can be stated as follows (CCP):

$$\text{(CCP)} \quad \text{Min} \sum_{i\in I}\sum_{j\in J} d_{ij}x_{ij} \tag{1.1}$$

such that,

$$\sum_{j\in J} x_{ij} = 1, \quad \forall i \in I, \quad \forall j \in J, \tag{1.2}$$

$$\sum_{j\in J} y_j \leqslant p, \quad \forall i \in I, \tag{1.3}$$

$$x_{ij} \leqslant y_j, \quad \forall i \in I, \quad \forall j \in J, \tag{1.4}$$

$$\sum_{i\in I} q_i x_{ij} \leqslant Q_j, \quad \forall j \in J, \tag{1.5}$$

$$x_{ij}, y_j \in \{0,1\}, \quad \forall i \in I, \quad \forall j \in J, \tag{1.6}$$

where n is the number of individuals, p the number of clusters, $p \geqslant \left\lceil \sum_{i\in I} q_i / \sum_{j\in J} Q_j \right\rceil$, $a_{i[]}$ are the components of an individual i with an array $i[]$ of its characteristics, in $\Re^l$, $||I|| \equiv n$, $||J|| \equiv p$, $d_{ij} = \left[\sum_{k=1}^{l} (a_{ik} - a_{jk})^2\right]^{1/2}$ the dissimilarity measure between i and its median j, x_{ij} is 1, if the individual i is assigned to cluster j and 0 otherwise, y_j is 1, if cluster j is used and 0 otherwise, I the set of individuals, J the set of medians, Q_j the maximum capacity of cluster j, q_i the demand of the individual i.

In the CCP model, the total dissimilarity in the clusters must be minimised (1.1). The number of clusters may not exceed the given number of clusters p (1.2). An individual is assigned to only one cluster (1.3). Every individual is assigned to a cluster (1.4). The cluster capacity must cover the demands of its individuals (1.5). Eq. (1.6) specifies the decision variables.

The above formulation is also known as the Capacitated p-Median Problem when Q_j is homogeneous, and was considered by [6–10].

For the CCP, the centre of a given cluster is a particular individual of the cluster from which the sum of the dissimilarities to all other individuals in the cluster is minimised (*scatter* of the cluster) [6]. In Fig. 1, we have the visualisation of a solution of a CCP instance. This instance of CCP considers 25 individuals with unit demand, where we wish to find 3 clusters with capacity limited to 9 individuals at most, with minimum dissimilarity in $\Re^2$ or $a_s = (a_{s1}, a_{s2})$, $\forall_s \in I$ or J. Note that the set J is formed by the medians (as the centres) of each cluster. The capacitated p-median problem is a special version of the CCP where the coefficients of the objective function are distances [9].

For large scale instances, which are found in many real life situations, it seems reasonable to use heuristic approaches to solve this problem. The heuristics for this problem must be designed in consequence of good strategies knowledge to find initial and improved solutions. Some authors consider, for solving CCP instances (>1000 individuals), the use of procedures that are, in fact, very time consuming, either in the

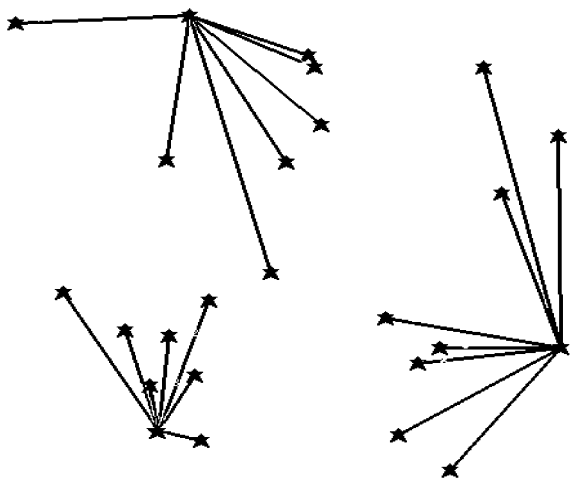


Fig. 1. CCP solution for an instance with 25 individuals and $Q_j = 9$, and $q_i = 1$.

constructive phase interactively using Regret functions, simulated annealing or Lagrangean relaxation using surrogate constraints or in the improvement phase, generally using tabu search metaheuristic and more recently GRAMPS (GRASP with adaptative memory programming) [3,5–7,10–14].

The capacitated (geo) centred clustering problem (CCCP), or continuous capacitated clustering problem, can be viewed as the problem of defining a set of clusters with limited capacity and minimum dissimilarity between the formed clusters, where each cluster has a centre located at the geometric centre of its individuals and covers all the demands of a set of individuals. It is different from the CCP, the clusters are centred at the centre of their individuals' co-ordinates, where for the CCP, the clusters are centred by their medians.

The CCCP has a wide range of applications such as: the design of garbage collection zones, sales force territorial design, depot location in distribution systems, location of switching centres in communication networks, location of off-shore platforms for oil exploration, clustering of customers into different marketing segments in marketing studies, taxation to municipalities, information systems design, routing newsboys to newspaper subscribers delivery, routing agents to dengue disease combat, and others [6,15–17].

In this article, we are particularly interested in the case where the CCCP problem is in $\Re^2$. We design and adapt heuristic/metaheuristic techniques that are taken from the literature. This problem is applied to the design of territories to sales force and garbage collection zones/circuits.

This article is organised as follows, in Section 2 we introduce the new problem and its formulation considering two aspects, when the number (p) of clusters is given and when we want to find the appropriate number of clusters to cover the individuals' demands. In Section 3 we show two log-polynomial heuristic algorithms for the generic capacitated clustering and p-CCCP. In Section 4, we show the results using instances from the CCP literature, considering variations of the constructive phase. Some results in the design of garbage collection zones, and sales force areas distribution using this approach implemented for the system SisRot (FULL and TRANSLIX), are also reported.

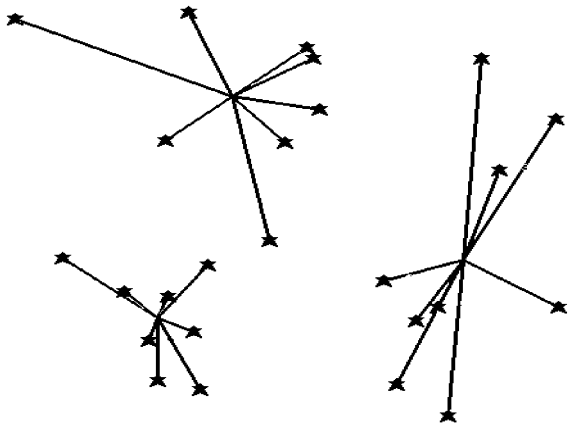


Fig. 2. Visualisation of the *p*-CCCP solution for the instance proposed in Fig. 1.

2. Modelling the CCCP

The CCCP consists in defining a set of clusters with limited capacity and maximum proper similarity per cluster. Each cluster is composed of individuals from whom we can compute a centre value and hence, determine a dissimilarity measure between clusters. The clusters must cover the demands of their individuals. In Fig. 2, we can see the same solution of the above instance, from a CCCP point of view.

We consider two different formulations. In the first one (*p*-CCCP), the number of clusters is known in advance and in the second one (*g*-CCCP), the number of clusters and the dissimilarities are minimised. The first formulation (*p*-CCCP) can be stated as follows:

$$(p\text{-CCCP}) \quad \text{Min} \sum_{i\in I}\sum_{j\in J} ||a_i - \overline{x_j}||^2 \; y_{ij} \tag{2.1}$$

such that,

$$\sum_{j\in J} y_{ij} = 1, \quad \forall i \in I, \tag{2.2}$$

$$\sum_{j\in J} y_{ij} = n_j, \quad \forall j \in J, \tag{2.3}$$

$$\sum_{i\in I} a_i y_{ij} = n_j \overline{x}_j, \quad \forall j \in J, \tag{2.4}$$

$$\sum_{i\in I} q_i y_{ij} \leqslant Q_j, \tag{2.5}$$

$$\bar{x}_j \in \Re^l, n_j \in N, y_{ij} \in \{0,1\}, \;\; \forall i \in I, \;\; \forall j \in J, \tag{2.6}$$

where, $\bar{x}_j$ is the centroid of a cluster j, n_j the number of individuals in cluster j, $y_{ij} = 1$, if the individual i is assined to cluster j, and 0 otherwise, a_i the position of the individual i in the $\Re^l$ space, Q_j the maximum

load of the cluster j, q_i the demand of the individual i, I the set of individual $s = n$, J the set of Clusters $= p$.

In the above model, the objective function minimises the sum of dissimilarities in each clusters (2.1). An individual is assigned to only one cluster (2.2). Eq. (2.3) gives the number of individuals in each cluster. Eq. (2.4) locates the centre of each cluster at its geometric centre. Eq. (2.5) maintains the demand of individuals limited to the capacity of each particular cluster (in our case, $Q_j = Q, \forall j \in J$), and the equations (2.6) define the decision variables, and the upper limits to the number of individuals per group.

The p-CCCP is slightly different from the CCP. In the p-CCCP, the centre of a cluster is not necessarily an individual, it is the centre value computed with respect to all the individuals of the same cluster. This introduces non-linearity in the model, in the objective function and in some set of constraints. This difference in the two models implies that for a given instance of a problem, the clusters found by a solution of the CCP are not necessarily the same as the ones found by the p-CCCP. To illustrate this, in Fig. 3(a)–(c) we have from the same set of points of the above instance, a resolution of an instance where $Q = 6$, $p = 5$, $q_i = 1$ $(i = 1, \ldots, 25)$.

The second formulation can be stated as follows (Generic CCCP):

$$\text{(Generic CCCP)} \quad \text{Min} \left(F \sum_{j \in J} z_j \right) + \sum_{j \in J} z_j \left(\sum_{i \in I} ||a_i - \overline{x_j}||^2 y_{ij} \right) \tag{2.7}$$

such that,

$$\sum_{j \in J} y_{ij} = 1, \quad \forall i \in I, \tag{2.8}$$

$$\sum_{i \in I} a_i y_{ij} = \bar{x}_j \left(\sum_{i \in I} y_{ij} \right), \quad \forall j \in J, \tag{2.9}$$

$$\sum_{i \in I} q_i y_{ij} \leqslant Q_j z_j, \quad \forall j \in J, \tag{2.10}$$

$$\bar{x}_j \in \Re^l, z_j, y_{ij} \in \{0, 1\}, \forall i \in I, \forall j \in J, \tag{2.11}$$

where $\bar{x}_j$ the centroid of a cluster j, $y_{ij} = 1$, if the individual i is assigned to cluster j, and 0 otherwise, $z_j = 1$, if a cluster j is open, and 0 otherwise, a_i the position of the individual i in the $\Re^l$ space, Q_j the capacity of the cluster j, q_i the demand of the individual i, I the set of individual, $||I|| = n$, J the set of possible clusters, $1 \leqslant ||J|| \leqslant n$, F the fixed cost for opening a cluster.

In the model above, the objective function minimises the number of clusters and the sum of dissimilarities of each of those clusters (2.7). An individual is assigned to only one cluster (2.8). Eq. (2.9) locates the centre of each cluster at its geometric centre. Eq. (2.10) maintains the demand of individuals limited to the capacity of each particular cluster (in our case, $Q_j = Q, \forall j \in J$), and in Eq. (2.11) we specify the decision variables.

The major difference in the second model is that we wish to find the number of clusters that covers, with minimum dissimilarity, all the individuals, with a capacity constraint for each cluster. A bin packing

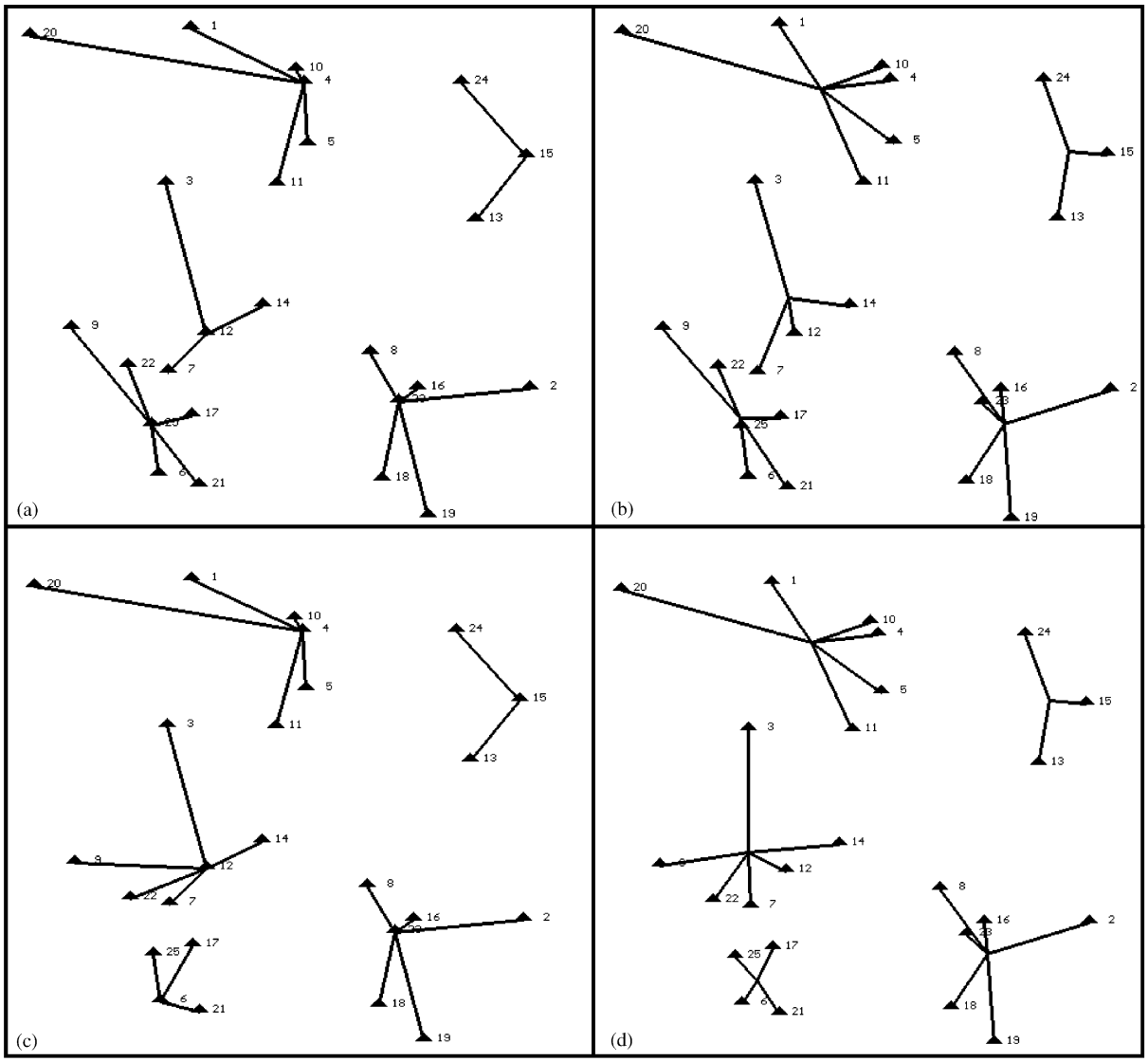


Fig. 3. Comparative visions on the same set of groups and instance for CCP and CCCP. (a) CCP Optimum Solution, $f_{\min} = 1216.62^2$. (b) Solution 3(a) in CCCP, $f_{\min} = 1273.46^2$. (c) Solution for the CCP, $f_{\min} = 1231.00^2$. (d) CCCP best known Solution, $f_{\min} = 1251.44^2$.

constraint is included in the clustering, and this constraint changes drastically the problem. We call it a generic centred capacitated clustering, since it has general purposes. The above formulations show how difficult this problem is, and as a consequence it can be shown that the CCP, and similarly the CCCP, can be polynomially reduced to a NP-Complete equivalent problem [18].

3. Heuristics for CCCP

The key aspects of the CCCP use the geometric positioning of the individuals in their related clusters. The problem of exchanging individuals from Neighbour clusters may be solved easily, if we find a good approximation of the clusters centres within a refinement phase. Considering this, we propose a two-phase algorithm where the first phase uses a constructive cluster procedure (Forgy algorithm, [19]). This procedure builds an initial solution oriented by a log-polynomial algorithm using structured geometric balanced q-trees. The second phase is a refinement of the variable neighbourhood search (VNS) [20,21].

For our method we also consider classic clustering methods from the literature. The unconstrained clustering is in itself NP-Complete, exact methods were proposed for short instances and some heuristic methods were also proposed to find feasible solutions with results of good quality [1,2,22–24]. All the methods consider an initial partition $g_1^0, \ldots, g_p^0$ of E and then [4], try to improve the solution exchanging the elements to its most promising group, for some criteria, and then recalculate the centre. The procedures continue until no improvement is achieved. These procedures are called Procedures of Means, in essence they have the same objective, but they differ by their strategies to find the final solution. The preliminary method of means is the Forgy's method (this method is also called H-Means), [19,21]:

3.1. The Forgy's method

Step 0: Set $k = 0$
Step 1: If $k = 0$, define $g_1^0, \ldots, g_p^0$ of E;
Otherwise, build a new $g_1^k, \ldots, g_p^k$ of E, assigning each individual to a group where its centre is closest. Set $k = k + 1$.
Step 2: Calculate the centroids of these groups.
Step 3: If $f(g_m^k) = f(g_m^{k-1})$, stop;
Otherwise, continue step 1.

An initial solution (partition) is given to the method as the centroids of this partition. In general, this partition is chosen at random, but with a little extra work in this initial step the method finds better results. The local minimum convergence is guaranteed, and the time required to achieve this result is $O(n)$. The problem of this method is the degeneracy, i.e. the number of final clusters formed may be different from the p desired. To avoid this, Hansen and Mladenovic (2001) proposed a step to isolate in clusters of a unique individual the difference between the number of achieved clusters (NAC) and p. The individuals are taken from the most distant to its centre to the least, in decreasing order. This step is included as follows (H-Means+).

3.2. The H-Means method

Step 0: Set $k = 0$
Step 1: If $k = 0$, define $g_1^0, \ldots, g_p^0$ of E;
Otherwise, build a new $g_1^k, \ldots, g_p^k$ of E, assigning each individual to a group where its centre is closest. Set $k = k + 1$.

Step 2: Calculate the centroids of the groups formed
Step 3: If NAC$\leqslant p$ then
Order the elements accordingly to the decreasing distance to its cluster centre; Take and isolate the first *p-NAC* groups ordered. Make the centroids be these elements;
Reduce $f(g_m^k)$ from the distances taken
Step 4: If $f(g_m^k) = f(g_m^{k-1})$, stop;
Otherwise, continue 1.

There is a difference between H-Means+ and the original Forgy's method. But nevertheless, in this text we will use the same denomination for the Forgy method and the non-degenerate Forgy, that is the H-Means+ method proposed by Hansen and Mladenovic (2001).

A variation of the Forgy's method is the well known KMeans method proposed by Jansen (1966) and later by MacQueen (1967). In the MacQueen's version we also have an initial given partition with p centres but the evaluation of the centres is a little different from the Forgy's method. In this method at each allocation of an individual to its group centre, a new evaluation of the centroids is made. It introduces a computational effort to the method as can be seen below.

3.3. The KMEANS method

Step 0: Set $k = 0$
Step 1: If $k = 0$, define $g_1^0, \ldots, g_p^0$ of E;
Step 2: Take the object k and attribute to a new group g_l, $(l \neq i)$, where the centroids would be obtained by [2]

$$\bar{x}_l = \frac{n_l \bar{x}_l - x_j}{n_l - 1} \quad \text{and} \quad \bar{x}_i = \frac{n_i \bar{x}_i - x_j}{n_i - 1},$$

where n_l is the cardinality of the group g_l, n_i the cardinality of the group g_i, $\bar{x}_l$ the centroid of the group g_l, $\bar{x}_i$ the centroid of the group C_i, x_j the individual j;
Step 3: Let v_{ji} is the improvement achieved by the objective function (measure of dissimilarity):

$$v_{ji} \leftarrow \frac{n_i}{n_i + 1} ||\bar{x}_i - x_j||^2 - \frac{n_l}{n_l - 1} ||\bar{x}_l - x_j||^2, \quad x_j \in g_l, \quad \bar{x}_i \notin g_l,$$

where

v_{ji} = improvement of the objective function by reallocating j;

Step 4: If no improvement was obtained, terminate (a local optimum was achieved) Otherwise, implement the greatest v_{ji}, go to step 2.

As the above methods, the JMeans version, proposed by Hansen and Mladenovic (2001), starts with a defined (random, p.ex.) p-partition. In this method, at each iteration, the individuals that are far from their group's centroid (with a tolerance ε) are reallocated to other centres that further improve the objective function. This strategy is much more elaborate than the others, although the method remains with the KMEANS complexity, O(n^2).

3.4. The JMeans method

Step 0: Set $k = 0$
Step 1: If $k = 0$, define $g_1^0, \ldots, g_p^0$ of E;
Step 2: Take objects of group g_i and attribute to a new group g_l, $(l \neq i)$. These objects must be far from their centroids for a given tolerance ε, mark them as non-occupied.
Step 3: (Jump neighbourhood). // exploring the neighbourhood.
For each $j\,(j = 1, \ldots, n)$ repeat the following steps:

(a) (Relocation). Add a new cluster centroid x_{M+1} at some unoccupied entity location x_j and the index i of the best centroid deletion; denote with v_{ij} the change in the objective function value;
(b) (Keep the best). Keep the pair of indices i' and j', where v_{ij} (defined in KMEANS) is minimum; (Move) Replace centroid $x_{i'}$ by $x_{j'}$ and update assignments accordingly to get the new partition P'_M; set $f' := \text{fopt} + v_{i'j'}$.

Step 4: If no improvement was obtained, terminate (a local optimum was achieved) Otherwise, implement the greatest *partition*, go to step 2.

Many other methods are studied by the literature using other strategies or other techniques from neural networks (Kohonen), fuzzy sets to alternative *c*-means. Major related literature in cluster methods and applications can be viewed in IEEE Transactions in Pattern Analysis and Machine Intelligence, and in Pattern Recognition. For a mathematical programming overview on clustering, refer to [4]. For a better review of the state-of-the-art unconstrained methods, refer to [25–30].

3.5. Phase 1: Constructive step

In this phase, the coordinates of each individual are included in a balanced *q*-tree structure. As the points (individuals) are included in the tree, the structure self-adjusts, to position all the roots of sub-trees approximately as the 1-median of their sub-trees (points already included). Some rotations, based on AVL-trees and extended to 2-dimentional data structures, are used to balance the sub-trees. This algorithm is O(log *n*) for the insertion and for the deletion parts [20,31]. The purpose of this algorithm is to define precisely the Neighbourhood of the individuals, and to map as accurately as possible the initial cluster solution for a given constructive/improvement cluster algorithm such as the Forgy Algorithm or the JMeans Algorithm [19,21].

The guided partitioning phase can be viewed in Fig. 4. First, the individuals are positioned in the *q*-tree according to their relative position (quadrants). As the individuals are placed, the tree is adjusted (rotated) to balance its nodes. This process is similar to adjusting sons in the tree by the approximate median of each internal node.

The two cluster methods (the Forgy method and the JMEANS method), described above in this section, can be considered as improvement type algorithms, and encompass the idea of trying to relocate individuals from their initial cluster to a new cluster, until the individuals' positions "minimise" the total dissimilarity [21]. All these types of methods can be used in cascade, Forgy and then JMeans or vice versa, and they can give different results if we consider the initial partition given to any of them. To observe this phenomenon applied to our case, we decided to build the algorithms using our structured *q*-tree

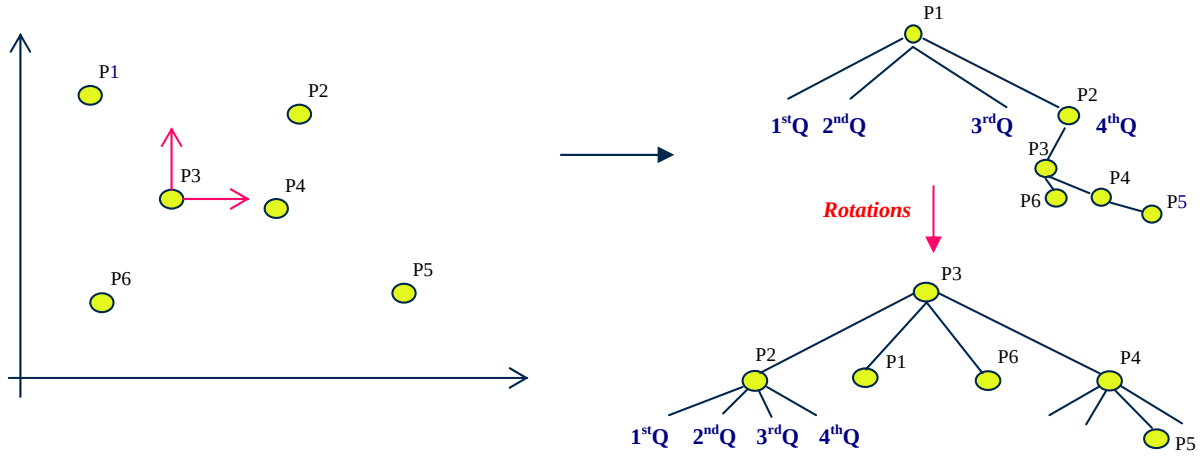


Fig. 4. A view of the *q*-tree procedure.

procedure as the starting step of those algorithms. The procedures were tested for a set of instances where the individuals are in $\Re^2$, and are uniform randomly distributed in the space where $X \in [-50, 4950]$ and $Y \in [-390, 4610]$. The results are shown in Table 1, which shows the objective function (dissimilarity measure), the number of iterations and the time to find a local optimum solution.

In Table 1, the algorithms are named in the following way: Forgy—Forgy with a random starting solution, JMeans—JMEANS with a random starting solution; Forgy A—Forgy with the *q*-tree solution as the starting solution; JMeans A—JMeans with the *q*-tree solution as the initial solution.

When starting with *q*-tree solution, we divide the number of individuals by the number of *p*-clusters and distribute this result between the *p*-clusters. For the tests, we made 10 different instances (samples) for each number of points $(1000, 2000, \ldots, 5000)$. The clusters are built for the first three tests (with 1000, 2000 and 3000 points/individuals), we can see the best dissimilarity solution for the Forgy A in almost all results. These methods are working in cascade, and it is only outperformed by another method (JMeans) in the tests with 3000 and 5000 points/individuals, although the results from JMeans A and Forgy A are very close.

Another important observation is the number of iterations. The Forgy method with an initial random solution does not give good results. However, the Forgy method combined with the structured *q*-tree procedure quickly provides an initial solution for the other methods. The resulting solutions obtained are very interesting in terms of time and dissimilarities. Hence, we can see, that for the majority of methods, the results produced are much better in terms of time, number of iterations and total dissimilarity if used with a Forgy method combined with the structured *q*-tree procedure as an initial solution. We extend these results for the capacitated centred clustering.

The unconstrained clustering methods can be modified to encompass the load constraint by including, before the exchanging step, a check of the exchange feasibility. In this step, the load of the individual is removed from the source cluster and relocated to the target cluster. If this is feasible and there is any improvement in the objective function, the exchange is performed.

For the capacitated case, we initially process the geometric organisation and positioning of the individuals using a *q*-tree. We use two different strategies in building feasible initial clusters. In the first

Table 1
Evaluation of the effect in better partitioning and cascading Means unconstrained min-sum-of-squares cluster methods (average of 10 samples for each number of points)

Number of individuals	Method	Fobj	Iter	Time (s)
1000	Forgy	20477.89	24	0.27
	Forgy A	20412.94	27	0.31
	JMeans	20462.31	1191	74.57
	JMeans A	20435.54	658	41.36
2000	Forgy	29006.09	35	0.76
	Forgy A	28985.54	31	0.68
	JMeans	28941.49	2682	649.14
	JMeans A	29017.52	1353	327.64
3000	Forgy	35584.30	40	1.27
	Forgy A	35624.22	32	1.04
	JMeans	35575.93	3951	2116.03
	JMeans A	35525.99	1871	1005.57
4000	Forgy	40942.23	45	1.90
	Forgy A	40913.63	49	2.08
	JMeans	40979.56	5614	5322.88
	JMeans A	40979.02	2864	2720.84
5000	Forgy	45900.40	42	2.19
	Forgy A	45898.12	47	2.51
	JMeans	45891.85	7035	10370.62
	JMeans A	45927.05	3462	5097.71

strategy (Next Fit), we consider a first cluster, insert the root in the actual cluster, and delete it from the q-tree. We continue this step until the cluster capacity is reached. If the actual root overloads the present cluster capacity, we open another cluster and proceed with the procedure until all individuals are clustered (q-tree is empty). In the second (Best Fit), we start as in the first strategy, but once there are more than one cluster and a root candidate to be inserted, the procedure searches for the closest feasible insertion cluster (closest considering the root and the ith cluster centre). All these strategies work like the classical heuristics from the bin packing problem (Next Fit and Best Fit). The meaning of best here is the closest centre [32].

After this step, once the cluster(s) is/are feasible, we execute the algorithms Forgy (H-Means+) and/or JMEANS, both constrained in the exchange phase.

The procedure CAPCluster in Fig. 6, does not take into account simultaneously the two proposed problems, p-CCCP and g-CCCP. To consider this, we build another method based on the unconstrained clustering initial solution. The major idea is: from a solution produced by the unconstrained clustering step, if it is not feasible make it feasible and try to improve the feasible clusters found using CAPMEANS(). Fig. 7 shows the UCCAPCluster procedure which proceeds like this.

```
1.  Procedure NextFit ;
2.  // Building feasible clusters with the first strategy using q-tree
3.  // I (In) : Set of Individuals
4.  // Qmax : Maximum cluster demand
5.  // q (In) : demand vector of individuals
6.  // Q (Out) : load vector of clusters
7.  // IC (Out) : Assignment vector of individual to a cluster
8.  // NC (Out) : Final number of clusters
9.  for ∀i ∈ I do
10.   call q-tree_Insert(i); // Insert individual in the balanced q-tree
11.  j:=1; Qj:=0;
12.  repeat
13.    i:=q-tree_root(); // Take the root from q-tree generated
14.    If Qj+qi≤ Qmax Then
15.       call insert_individual_in_cluster(j, i, ICj, Qj , qi);
16.    else
17.      begin // Open a new cluster
18.         j:=j+1;
19.         Qj:=0;
20.      end;
21.   call q-tree_Delete(i); // Remove the actual root
22.  until q-tree_empty(); // Check if the q-tree is empty
23.  NClusters:=j; // Final number of clusters
24.  return (Q, IC,NC)
25.  end NextFit
```

```
1.  Procedure BestFit ;
2.  // Building feasible clusters with the second strategy using
    q-tree
3.  // I (In) : Set of Individuals
4.  // Qmax : Maximum cluster demand
5.  // q (In) : demand vector of individuals
6.  // Q (Out) : load vector of clusters
7.  // IC (Out) : Assignment vector of individual to a cluster
8.  // NC (Out) : Final number of clusters
9.  for ∀i ∈ I do
10.   call q-tree_Insert(i); // Insert in the balanced q-tree
11.  j:=1; Qj:=0;
12.  repeat
13.    i:=q-tree_root(); // Take the root from q-tree generated
14.    // Find Closest feasible open cluster
15.    Feasible :=FindBestOpenCluster(j, qi);
16.    If Feasible Then
17.       call insert_individual_in_cluster(j, i, ICj, Qj , qi);
18.    else
19.    // Open a new cluster for the incoming individual
20.      begin
21.         j:=j+1;
22.         Qj:=0;
23.         call insert_individual_in_cluster(j, i, ICj, Qj , qi);
24.      end;
25.   call q-tree_Delete(i); // Remove the actual root
26.  until q-tree_empty(); // Check if the q-tree is empty
27.  NClusters:=j; // Final number of clusters
28.  return (Q, IC,NC)
29.  end BestFit
```

Fig. 5. Next Fit and Best Fit algorithms using a structured balance *q*-tree.

```
1.  Procedure GenericCAPCluster
2.  // A constructive procedure for the g-CCCP
3.  // Build initial feasible clusters
4.  call NextFit(); // or BestFit();
5.   // Improvement phase using methods of Means (Forgy, Jmeans)
6.   call Dissimilarity(IC,Objective_function); // Calculate centres and dissimilarity
7.   call CAPMEANS();  // Calculate avoid generating the initial partition
8.  end // GenericCAPCluster
```

Fig. 6. A constructive procedure for the *g*-CCCP.

For simplicity our proposed method CAPCluster can be viewed below in Figs. 5 and 6:

The UCCAPCluster procedure in Fig. 7 (UC-unconstrained to constrained), shows between lines 17–37 that the individual which overloads the cluster is relocated to a new feasible cluster, closest to this individual. If no cluster can be used, a new one is opened. Line 9 creates a cluster vector with the lists of their elements, from the solution generated by the unconstrained non-degenerate procedure, H-Means+. In line 10, there is an ordering of all clusters by the decreasing demand of their individuals (vector of lists, MatIC). In this ordering process, one can use the distance or the distance/demand criteria, instead of the demand only, the use of these criteria can change the results, depending on the instance.

```
1.  Procedure UCCAPCluster;
2.  // A constructive procedure for the g-CCCP and p-CCCP
3.  if p is not given then
4.    g:=BestFit(); // Number of clusters produced by BestFit
5.  else
6.     g:=p;
7.  // Unconstrained Phase
8.  call Forgy(g); // Build clusters using H-Means+ with g-clusters
9.  call Build(MatIC);//Cluster vector with the list of its individuals and their demands
10. call Order(MatIC);//Order each cluster by the decreasing demand of its individuals
11. If Clusters are not feasible Then
12.   for j := 1 to g do
13.     If MatIC[j].Q_total ≤ Q_max Then // Reallocate the infeasible individuals
14.       begin
15.         L_j:=0;
16.         for i:=1 to NIC[j] do
17.             If MatIC[j,i].q + L_j ≤ Q_max Then
18.               begin
19.                  Feasible:=FindBestOpenCluster(k, q_i);
20.                  If Feasible Then
21.                    call insert_individual_in_cluster(k, i, IC_j, Q_j , q_i);
22.                  else
23.                  // Open a new cluster for the incoming individual
24.                   begin
25.                      g := g + 1;
26.                      Q_j := 0;
27.                      call insert_individual_in_cluster(g, i, IC_j, Q_j , q_i);
28.                   end
29.               end;
30.               else
31.                  L_j:= L_j + MatIC[j,i].q;
32.       end; // for j
33. // Improvement phase using methods of Means (Forgy, Jmeans)
34. call Dissimilarity(IC,Objective_function); // Calculate centres and dissimilarity
35. call CAPMEANS();  // Calculate avoid generating the initial partition
36. end // GenericCAPCluster
```

Fig. 7. UCCapCluster algorithm for *p/g*-CCCP instances.

Line 4 finds the number of clusters at minimum or close to the minimum number, taking into account the *g*-CCCP model. We found by experimentation that the Best Fit() procedure obtains least/equal number of clusters than Next Fit(), see results below.

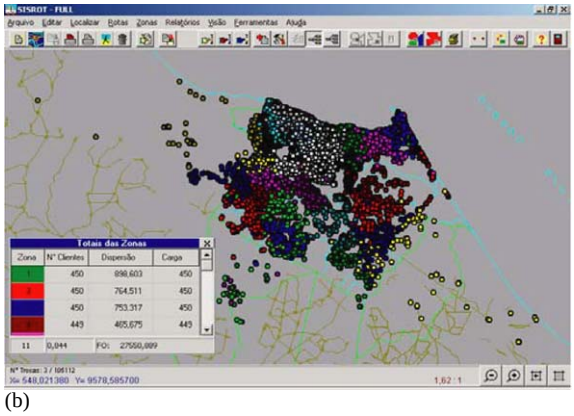
3.6. Phase 2: Metaheuristic step (VNS)

This step is processed independently, after analysing the clusters obtained in the first phase. We define the type of exchanges (number of individuals selected to proceed the exchange), and the time allowed for each exchange in the VNS algorithm. We build a VNS procedure, that considers two major strategies:

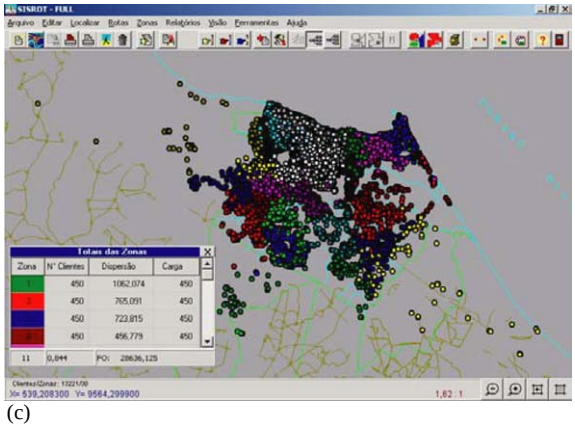
1. Closest centre: Randomly choosing the individuals, and trying to insert them in the closest cluster;

```
1.  Procedure VNS(Individuals, Groups,Initial_Objective_Value)
2.  Objective_function:=Initial_Objective_Value;
3.   while time < selected_time do
4.   begin
5.    call Select_Individuals();
6.    call Select_Clusters();
7.    call try_exchange();
8.    New_Objective:=evaluate_new_objective();
9.    If New_Objective < Objective_Function Then
10.   begin
11.    call implement("try_exchange()");
12.    Objective_Function:=New_Objective;
13.    Gain:= Initial_Objective_Value - Objective_Function;
14.   end
15.  else
16.   call desimplement("try_exchange()");
17.  end
18. end // VNS
```

(a)



(b)



(c)

Fig. 8. (a) VNS algorithm as the final improvement phase. (b,c) Evaluation of the *q*-tree+H-Means+ and then VNS with 1-exchange, 2-exchange and 3-exchange for an instance of 13,221 individuals and 30 groups.

2. Random centre: Randomly choosing the individuals, and trying to insert them in randomly selected clusters.

The proposed VNS algorithm, can be written generically as follows (Fig. 8):

In the proposed VNS procedure, the following set of procedures does as follows:

1. Select_Individuals()—according to the type of exchanges, a specified number of random distinct individual(s) is/are selected;
2. Select_Clusters()—according to the type of strategy, a specified number of clusters (equal to the number of individuals but not necessarily distinct), are selected for each individual to be moved to;
3. Try_exchange()—exchange selected individuals between the clusters;
4. Evaluate_new_objective()—calculate new dissimilarity and store the exchanges (individuals to clusters) if they are all feasible;
5. Implement()—proceed with the exchange, recalculating centres, and fixing the new solution proposed by try_exchange();
6. Desimplement()—restore the solution as it was in the previous iteration proposed by try_exchange().

In essence it is the same algorithm proposed by Hansen and Mladnovic [33], the difference here is in the implementation of lines 5–7. In our case, we fix the choice of the number of exchanges. According to the appropriate strategy used, instead of selecting a random k source individuals to k target clusters at each iteration, we maintain k fixed for all iterations, we call this fixed k-exchange. The experimental results obtained on the instances we have for this problem, suggested that fixing k was a good strategy. It can also be implemented as the original VNS algorithm [21].

VNS is a type of Monte Carlo integrated to a Neighbour search algorithm that is very simple and appropriate for this problem, since making and returning the exchangeable tries is easy to implement. The evaluation time is in the order of O(1).

4. Results from literature and "real" applications

We developed a first set of instances (TA) and test some other kindly offered by Professor Luiz Antônio Lorena, for the Capacitated p-Median problem [9]. For our tests, we use the cost function (dissimilarity) calculated for the CCP instances using the column generation program produced by Lorena and Senne [9]. We consider CCP costs, once its instances optimal/near optimal objective values are at the same order of magnitude to the CCCP for the same p or g. All the tests were made using a PC-AMD ATHLON 1.6 GHz 512 MB RAM, and all the instances are available at http://www.lcc.uece.br/∼negreiro/artigos/cccp.

In Table 2, we can see a set of evaluations for the procedure UCCAPCluster, for a number of instances TA specially generated for given demands ($q_i = 1$) and specific given capacities Q. The tests from Lorena and Senne [9], are the set of instances initiated by SJC and P3038. Our results return the g value achieved for each instance and strategy (Next Fit and Best Fit), we also see CCP related instance best feasible cost known ($g = p$), and for the VNS (1+1) phase with 1-exchange and then fixed 2-exchange, each running in 1 min. The percentage of improvement achieved between the construction phase and VNS phase are reported (imp%).

The set of TA instances do not appear in the Next Fit tests, since the same results were obtained in Best Fit. The Best Fit strategy shows how good it is in the sense of constructing the least number of clusters between the two strategies. For the CCP $\times$ g-CCCP results (Gap%), for the same p, the g-CCCP objective cost is $[-0.48, 28.96]$% far from the results of CCP produced by the Lagrangean/surrogate method, for the TA and SCJ instances, where the bounds were obtained using CPLEX [9,34].

Table 2
Results from UCCAPCluster × Lorena's considering a set of instances from the literature, and built for our evaluation

Instance	No points	*Q*	*P*	CCP	*p*-CCCP	Time (s)	VNS (1+1)	Imp %	GAP %
Next Fit									
p3038_600	3038	321	600	122020.66	192575.70	20.81	192024.83	0.28	57.37
p3038_700	3038	273	700	108653.04	176731.07	16.07	176731.07	0.00	38.52
p3038_800	3038	238	800	98483.26	184502.38	17.15	184502.38	0.00	87.34
p3038_900	3038	216	900	90131.62	176781.51	26.93	176781.51	0.00	96.13
p3038_1000	3038	191	1000	83012.98	159139.89	23.34	159139.89	0.00	91.70
SJC1_dat	100	720	10	17288.99	19429.459	0.020	17696.53	8.91	2.30
SJC2_dat	200	840	15	33370.20	35322.476	0.020	33423.84	5.37	0.16
SJC3a_dat	300	740	25	45335.16	50254.310	0.078	47985.29	4.51	5.52
SJC4a_dat	402	840	30	62026.94	76352.451	0.078	66689.96	12.65	6.99
Best Fit									
TA25	25	6	5	1216.03	1528.64	0.00	1257.51	17.73	3.30
TA50	50	11	5	4429.06	4605.03	0.00	4485.44	2.59	1.26
TA60	60	13	5	5357.36	5475.96	0.00	5391.47	1.54	0.63
TA70	70	17	5	6203.44	6292.06	0.00	6275.99	2.55	1.16
TA80	80	12	7	4153.64	5918.44	0.00	5846.94	1.20	28.96
TA90	90	23	4	9032.90	9646.38	0.00	9134.69	5.30	1.11
TA100	100	17	6	8181.04	8270.26	0.00	8141.70	1.55	−0.48
SJC2_12_dat	200	840	12	38796.80	49094.94	0.020	41949.56	14.55	7.52
SJC3a_16_dat	300	740	16	64129.20	96579.12	0.020	72545.49	24.88	11.60
SJC4a_21_dat	402	840	21	–xxx–	88981.33	0.047	87367.94	1.81	–xxx–

For all P3038, we do not have the results for the *g*-CCCP from Lorena's program ($p = g$), the results reported are from the paper [9]. The great difference between obtained results may be further investigated.

For the comparison Next Fit × Best Fit, Table 3, the difference between both methods in the number of clusters formed can be noticed. For example, the difference for the instance P3038/$Q_{\max} = 191$, where Next Fit and Best Fit results differs in more the 10% in the number of clusters.

Considering the effect of time to obtain good solutions, Table 3, we have for UCCAPCluster heuristic a performance that is less time consuming. Very high scale instances can be hold, and the method can be used to obtain initial solutions that are 0–50% to the best known for the set of instances tested. For CCP, pex., the instance SJC4a_dat, the method proposed by Lorena and Senne [9] takes 1.34 h to solve the instance. For this particular instance, the difference between CCP cost and *g*-CCCP is 6,99%, with 2 min of VNS.

We also made our tests for instances from real meaning, garbage collection and sales force territorial design. For this cases we develop a set of instances that are related to these two applications to evaluate the progress of our algorithm.

The second set of instances (7), was formed from sub-sets of 13,221 customers of a food distributor that operates in the metropolitan area of Fortaleza, capital of Ceará State in Brazil. The demand of an

Table 3
Comparison between proposed methods for g-CCCP case

Lorena's instances								
SJC1	Next Fit				Best Fit			
NI: 100, $Q = 720$	$G = 10$				$g = 9/10$			
Method	Obj func	Time (s)	VNS (1+1)		Obj func	Time (s)	VNS (1+1)	
q-Tree + Forgy	21782.34	0.00	19493.53	10.51%	(9) 27462.61	0.00	20495.54	25.36%
UCCAPCluster	19429.45	0.00	17696.53	8.91%	(10)19131.00	0.01	18005.17	5.88%
SJC2	Next Fit				Best Fit			
NI: 200, $Q = 204$	$g = 12/13$				$g = 12$			
Method	Obj func	Time (s)	VNS (1+1)		Obj func	Time (s)	+VNS (1+1)	
q-Tree + Forgy	(12)49180.31	0.02	39576.09	19.52%	79414.83	0.00	41171.34	48.15%
UCCAPCluster	(13)39110.14	0.02	36955.71	5.50%	49094.94	0.02	41949.56	14.55%
SJC3	Next Fit				Best Fit			
NI: 300, $Q = 740$	$G = 17$				$g = 16$			
Method	Obj func	Time (s)	VNS (1+1)		Obj func	Time (s)	VNS (1+1)	
q-Tree + Forgy	81567.71	0.03	62912.61	22.87%	108936.88	0.01	68354.34	37.25%
UCCAPCluster	61285.36	0.06	59266.17	3.29%	96579.12	0.02	72545.49	24.88%
SJC4	Next Fit				Best Fit			
NI:402, $Q = 840$	$G = 21/23$				$g = 20/21$			
Method	Obj func	Time (s)	VNS (1+1)		Obj func	Time (s)	VNS (1+1)	
q-Tree + Forgy	(21)98257.97	0.03	84165.56	14.34%	(20)126763.14	0.04	97444.74	23.12%
UCCAPCluster	(23)79580.77	0.04	77976.48	2.01%	(21)88981.03	0.04	87367.94	1.18%
P3038_600	Next Fit				Best Fit			
NI: 3038, $Q = 321$	$g = 538$				$g = 498$			
Method	Obj func	Time (s)	VNS (1+1)		Obj func	Time (s)	VNS (1+1)	
q-Tree + Forgy	199648.18	7.95	198065.35	0.79%	541160.19	9.57	535613.70	10.24%
UCCAPCluster	193788.72	12.76	193788.72	0.00%	372170.64	12.21	371771.22	0.05%
P3038_700	Next Fit				Best Fit			
NI: 3038, $Q = 273$	$g = 639$				$g = 582$			
Method	Obj func	Time (s)	VNS (1+1)		Obj func	Time (s)	VNS (1+1)	
q-Tree+ Forgy	198309.60	7.90	196817.68	0.75%	508272.45	11.76	507966.93	0.00%
UCCAPCluster	178441.39	12.5	178441.39	0.00%	365525.68	11.50	363243.31	0.62%
P3038_800	Next Fit				Best Fit			
NI: 3038, $Q = 238$	$g = 745$				$g = 669$			
Method	Obj func	Time (s)	VNS (1+1)		Obj func	Time (s)	VNS (1+1)	
q-Tree + Forgy	174057.10	8.08	174057.10	0.00%	538082.35	10.23	535195.53	0.53%
UCCAPCluster	169633.94	13.09	169633.94	0.00%	354653.42	15.34	354653.42	0.00%

Table 3 (*continued*)

Lorena's instances								
SJC1	Next Fit				Best Fit			
P3038_900	Next Fit				Best Fit			
NI: 3038, $Q = 216$	$g = 853$				$g = 761$			
Method	Obj func	Time (s)	VNS (1+1)		Obj func	Time (s)	VNS (1+1)	
q-Tree + Forgy	156165.36	7.14	156165.36	0.00%	421702.32	10.15	420789.82	0.21%
UCCAPCluster	144180.41	13.43	144180.41	0.00%	303916.41	16.28	303916.41	0.00%
P3038_1000	Next Fit				Best Fit			
NI: 3038, $Q = 191$	$g = 966$				$g = 846/849$			
Method	Obj func	Time (s)	VNS (1+1)		Obj func	Time (s)	VNS (1+1)	
q-Tree + Forgy	146585.25	9.98	146585.25	0.00%	(846)445340.69	9.75	443726.45	0.36%
UCCAPCluster	131445.91	11.81	134396.99	0.00%	(849)291294.55	18.93	291294.55	0.00%
TA instances								
TA25	Next Fit				Best Fit			
NI: 25, $Q = 6$	$g = 6$				$g = 5$			
Method	Obj func	Time (s)	VNS (1+1)		Obj func	Time (s)	VNS (1+1)	
q-Tree + Forgy	2510.46	0.00	1256.62	49.99%	2341.42	0.00	1256.62	46.33%
UCCAPCluster	1717.05	0.00	1251.44	27.11%	1528.64	0.00	1257.51	17.73%
TA50	Next Fit				Best Fit			
NI: 50, $Q = 11$	$g = 5$				$g = 5$			
Method	Obj func	Time (s)	VNS (1+1)		Obj func	Time (s)	VNS (1+1)	
q-Tree + Forgy	6848.86	0.00	4493.09	34.39%	6848.86	0.00	4498.72	34.31%
UCCAPCluster	5324.71	0.00	4476.12	15.93%	4605.03	0.00	4485.44	2.59%
TA60	Next Fit				Best Fit			
NI: 60, $Q = 13$	$g = 5$				$g = 5$			
Method	Obj func	Time (s)	VNS (1+1)		Obj func	Time (s)	VNS (1+1)	
q-Tree + Forgy	7942.48	0.00	5386.74	32.17%	7942.48	0.00	5390.73	32.12%
UCCAPCluster	6179.57	0.01	5356.58	13.31%	5475.96	0.00	5391.47	1.54%
TA70	Next Fit				Best Fit			
NI: 70, $Q = 17$	$g = 5$				$g = 5$			
Method	Obj func	Time (s)	VNS (1+1)		Obj func	Time (s)	VNS (1+1)	
q-Tree + Forgy	8900.64	0.00	6241.55	29.87%	8900.64	0.00	6241.55	29.87%
UCCAPCluster	6463.82	0.00	6241.55	3.43%	6292.06	0.00	6275.99	2.55%
TA80	Next Fit				Best Fit			
NI: 80, $Q = 12$	$g = 7$				$g = 7$			
Method	Obj func	Time (s)	VNS (1+1)		Obj func	Time (s)	VNS (1+1)	
q-Tree + Forgy	10178.94	0.00	5730.28	43.70%	10178.94	0.00	5730.28	43.70%

Table 3 (*continued*)

Lorena's instances								
UCCAPCluster	5913.44	0.00	5730.28	3.09%	5918.44	0.00	5846.94	1.20%
TA90	Next Fit				Best Fit			
NI: 23, $Q = 23$	$g = 4$				$g = 4$			
Method	Obj func	Time (s)	VNS (1+1)		Obj func	Time (s)	VNS (1+1)	
q-Tree + Forgy	11176.82	0.00	9103.21	18.55%	11176.82	0.00	9103.21	18.55%
UCCAPCluster	9979.46	0.00	9103.21	8.78%	9646.38	0.00	9134.69	5.30%
TA100	Next Fit				Best Fit			
NI: 100, $Q = 17$	$g = 6$				$g = 6$			
Method	Obj func	Time (s)	VNS (1+1)		Obj func	Time (s)	VNS (1+1)	
q-Tree + Forgy	13033.27	0.00	8128.38	37.63%	13033.27	0.00	8283.57	36.44%
UCCAPCluster	8533.86	0.00	8122.67	4.81%	8270.26	0.00	8141.70	1.55%

individual is one unit of service, and all individuals must be covered by a district. For this instance, capacity means a number of visited customers per month. Fig. 8(a)–(c) show the evolution of the process of calculating capacitated sales force districts in the Fortaleza's area.

Table 4 compares two methods, Forgy (q-tree+H-Means+) and UCCAPCluster. The VNS phase was applied to test if any improvements were made (running for 5 min in a static 2-exchanges). The objective function is the total dissimilarity as in the general CCCP model. Below the VNS title is the improvement achieved by using the VNS for the same methods above. As we can see, the Forgy+VNS dominates in the instances from DONI1 to DONI6. In DONI2 instance, Best Fit strategy for modified Forgy reaches the least number of clusters possible for it (5).

In Table 4, the improvement achieved in large instances is very small as we maintain the time as the other instances for the VNS step. For evaluating if VNS can obtain better results if we give more time, we experimented for the instance DONI7 (13,221 customers) 20 min after the solution obtained by modified Forgy with Best Fit. In the experiment, from the initial solution we proceed first the 1-exchange (running for 5 min, 3.13% improvement, 175 success from 92,895 tries), 2-exchanges (10 min, 0.62% improvement from the q-tree+Forgy solution, 35 success from 205,385 tries) and finally 3-exchanges (5 min, 0.06% improvement from the last improved result, 3 success from 105,109 tries)–VNS (5+10+5). Note that even for this time the solution is still far from the best known 23478.79, Table 4. Fig. 9 shows the decreasing steps of the objective function using VNS, note that in Table 4 we find a better result. This experiment shows that the VNS step may obtain very different results, and it is not stable for these type of instances.

In the third set of instances, we have a set of unoriented and oriented street segments from Fortaleza/CE, where each segment demands a weight uniformly and randomly distributed between 0 and 100 kg per segment. All segments are identified by their centroids, and the weights of the segments are assigned to their centroid. The total load of the segments for these instances is $Q_{\text{total}} = 190.001$ kg (3780 centroids) and $Q_{\text{total}} = 2034.699$ kg (40,919 centroids). We choose vehicles' areas weighted with a maximum of 30.000 kg (local load packers can perform daily 3 trips of 10 t, or 4 trips of 7.5 t).

Table 4
Set of instances from sales force districting

DONI1	Next Fit				Best Fit			
NI: 1000, $Q = 200$	$g = 6$				$g = 6$			
Method	Obj func	Time (s)	VNS 5 min		Obj func	Time (s)	VNS 5 min	
q-Tree + Forgy	3950.91	0.00	3072.80	22.22%	3577.04	0.00	3065.64	28.10%
UCCAPCluster	3590.91	0.07	3021.41	15.86%	3577.04	0.04	3026.27	15.40%
DONI2	Next Fit				Best Fit			
NI: 2000, $Q = 400$	$g = 6$				$g = 5/6$			
Method	Obj func	Time (s)	VNS 5 min		Obj func	Time (s)	VNS 5 min	
q-Tree + Forgy	7596.84	0.04	6080.70	19.95%	(5) 9795.03	0.00	7596.91	22.44%
UCCAPCluster	7514.13	0.10	6449.11	14.18%	(6) 8405.21	0.01	6412.13	23.71%
DONI3	Next Fit				Best Fit			
NI: 3000, $Q = 400$	$g = 8$				$g = 8$			
Method	Obj func	Time (s)	VNS 5 min		Obj func	Time (s)	VNS 5 min	
q-Tree + Forgy	9575.20	0.04	8786.60	8.23%	9575.20	0.03	8769.05	8.41%
UCCAPCluster	11587.71	0.18	8989.92	22.42%	11587.71	0.17	9000.74	22.33%
DONI4	Next Fit				Best Fit			
NI: 4000, $Q = 400$	$g = 10$				$g = 10$			
Method	Obj func	Time (s)	VNS 5 min		Obj func	Time (s)	VNS 5 min	
q-Tree + Forgy	18341.50	0.02	11639.93	36.53%	18341.50	0.02	11516.14	37.21%
UCCAPCluster	15934.29	0.54	14626.69	8.21%	15934.29	0.54	14633.45	8.16%
DONI5	Next Fit				Best Fit			
NI: 5000, $Q = 450$	$g = 12$				$g = 12$			
Method	Obj func	Time (s)	VNS 5 min		Obj func	Time (s)	VNS 5 min	
q-Tree + Forgy	11848.74	0.20	11635.18	1.80%	12603.80	0.14	11929.30	5.35%
UCCAPCluster	14459.60	0.60	12605.70	12.82%	12965.86	0.42	12254.97	5.48%
DONI6	Next Fit				Best Fit			
NI: 10000, $Q = 450$	$g = 23$				$g = 23$			
Method	Obj func	Time (s)	VNS 5 min		Obj func	Time (s)	VNS 5 min	
q-Tree + Forgy	20117.20	0.50	18443.50	8.31%	24672.55	0.31	20038.29	18.78%
UCCAPCluster	23449.39	2.21	23289.81	0.68%	23437.21	2.34	23121.46	1.35%
DONI7	Next Fit				Best Fit			
NI: 13221, $Q = 450$	$g = 30$				$g = 30$			
Method	Obj func	Time (s)	VNS (5+10)		Obj func	Time (s)	VNS (5+10)	
q-Tree + Forgy	30896.43	0.45	26447.37	14.39%	28636.09	0.84	25010.48	11.10%
UCCAPCluster	25341.22	3.09	24208.53	5.15%	24553.51	3.36	23478.79	4.37%

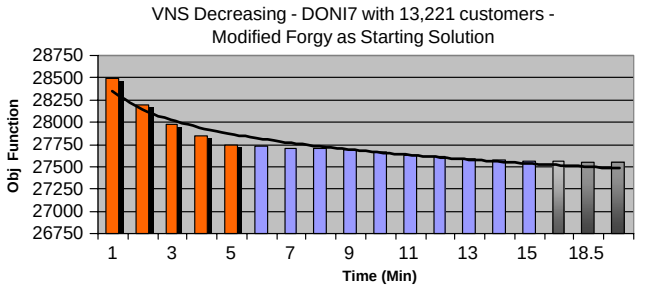


Fig. 9. VNS running for 20 min, and the value of the objective function using 1–3 exchanges—in shadow 1-exchange results, in grey 2-exchange and in degrade 3-exchange.

Figs. 10(a) and (b) show the evaluation process of calculating capacitated garbage collection zones, using UCCAPCluster approach for a sub-area of Fortaleza. In this evaluation, we can see what happens with the initial feasible partition using *UCCAPCluster* algorithm (a), then the solution obtained by VNS, after 60 min of 1-exchange, 60 min of 2-exchanges. In this instance, we have $p = 7$ for every case and the load distribution can change between clusters. The achieved improvement is 0.28%.

Figs. 11(a) and (b) show another evaluation process for all the city's street segments. In this evaluation, we can also see what happens with the initial feasible solution using only the UCCAPCluster. We have $p = 68$ clusters and the load distribution can be flexible between them. We avoid using VNS in this case.

In Table 5, we have the results for the proposed UCCAPCluster algorithm. For a better view, we include the same elements proposed by Table 3, now, considering the instances from Figs. 10 and 11.

We have also done, for these instances, tests to evaluate the most appropriate way of processing the VNS phase. We also tried mixed exchanges, 1–4 exchanges, randomly selected (as the original VNS, [21]), but the results obtained are much poorer than the ones reported here. Because of this, we think that the best way to use VNS for these instances and approaches is in the sequential form, or exchanges one at a time.

5. Conclusion

In this work we propose a new problem we call the capacitated centred clustering problem (CCCP), or capacitated continuous clustering problem. We propose two different models for different views: *p*-CCCP, where the number of clusters is given, and Generic CCCP, where we wish to find the best (p) number of clusters while minimising the dissimilarity between clusters.

Since the problems are NP-Complete, we prepare a two phases heuristic method for the Generic case, using a structured *q*-tree and constrained Means (Forgy-HMeans+ and JMeans) method to generate approximate solutions we call CAPCluster. We evaluate empirically the process of cascading improvement clustering (Means) methods. We also propose a more general procedure UCCAPCluster that can hold the *p/g*-CCCP, which is much more appropriate than the CAPCluster procedure for the instances tested.

Instances from the CCP literature, and very large instances from sales force districting of a food distributor and design of garbage collection zones, in Fortaleza, were used to evaluate the method. We compare the results using CCP costs obtained from Column Generation Antônio Lorena's program, since CCP values are close to CCCP, if in the instance, the number of medians (p) and the capacity Q is the

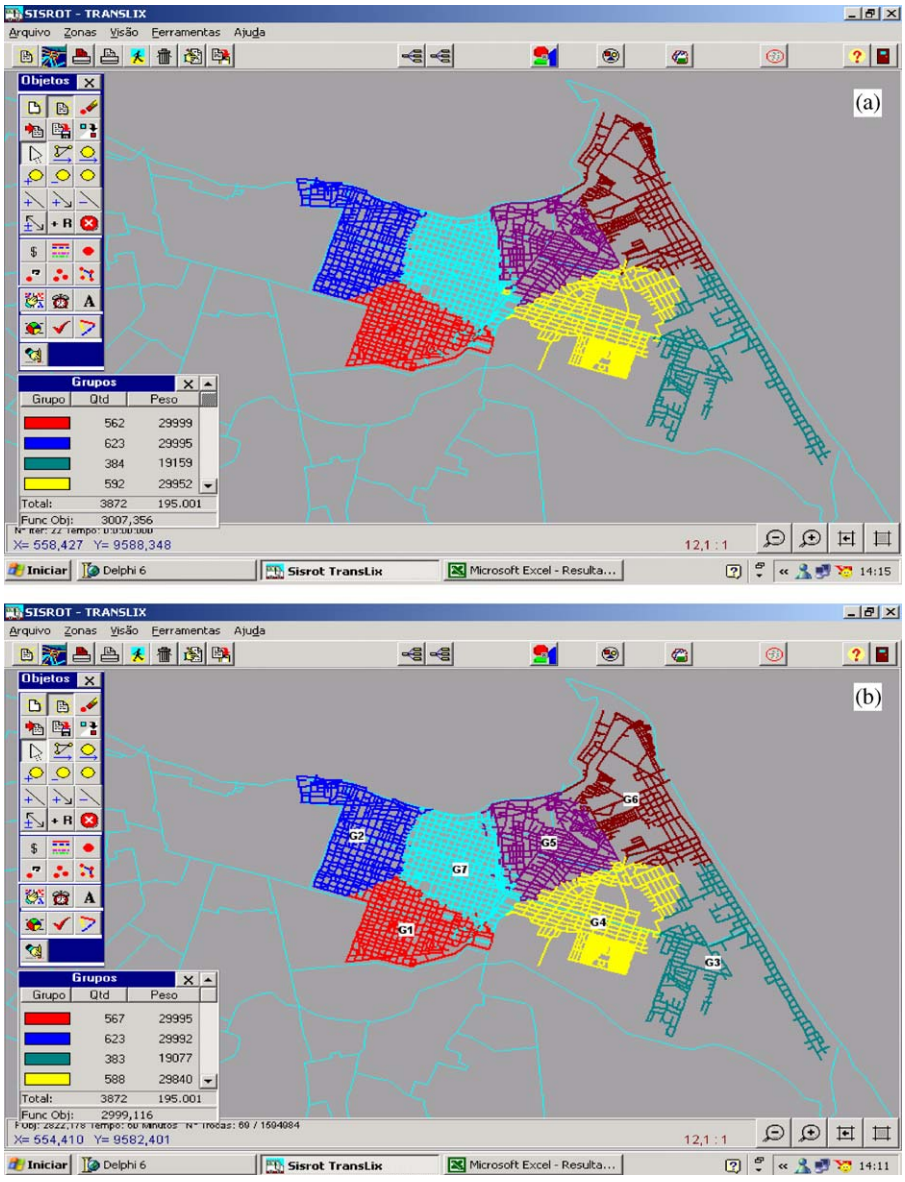


Fig. 10. (a) Garbage collection great area and initial solution from the UCCAPClust algorithm; (b) Final solution after 120 min of VNS and 0.27% of improvement.

same for both problems. We obtained in the majority of the instances tested that the objective function values of the CCCP are greater than the ones obtained for CCP.

All the tests can be found at http://www.lcc.uece.br/~negreiro/artigos/cccp. The tests were done in the SISROT® (Full and TRANSLIX) system, a software designed for solving related routing problems after clustering.

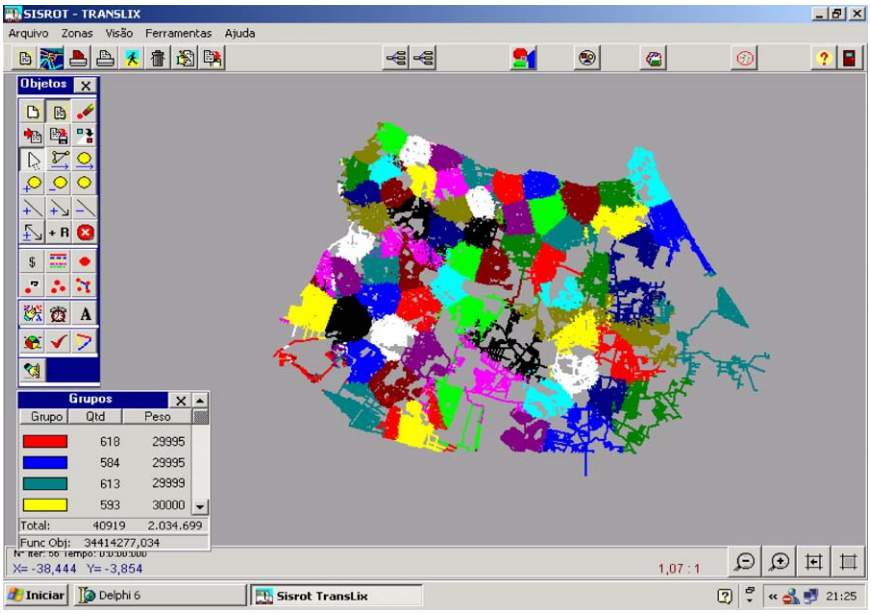


Fig. 11. Solution from the constructive step using UCCAPCluster procedure.

Table 5
Set of instances related to garbage collection in Fortaleza-CE/Brazil

GC01 NI: 3872, $Q = 30$ t	Next Fit/Best Fit $g = 7$			
Method	Obj func	Time (s)	+VNS (60+60)	
q-Tree+ Forgy	3095.10	0.09	3013.84	2.62%
UCCAPCluster	3007.35	0.24	2999.11	0.27%
GC02 NI: 40919, $Q = 30$ t	Next Fit/Best Fit $g = 68$			
Method	Obj func	Time (s)	+VNS	
q-Tree+ Forgy	160022155.44	187.50	-XXX-	-XX-
UCCAPCluster	34414277.03	9.84	-XXX-	-XX-

Acknowledgements

We would like to thank Professor Philippe Michelon from the Université de Avignon/France for his appointments in our models and comments, and Diane Béland for her English review of this text. Thanks to Professor Luiz Antônio Lorena, from the Instituto Nacional de Pesquisas Espaciais/Laboratório Associado de Computação e Matemática Aplicada (INPE/LAC) for his kind support in the CCP results. We also thank the anonymous referees for their suggestions to improve this work. This work is supported by

FUNCAP (Fundação Cearense de Apoio ao Desenvolvimento Científico e Tecnológico) by the grant for the project “Otimização Combinatória em Grafos”, and GRAPHVS.

References

[1] Hartingan JA. Clustering algorithms. New York: Wiley; 1975.
[2] Späth H. Clustering algorithms for data reduction and classification of objects. Chichesten: Ellis Hornwood; 1980.
[3] Koskosidis Y, Powell WB. Clustering algorithms for consolidation of customer orders into vehicle shipments. Transportation Research B 1992;26B(5):365–79.
[4] Hansen P, Jaumard B. Cluster analysis and mathematical programming. Mathematical Programming 1997;79:191–215.
[5] Mulvey JM, Beck MP. Solving capacitated clustering problems. European Journal of Operations Research 1984;18: 339–48.
[6] Osman I, Christofides N. Capacitated clustering problems by hybrid simulated annealing and tabu search. International Transactions in Operations Research 1994;1(3):317–36.
[7] França PM, Sosa NM, Pureza V. An adaptive tabu search algorithm for the capacitated clustering problem. International Transactions in Operations Research 1999;6:665–78.
[8] Lorena LA, Senne ELF. Local search heuristic for the capacitated *p*-median problem. Networks and Spatial Economics 2003;3:407–19.
[9] Lorena LA, Senne ELF. A column generation approach for the capacitated *p*-median problems. Computers & Operations Research 2004;31:863–76.
[10] Ahmadi S, Osman IH. Greedy random adaptive memory programming search for the capacitated clustering problem. European Journal of Operational Research 2005;162(1):30–44.
[11] Lorena LA, Furtado JC. Constructive genetic algorithm for clustering problems. Evolutionary Computation 2001;9(3): 309–27.
[12] Glover F, Laguna M. Tabu search. Dordrecht: Kluwer Academic Publishers; 1997.
[13] Downsland KA. Simulated annealing. In: Reeves CR, editor. Modern heuristic techniques for combinatorial problems. Oxford: Blackwell Scientific; 1993. p. 20–69.
[14] Collins NE, Eglease RW, Golden BL. Simulated annealing an annotated bibliography. American Journal of Mathematics and Management Sciences 1988;9:209–307.
[15] Negreiros Gomes MJ, Palhano AWC. Uma Aplicação para o problema generalizado de percurso de veículos. Anais do XXV SBPO–NATAL/RN, 2003.
[16] Kaufman L, Roussweuw P. Finding groups in data: an introductory to cluster analysis. New York: Wiley; 1990.
[17] Negreiros Gomes MJ, Almeida PG, Guarany A, Xavier AE. Análise de Agrupamentos para a Taxa de Resíduos Sólidos de Fortaleza. Limpeza Urbana 2002;57:10–7.
[18] Garey MR, Johnson DS. Computers and intractability: a guide to the theory of NP-completeness. San Francisco: Freeman; 1979.
[19] Forgy EW. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. Biometrics 1965;21(3):768.
[20] Farias B, Tito M, Quinderé M, Negreiros_Gomes M. Uma Avaliação de Algumas Técnicas de Pesquisa em Intervalos. Anais do VIIICLAIO-XXVIII SBPO;Rio de Janeiro, Agosto;1996.
[21] Hansen P, Mladenovic N. JMeans: a new local search heuristic for minimum sum-of-squares clustering. Pattern Recognition 2001;34(2):405–13.
[22] Brucker J. On the complexity of clustering problem, Lecture notes in economics and mathematical systems, vol. 157;1978. p. 45–54.
[23] Everitt BS, Landau SL, Leese M. Cluster analysis. fourth ed., Paris: Arnold; 2001.
[24] Diehr G. Evaluation of a branch-and-bound algorithm for clustering. SIAM Journal on Scientific and Statistical Computing 1985;6:266–84.
[25] Bouguettaya A, Le Vies Q. Data clustering analysis in a multidimensional space. Information Sciences 1998;112:267–95.
[26] Chiou Y-C, Lan LW. Genetic clustering algorithms. European Journal of Operations Research 2001;135:413–27.
[27] Jain AK, Dum RPW, Mao J. Statistical pattern recognition: a review. IEEE Transactions on Pattern Analysis and Machine Intelligence 2000;22(1):4–37.

[28] Josien J, Liao TW. Simultaneous grouping of parts and machines with integrated fuzzy clustering method. Fuzzy Sets and Systems 1999;126:1–12.
[29] Kiang MY. Extending the Kohonen self-organizing map networks for clustering analysis. Computational Statistics & Data Analysis 2001;38:161–80.
[30] Wu K-L, Yang M-S. Alternative c-means clustering algorithms. Pattern Recognition 2002;35:2267–78.
[31] Cormen TH, Leiserson CE, Rivest RL. Introduction to algorithms. Cambridge and New York: The MIT Press and McGraw-Hill Book Company; 2000.
[32] Martello S, Toth P. Knapsacks problems, algorithms and computer implementations. New York: Wiley; 1990.
[33] Hansen P, Mladenovic N. Variable Neighborhood Search. Chapter 3.6.2. Handbook of Applied Optimisation. In: Panos M. Pardalos, Mauricio GC, editors. New York, USA: Resendeby Oxford University Press, Inc.; 2002.
[34] CPLEX Optimization, Inc., Suite 279, 930 Tahoe Blvd. Bldg. 802. Incline Village. NV 89451-9436, e-mail: cplex.com