Pesquisa Operacional (2012) 32(3): 497-522

Printed version ISSN 0101-7438 / Online version ISSN 1678-5142
www.scielo.br/pope

HEURISTICS FOR MINIMIZING THE MAXIMUM WITHIN-CLUSTERS DISTANCE

José Augusto Fioruci[1], Franklina M.B. Toledo[1*] and Mariá Cristina V. Nascimento[2]

Received February 4, 2011 / Accepted April 28, 2012

ABSTRACT. The clustering problem consists in finding patterns in a data set in order to divide it into clusters with high within-cluster similarity. This paper presents the study of a problem, here called MMD problem, which aims at finding a clustering with a predefined number of clusters that minimizes the largest within-cluster distance (diameter) among all clusters. There are two main objectives in this paper: to propose heuristics for the MMD and to evaluate the suitability of the best proposed heuristic results according to the real classification of some data sets. Regarding the first objective, the results obtained in the experiments indicate a good performance of the best proposed heuristic that outperformed the *Complete Linkage* algorithm (the most used method from the literature for this problem). Nevertheless, regarding the suitability of the results according to the real classification of the data sets, the proposed heuristic achieved better quality results than *C-Means* algorithm, but worse than *Complete Linkage*.

Keywords: clustering, heuristics, GRASP, minimization of the maximum diameter.

1 INTRODUCTION

The data clustering problem aims at identifying in a given data set similar characteristics among its objects in order to divide them into clusters. An example of similarity is the proximity of objects in the data set, that is, the objects within a specific group should be closer to each other than to objects located in other clusters. Some data clustering applications are: data mining (Boginski *et al.*, 2006; Romanowski *et al.*, 2006), multiple protein sequence alignment (Kawaji *et al.*, 2004; Krause *et al.*, 2005), gene expression (Higham *et al.*, 2007; Huttenhower *et al.*, 2007) and image segmentation (Wu & Leahy, 1993), among many others.

*Corresponding author
[1]Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, SP, Brasil. E-mails: jaf@icmc.usp.br; fran@icmc.usp.br
[2]Instituto de Ciência e Tecnologia, Universidade Federal de São Paulo, 12231-280 São José dos Campos, SP, Brasil. E-mail: mcv.nascimento@unifesp.br

Data clustering is not an easy task because, in addition to the different sizes of most data sets, their clusters are often not clearly identifiable. Thus, several models to approach data clustering problem have been proposed in literature (Hansen & Mladenovic, 2001; Hansen & Jumard, 1997; Jain & Dubes, 1988; Rao, 1971). Each of these models has its bias, showing a better performance for specific types of data sets depending on the similarity measure adopted.

Some mathematical models for the data clustering problem are presented in (Rao, 1971). Particularly, the author proposed a mathematical model whose objective is to find a partition for a given number of clusters (defined *a priori*) that minimizes the maximum diameter (MMD) among all the clusters. In this formulation, only the distances between pairs of objects are considered, not necessarily their exact positions in the feature space. Rao (1971) also proposed a simple and efficient optimal algorithm to solve this problem when only two clusters are considered in its definition. Considering the combinatorial nature of the MMD problem, whose problem is to decide whether or not its solution is NP-hard, this paper proposes four heuristic methods for the clustering problem: two greedy heuristics (CH and GHLS) and two *Greedy Randomized Adaptive Search Procedures* (Feo & Resende, 1995; Resende & Ribeiro, 2010) (GRASP-I and GRASP-II).

For assessing the suitability of the solution methods proposed for the MMD problem, we use some instances from the literature (Nascimento, 2010) in the computational tests. These instances were produced for the analysis of the network community detection problem. The results from applying the methods to these data sets were evaluated according to two criteria: 1) the objective function value (the value of the largest diameter); and 2) the adequacy of the solution found using an external evaluation criterion for data clustering, the *Normalized Mutual Information* index, here referred to as NMI (Danon *et al.*, 2005). For the first experiment, we evaluated the solutions found by the proposed heuristics and compared with each other. In the second experiment, we used the software CPLEX 12.2 (IBM ILOG, 2010) in order to look for the optimal solutions of the data sets. In this experiment, we compared the solutions found by the best of the four proposed heuristics, the upper bounds found by CPLEX and the results of a benchmark heuristic for the MMD problem, the *Complete Linkage* (Hansen & Delattre, 1978). For the third experiment, to assess the adequacy of the solutions according to NMI, the partitions found by *Complete Linkage*, a well known algorithm in the literature, *C-Means* (Bezdek, 1981), the best proposed heuristic in this paper, GRASP-II were compared with the real data classification.

The results showed that GRASP-II achieved better solutions than others proposed heuristics for 85% of the data sets. Moreover, this metaheuristic got an exact solution for all twelve data sets, for which CPLEX provided exact solutions. However, the *Complete Linkage* was the heuristic that showed the best performance according to the NMI, while GRASP-II obtained better quality results than *C-Means* and CPLEX. Note that these conclusions are with regard to the application and specific characterization of the evaluated graphs.

The remainder of the paper is organized as follows. Section 2 describes the mathematical model proposed by Rao (1971) for the studied problem. For this paper to be self-explanatory, Section 3 presents the optimal method proposed by Rao (1971) for the 2-Clusters problem. The proposed

heuristics are detailed in Section 4. An algorithm for the *Complete Linkage* method is provided in Section 5. The computational results are reported in Section 6. To sum up, some concluding remarks are addressed in Section 7.

2 MATHEMATICAL MODEL

Rao (1971) proposed a mathematical formulation for the cluster analysis whose objective is to find a partition from a data set that minimizes the maximum within cluster distance. This problem is also known as the minimization of the largest diameter among the clusters. The mathematical formulation is described by (1)-(4) and the notations are given as follows:

Parameters

- N – number of objects of the data set;
- M – number of clusters of the final partition;
- d_{ij} – distance between objects i and j;

Variables

- x_{ik} – binary variable that relates object i to cluster k($x_{ik} = 1$, if object i is in cluster k; 0 otherwise);
- Z – value of the largest diameter among the M clusters (continuum variable);

$$\text{Min} \quad Z \tag{1}$$

$$\text{subject to} \quad d_{ij}x_{ik} + d_{ij}x_{jk} - Z \leq d_{ij}$$
$$i = 1, \ldots, N-1; \quad j = i+1, \ldots, N; \quad k = 1, \ldots, M; \tag{2}$$

$$\sum_{k=1}^{M} x_{ik} = 1 \quad i = 1, \ldots, N; \tag{3}$$

$$x_{ik} \in \{0, 1\}; Z \geq 0 \quad i = 1, \ldots, N; \quad k = 1, \ldots, M. \tag{4}$$

The objective function (1) minimizes the value of Z that represents the maximum within clusters distance. Constraints (2) ensure that the value of the largest cluster diameter is lesser than or equal to the Z variable value. Constraints (3) impose that each object belongs to just one cluster. Constraints (4) enforce the non-negativity of the variable Z and the binary nature of the other variables.

Brusco & Stahl (2005) point out that this problem has the tendency to produce clusters with just one object (isolated clusters). The authors proposed a specialized branch-and-bound (B&B) method to solve the problem. However, we evaluated this algorithm and for data sets with hundreds of objects the elapsed time to find the optimal solution was very high. We also considered the B&B with a good initial solution in order to fasten its convergence. However, the results remained poor. For example, for an instance with 336 objects and 8 groups, we could not obtain the optimal solution after two days trying to solve the problem. For this reason, to com-

pare the heuristic results with the optimal solution, here, we used the software CPLEX 12.2 (IBM ILOG, 2010).

In this paper, we use the usual definition of partition to represent a clustering. In this case, a partition is the set of groups $G_1, G_2, \ldots, G_M$, where the elements of G_i are objects that belong to a same cluster and if $i \neq j$, elements from G_i belong to a cluster different to the elements from G_j. Moreover, the union of all G_i's is the whole set of objects from the data set, and the intersection of these sets is empty.

3 2-CLUSTERS ALGORITHM FOR MMD PROBLEM

Rao (1971) proposed a polynomial optimal algorithm to solve the MMD problem for instances where the number of clusters in the final partition is equal to two. Here, this algorithm is called 2-Clusters algorithm. In this paper we propose four heuristics based on the 2-Clusters algorithm. The 2-Clusters algorithm is based on a simple idea: at each step it tries to assign the two most distant objects to different clusters.

In our description of the 2-Clusters algorithm, we maintain the same notation and nomenclature used in Section 2. In addition, D is the matrix of the pairwise distances between objects, labels A and B are used to define the two definitive clusters and $R(i)$ is the function that defines the label of object i. For example, if object i is labeled as A, then $R(i) = A$. In this case, object i is definitely assigned to cluster A. It is also possible to assign object i temporarily to one cluster. For this situation the author used a temporary label k, that is, $R(i) = k$, where k is an integer number between $-N$ and N. The 2-Clusters algorithm is described next.

In Figure 1, we show the step by step execution of the 2-Clusters algorithm using a hypothetical data set. In each step, the arrow signs the two objects with the largest paired distance. Circles with gray and black nucleus represent the definitive labels A and B, respectively. Temporary labels (k) are represented by squares and triangles.

In the first step (Fig. 1a), all objects are unlabeled. In Figure 1b, the pair of objects with the largest paired distance receives the definitive labels (black and gray). This distance is updated as zero in the distance matrix. In Figure 1c, the new pair of objects with the largest distance is selected and they are labeled to belong to different clusters. After that, the distance between this pair of objects is updated as zero in the distance matrix. In this case, this pair of objects receives different temporary labels, which are the black and the gray squares. This process is repeated in Figure 1d, now with triangle labels instead of squares. In Figure 1e, the largest distance corresponds to the distance between an unlabeled object and an object labeled with a gray color. Consequently, the unlabeled object receives the definitive label different from the gray one, that is, the black label. The largest distance in Figure 1f is determined by an object with a black triangle label (see Fig. 1d) and an object with the definitive black label. Therefore, the object with the temporary label receives the definitive gray label and all other objects with black triangle labels also receive the definitive label gray. In addition, all objects with gray triangle labels receive the final black label. In Figure 1g, the objects that determine the largest distance already have final labels and

Algorithm 2-Clusters

Step 1. Do $it \leftarrow 0$ and $R(i) \leftarrow 0$ for $i = 1, 2, \ldots, N$.

Step 2. Find p and q where $d_{pq}(d_{pq} > 0)$ is the largest element of D.

Step 2.1. If $it = 0$, then $R(p) \leftarrow A$, $R(q) \leftarrow B$ and $d_{pq} \leftarrow 0$. Go to Step 3.

Step 2.2. Otherwise, if $R(p) = 0$ and $R(q) = 0$, then $R(p) \leftarrow p$ and $R(q) \leftarrow -p$. Go to Step 3.

Step 2.3. Otherwise, if $R(p) = A$ (or B) and $R(q) = 0$, then $R(q) \leftarrow B$ (or A). Go to Step 3.

Step 2.4. Otherwise, if $R(q) = A$ (or B) and $R(p) = 0$, then $R(p) \leftarrow B$ (or A). Go to Step 3.

Step 2.5. Otherwise, if $R(p) = k$ (or $R(q) = k$) and $R(q) = 0$ (or $R(p) = 0$), then $R(q) \leftarrow -k$ (or $R(p) \leftarrow -k$). Go to Step 3.

Step 2.6. Otherwise, if $R(q) = A$ (or B) and $R(p) = k$, then all the objects with label k are assigned to cluster B (or A) and all the objects with label $-k$ are assigned to cluster A (or B). Go to Step 3.

Step 2.7. Otherwise, if $R(p) = A$ (or B) and $R(q) = k$, then all the objects with label k are assigned to cluster B (or A) and all the objects with label $-k$ are assigned to cluster A (or B). Go to Step 3.

Step 2.8. Otherwise, if $R(p) = k$ and $R(q) = k'$, with $k \neq k'$, then all the objects with label k' receive the label $-k$ and all the objects with label $-k'$ receive the label k. Go to Step 3.

Step 2.9. Otherwise, if the objects p and q are with opposed labels, i.e., $R(p) = A$ and $R(q) = B$ (or $R(p) = B$ and $R(q) = A$) or $R(p) = k$ and $R(q) = -k$ (or $R(p) = -k$ and $R(q) = k$), go to Step 3.

Step 2.10. Otherwise, if the objects p and q have same label ($R(p) = R(q)$) then go to Step 4.

Step 3. Do $d_{pq} \leftarrow 0$ and $it \leftarrow it + 1$. Go back to Step 2.

Step 4. For $k = 1, 2, \ldots, N$, assign all the objects with temporary label k to cluster A and all objects with label $-k$ to cluster B. If there is some unlabeled object so it can be assigned to cluster A or B arbitrarily. Return the final partition.

Algorithm 1 – 2-Clusters algorithm (Rao, 1971).

we just assign distance zero between them to matrix D. The same process as Figure 1g, but with different objects, occurs in Figure 1h. Thus, the object with the black squared label receives the black definitive label and the object with the gray squared label receives the gray definitive label. In the end, all the objects have definitive labels and the final partition with 2 clusters is illustrated in Figure 1i.

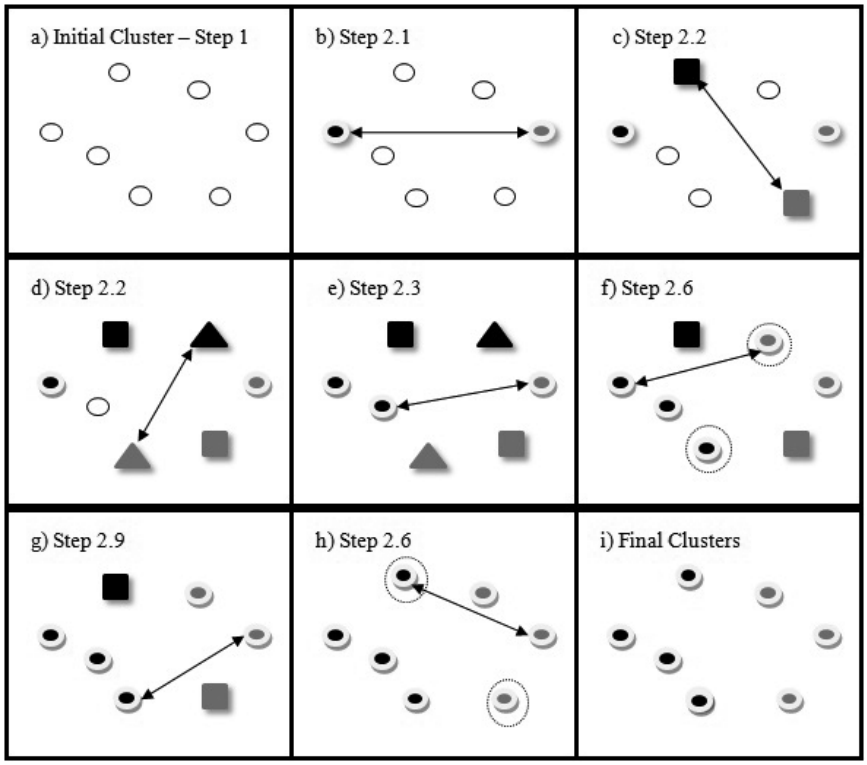


Figure 1 – 2-Clusters algorithm step-by-step.

According to Rao (1971), the optimality proof for 2-Clusters algorithm is obvious. Nevertheless, in this paper we present a formal proof of the optimality for this algorithm. For such, consider first the definition of partition. A partition $\{G_1, G_2, \ldots, G_M\}$ is defined as a set of M non-empty clusters. In these clusters all two by two cluster intersections is empty and the union of all clusters results in the data set.

Theorem: Let a data set be with n elements. Its partition $\pi = \{G_1, G_2\}$ produced by the 2-Clusters algorithm is optimum.

Proof: Let $L = \{d_1, d_2, \ldots, d_m\}$ be the m elements from matrix D sorted in decreasing order. Let π be a partition produced according to the 2-Clusters algorithm and Z_π be its largest diameter. Suppose that there is a partition π' with the largest diameter equals to $Z_{\pi'}$ and $Z_{\pi'} < Z_\pi$. Thus, objects i and j that are responsible for the diameter of π', *i.e.*, such that $d_{ij} = Z_{\pi'}$ in partition π', are assigned to different clusters. By construction, i and j were assigned to a same cluster in π, suppose, without loss of generality, to cluster G_1, by the following reason: there is a d_k from L such that $d_k = d_{ij} = d_{k'}$, with $k < k'$ and $dk = \max\{d_{ir}, d_{jr}$ where r belongs to $G_2\}$. If $k' < k$, by construction, i and j would be assigned to different clusters at iteration k'. Therefore, the value of Z from π' would be at least d_k that is greater than Z from partition π, that is a contradiction. □

Taking the complexity of the 2-Clusters algorithm into account, Step 2 is very important. In this step, we look for the pair of objects with the largest distance in D. Because D is a symmetric matrix, in the worst case, it is necessary to inspect half of the elements of D. As a consequence, this procedure has complexity $O(n^2)$ where n is the order of D. A sequential search in D leads to an $O(n^4)$ algorithm. A binary heap improves the running time to $O(n^2 \log n)$, and an even better theoretical amortized bound of $O(n^2)$ may be reached with a Fibonacci heap.

In the following section, we present the four heuristic methods proposed to solve the MMD problem for finding partitions with more than two clusters.

4 PROPOSED METHODS

The heuristics proposed in this paper find a partition in a data set into $M > 2$ clusters to approximately solve the MMD problem, whose mathematical formulation was presented in Section 2. In this paper, four heuristics are proposed: a greedy constructive, which applies the 2-Cluster algorithm several times in the largest diameter clusters until obtaining the solution with the desired number of clusters; a greedy heuristic with local search, which starts with an infeasible solution since the initial solution has a higher number of clusters than allowed, and at each iteration, groups pairs of clusters performing a local search; and two GRASP heuristics, which are based on the constructive heuristic and the greedy heuristic with local search, making the union of clusters in a randomized greedy form. Next, the heuristics proposed are detailed.

4.1 Constructive Heuristic (CH)

The constructive greedy heuristic proposed in this paper uses the 2-Clusters algorithm recursively. In the first step, the 2-Clusters algorithm is used to divide the data into two clusters. Next, the algorithm is applied again to the largest sized group and so on, until the M clusters are obtained.

4.2 Greedy Heuristic with Local Search (GHLS)

A local search heuristic means looking for a better solution in the neighborhood of a given solution. Then, the solution is replaced by the neighboring solution and the process is repeated until the current solution does not have better neighboring solutions. Considering that the used model seeks to minimize the largest diameter among the M clusters from a partition, neighboring solutions with lower maximum diameter value are sought.

The proposed local search was defined based on the neighborhood obtained using the following movement: remove the pair of elements that defines the largest diameter among all clusters and allocate them to other clusters so that the largest diameter is reduced. The local search is applied to a partition of the objects from the data set. The local search proposed is detailed in Algorithm 2.

Local Search

Step 1. Given a partition of objects from the data set $\{G_1, G_2, \ldots, G_M\}$, let G_k be the set of elements with the largest paired distance (diameter) and D_k be the value of this diameter.

Step 2. Find in G_k the pair of elements (a, b) whose $d(a, b) = D_k$, where $d(a, b)$ is the distance from a to b.

Step 3. Try to assign the objects a and b to different clusters, since the allocation of these objects to others clusters do not increase the largest diameter and since this allocation causes the smallest change on the diameters of the clusters that receive the objects a and b.

Step 4. If any object does not change the cluster in Step 3, then Stop and return the partition found. Otherwise, go to Step 1.

Algorithm 2 – Procedure of Local Search.

The starting point of GHLS is a partition with $\lfloor K^*M \rfloor$ clusters generated from the previously described CH, where $K \in \mathbb{R}$ is chosen experimentally as reported in Section 6. The constant $\lfloor K^*M \rfloor$ indicates the greatest integer value lesser than or equal to K^*M. The two clusters whose union results in the smallest increase in the objective function are then selected and grouped. Notice that this is a greedy strategy. As a result, we obtain $\lfloor K^*M \rfloor - 1$ clusters. This process is repeated until we obtain a feasible solution, that is, a partition with M groups. Thus, we have the Greedy Heuristic with Local Search algorithm described by Algorithm 3.

Greedy Heuristic with Local Search

Step 1. Build a partition with $\lfloor K * M \rfloor$ clusters using CH.

Step 2. Do $m \leftarrow \lfloor K * M \rfloor$.

Step 3. If $m = M$, then go to Step 6.

Step 4. Find the pair of clusters G_i and G_j in the current partition, so that the cluster resulting from their union has the smallest diameter.

Step 5. Join G_i with G_j and do $m \leftarrow m - 1$. Go to Step 3.

Step 6. Apply the Local Search to the current partition and return the solution.

Algorithm 3 – Greedy Heuristic with Local Search algorithm.

4.3 GRASP

GRASP (Feo & Resende, 1995) is a metaheuristic consisting of two phases: a constructive and local search ones. The constructive phase consists in obtaining, iteratively, a pseudo-greedy solution. At each iteration of this phase, a set S of all possible choices of elements to be added to the current partial solution is evaluated. One from the best t, with $t \leq N$, options is drawn

and added to the partial solution. A shortlist of candidates is designated to this restricted list of candidates (RLC). At the end of this phase, a feasible solution to the problem is found, for which is applied a local search procedure.

The GRASP metaheuristic is known for finding good quality solutions in various optimization problems (Nascimento *et al.*, 2010; Marinakis *et al.*, 2008). For this reason, two GRASP metaheuristics for the MMD problem are proposed in this paper. The proposed constructive phase of GRASP is the GHLS with Step 4 modified. In this modification, instead of grouping the *clusters* of a partition π that produces the lowest maximum diameter, two clusters that provide one of the smallest t maximum diameters are grouped. The pseudo code of the constructive phase of the proposed metaheuristics is detailed in Algorithm 4. Note that in Step 4, the size of vector V is equal to the combination of m pairs of objects.

Constructive Phase (π)

Step 1. If $m = M$, return the feasible solution found.

Step 2. Otherwise, consider an ascending ordered vector V of size t, with all diameter values from every pairwise clusters union from partition π.

Step 3. Get a random real value α in the interval [0, 1].

Step 4. Get a random position j of the vector V, such $V_j < V_0 + \alpha^*(V_{t-1} - V_0)$, whose V_0 and V_{t-1} are the first and the last position of V, respectively.

Step 5. Update π joining the clusters corresponding to the diameter V_j and do $m \leftarrow m - 1$. Go to Step 1.

Algorithm 4 – Constructive Phase of the proposed GRASP.

In Step 3 of Algorithm 4, α changes in all iterations, thus this algorithm can be considered a reactive GRASP (Resende & Ribeiro, 2010). In Algorithm 4 it is necessary to build a sorted vector with the values of the diameters resulted from the pairwise junction of all clusters from π. To construct the vector V it is necessary to scan the distance matrix D and to sort it. Thus the order of Algorithm 4 is $O(n^2 + m^* \log(m))$.

The two GRASP (GRASP-I and GRASP-II) proposed in this paper differ because GRASP-II applies the local search to the partial solutions from the constructive phase and GRASP-I does not. At the end of the constructive phase of the proposed metaheuristics, at each iteration, a feasible solution is obtained, which is improved by the local search procedure. Several iterations of these steps are carried out and the highest quality solution is kept. The proposed metaheuristics are detailed in the pseudo codes presented, respectively, in Algorithms 5 and 6.

In Algorithms 5 and 6, the function Z_π gives the value of largest diameter among the clusters from partition π; the constants *Max_it* and *INFINITY* are, respectively, the maximum number of iterations of the proposed GRASP and a large initial value for the empty solution.

GRASP-I

Step 1. Do $it \leftarrow 0$; $Z_{\pi\,\min} \leftarrow INFINITY$.

Step 2. Build a partition with $\lfloor K^*M \rfloor$ clusters using CH.

Step 3. Do $m \leftarrow \lfloor K^*M \rfloor$.

Step 4. While $m > M$ do

  Step 4.1. $\pi \leftarrow Constructive\,Phase(\pi)$.

Step 5. Do $\pi \leftarrow Local\,Search(\pi)$.

Step 6. Do $it \leftarrow it + 1$.

Step 7. If $Z_\pi < Z_{\pi\,\min}$, do $\pi_{\min} \leftarrow \pi$.

Step 8. If $it = Max\_it$, return $\pi_{\min}$. Otherwise, go to Step 2.

Algorithm 5 – Pseudo code GRASP-I.

GRASP-II

Step 1. Do $it \leftarrow 0$; $Z_{\pi\,\min} \leftarrow INFINITY$.

Step 2. Build a partition with $\lfloor K^*M \rfloor$ clusters using CH.

Step 3. Do $m \leftarrow \lfloor K^*M \rfloor$.

Step 4. While $m > M$ do

  Step 4.1. $\pi \leftarrow Constructive\,Phase(\pi)$.

  Step 4.2. $\pi \leftarrow Local\,Search(\pi)$.

Step 5. Do $it \leftarrow it + 1$.

Step 6. If $Z_p < Z_{\pi\,\min}$, do $\pi_{\min} \leftarrow \pi$.

Step 7. If $it \geq Max\_it$, return $\pi_{\min}$. Otherwise, go to Step 2.

Algorithm 6 – Pseudo code GRASP-II.

The algorithms GRASP-I and GRASP-II use the constructive heuristic to generate an initial partition with $\lfloor K^*M \rfloor$ clusters. The order of the best case of this phase is $O(\lfloor K^*M \rfloor^* n^2)$ and in the worst case, its order is $O(\lfloor K^*M \rfloor^* n^4)$ (see Step 2). In each of the Max_it iterations of both metaheuristics, the constructive phase is repeated $\lfloor K^*M \rfloor - M$. The Local Search phase performed once in each iteration of GRASP-I, whereas in GRASP-II, the Local Search repeated $\lfloor K^*M \rfloor - M$ (the same number of times as the constructive phase). Considering m the number of clusters, in the best case, the Local Search performs only one iteration with order $O(m^*n^2)$. In the worst case, although unlikely, all pairs of objects can be changed having the order $O(m^*n^4)$. In the best case, GRASP-I has an order of $O(Max\_it^*((\lfloor K^*M \rfloor - M)^*(n^2 + m^* \log(m)) + m^*n^2))$ and, in the worst case, has an order of $O(Max\_it * ((\lfloor K^*M \rfloor - M)^*(n^2 + m^* \log(m)) + m^*n^4))$. In the best case, GRASP-II has an order of $O(Max\_it^*(\lfloor K^*M \rfloor - M)^*(m^* \log(m) + m^*n^2))$ and, in the worst case, it has an order of $O(Max\_it^*(\lfloor K^*M \rfloor - M)^*(m^* \log(m) + m^*n^4))$. Therefore,

if we consider only the number of objects (n), the proposed GRASP heuristics have the same order of complexity as the 2-Clusters Algorithm, presented in Section 3.

5 COMPLETE LINKAGE

Complete Linkage is a hierarchical agglomerative method for data clustering (Mingoti, 2007). An agglomerative hierarchical method considers, initially, each element of the data set with n objects as a group with a single object (initial clustering with n isolated groups). In the second iteration of the algorithm, two clusters are chosen, according to a measure of similarity (or dissimilarity), to be joined thus forming an $n-1$ cluster. The process is repeated until a group of only one cluster is obtained that contains all the objects, therefore, in n iterations.

The hierarchical property consists in the fact that at a certain iteration of the algorithm a pair of the elements appears in the same group. These elements will be kept together in the subsequent iterations. This property allows the construction of a tree of unions (dendrogram) that occurred in the iterations of the algorithm. Thus a partition with M clusters ($M > 1$) can be obtained by cutting the tree at the $(n - M + 1)$-th iteration.

In the *Complete Linkage* the similarity between two clusters G_1 and G_2 is defined as the largest distance between an object of G_1 and an object of G_2. In other words, the similarity between two clusters is defined as the diameter of the cluster resulting from the union of these clusters. Therefore, Hansen & Delattre (1978) suggest that the *Complete Linkage* can be considered as a heuristic for the MMD problem. Its pseudo code with the tree cut for M clusters is shown in Algorithm 7.

Complete Linkage

Step 1. Given a data set with n objects, consider each object as a cluster, *i.e.*, build a clustering with n clusters isolated.

Step 2. Perform $m \leftarrow n$.

Step 3. Find the pair of clusters G_i and G_j in the current partition, so that the cluster resulting from their union has the smallest diameter.

Step 4. Join G_i with G_j and do $m \leftarrow m - 1$.

Step 5. If $m = M$ Stop. Otherwise, go back to Step 3.

Algorithm 7 – *Complete Linkage* algorithm.

6 COMPUTATIONAL TESTS

The proposed heuristics were programmed in C and the computational experiments were performed on a microcomputer Core 2 Duo, 2.0 GHz with 3 GB of random access memory (RAM) under Windows 7 operational system. The tests using the software CPLEX version 12.2 were carried out on a cluster IBM Quad-core Intel (R) Xeon(R) CPU E5504 2.00 Ghz Linux with a processor with 2Gb of RAM. The values used for the parameter K of the pseudo code GRASP-I

and GRASP-II were: 1.5; 2 and 3. Other values of K were used in preliminary tests and, overall, the results obtained with $K < 1.5$ showed lower quality than those obtained when $K \geq 1.5$. When $K > 3$, the results did not show significant improvement to justify its use due to the considerable computational time increase. The best results were obtained for K equal to 2. Therefore, this value was adopted for the three proposed heuristics: GRASP-I, GRASP-II and GHLS. The maximum number of iterations of GRASP-I and GRASP-II, Max_it, was defined as 100, a value adjusted by means of computational experiments. In these experiments, we considered the trade-off between solution quality and necessary computational time to achieve the final solution.

For the computational tests, we used 60 graphs with the number of nodes ranging from 100 to 1000 containing structures from 3 to 50 *clusters*. Here, we will refer the nodes of the graphs as objects when using data clustering algorithms. These artificial graphs were generated by Nascimento (2010) using the following systematic: let $A = \{100, 200, 300, \ldots, 1000\}$ be the set of number of nodes and $B = \{3, 4, 5, 10, 20, 50\}$ be the set of number of clusters. For each $x \in A$, there is a single graph with $y \in B$ clusters. An example of a graph with 100 nodes and structure with 3 clusters is presented in Figure 2.

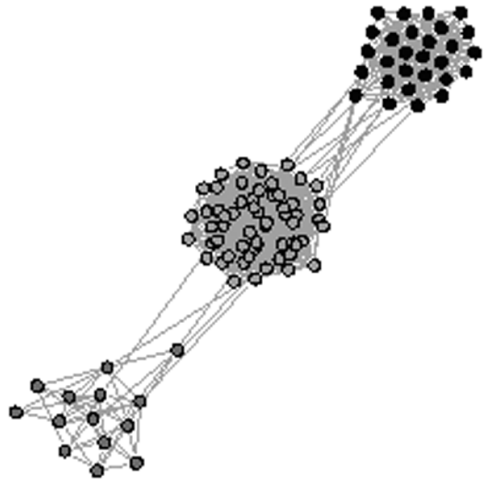


Figure 2 – Graph with 3 clusters and 100 nodes with labels of the clusters generated in its construction.

As the original measurement of weights between two nodes is relative to the similarity between them, a conversion was necessary to measure the distance between them. For such, the following conversion formula was used:

$$d_{ij} = 1 - \frac{w_{ij}}{1 + w_{\max}},$$

if $w_{ij} \neq 0$, and $d_{ij} = INFINITY$, if $w_{ij} = 0$. To obtain the distance matrix between all nodes, after this step, the shortest path between each pair of nodes was considered, which can be calculated using the Dijkstra's algorithm. With these distance matrices (referred here to as data sets), the experiments were developed. The results of the heuristics were evaluated according to two

criteria: 1) the quality of the solutions obtained according to the objective function value; and 2) the adequacy of the solution according to NMI (Danon *et al.*, 2005), which is an external evaluation criteria for data clustering. In the first case, to assess the proposed methods' solution quality, their results were compared with each other and with *Complete Linkage* (Hansen & Delattre, 1978), described in Section 5. The implementation used for this algorithm is available in the cluster package of R-project software (Ihaka, 1993). In addition, the results were compared with the optimal solution of the problems. Thus, the problems were solved using the optimization CPLEX 12.2 software (IBM ILOG, 2010). Also, to evaluate the suitability of our algorithm we considered the classical algorithm from literature, *C-Means* (Bezdek, 1981), whose used implementation is from the *e1071* package of R-project. Roughly speaking, the Fuzzy *C-Means*, or simply *C-Means* algorithm, was proposed in Bezdek (1981). This algorithm follows the Fuzzy logic, where each object belongs to one cluster with certain degree of membership and one object can belong to more than one cluster. The basic idea of the algorithm is to start with the random central points of the clusters and go to updating these points according to an objective function. The *C-Means* algorithm searches for a partition that minimizes the objective function that is average of the squares of the distances of each object to the centers of all clusters weighted with the degree of pertinence of each object with respect to the group.

The computational tests were performed in three experiments. In the first experiment, the four proposed methods (CH, GHLS, GRASP-I and GRASP-II) were applied to the described data sets. Furthermore, their solutions were compared with each other to verify their performance and to analyze the best proposal. In the second experiment, the solutions obtained by the best method were compared with those obtained by the *Complete Linkage* algorithm. In addition, both methods were evaluated with the solutions obtained by the CPLEX optimization software with the computational time limited to 3 hours. Finally, at the third step, the results obtained by the best proposed method, CPLEX, *Complete Linkage* and *C-Means* were evaluated regarding their classification suitability according to the NMI.

In the first two experiments, the gap was used as the main comparative variable. The used gap formula is:

$$gap = \frac{(heur\_sol - best\_heur)}{best\_heur}$$

where *heur_sol* is the heuristic solution and *best_heur* is the best heuristic solution.

6.1 Comparison between the proposed methods

Figure 3 illustrates the results obtained with the proposed heuristics, considering its objective function, that is, the variable Z value of the problem. In these graphs, one can observe the superiority of the solutions found by GRASP-II over the other heuristics. For 59 of the 60 data sets, GRASP-II had the results better than or equal to the solutions from the other heuristics. GRASP-I and GHLS had the results better than or equal to the solutions from the other heuristics in 25 and 15 instances, respectively. For only one of the instances CH had the best result and, in 7 data sets, its results were equal to the best solution found.

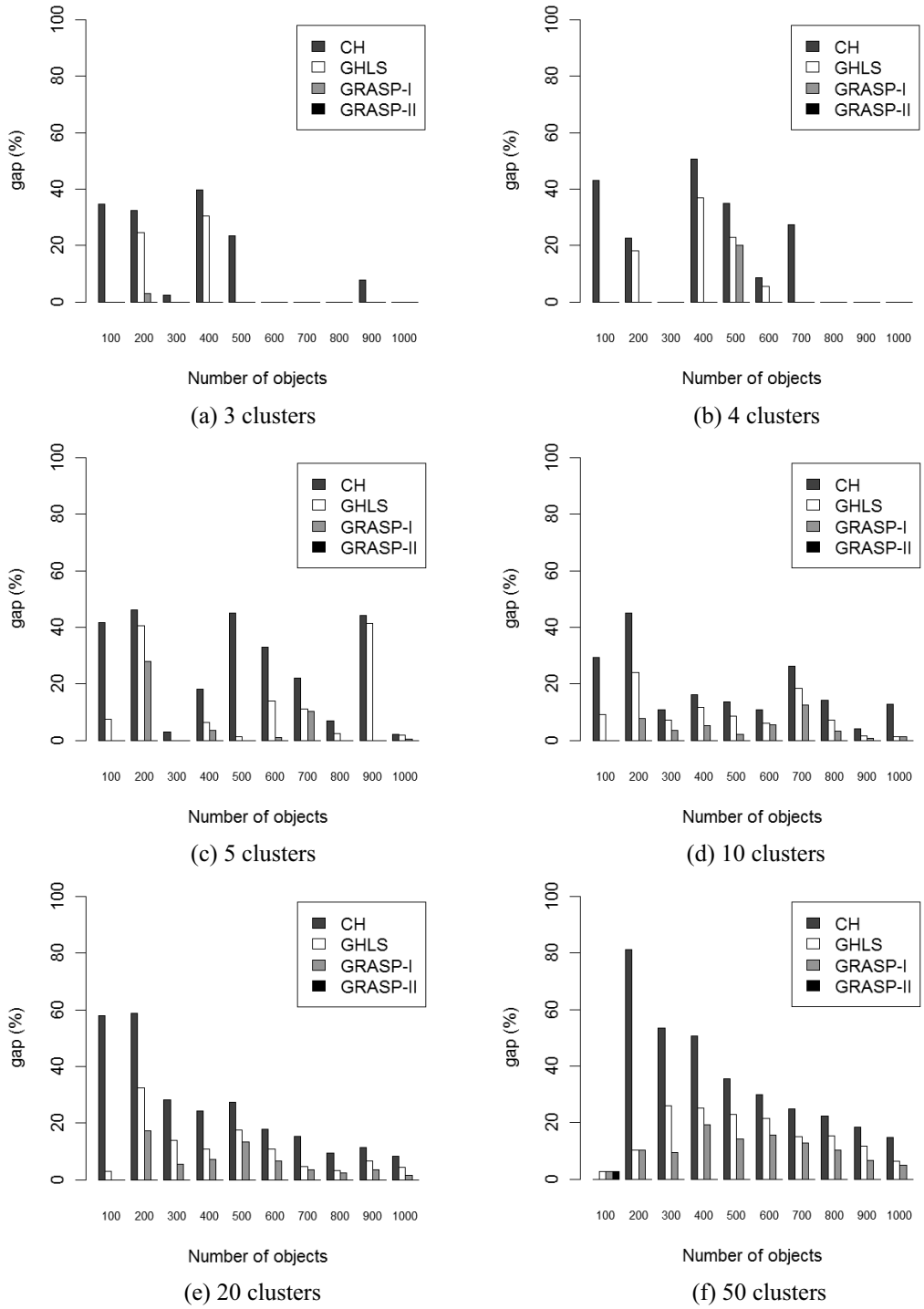


Figure 3 – This figure shows 6 graphics regarding the results of the proposed heuristics.

Regarding the computational time, as CH and GHLS are simplifications of GRASP-I, both had the computational time lower than the computational time of GRASP-I. In the worst case, they took about six minutes to run. GRASP-II is computationally more expensive than GRASP-I, because GRASP-II applies local search every time two clusters are grouped in the construction phase. In addition, in GRASP-II, the number of cluster unions in each iteration is proportional to the number of clusters of the final partition (M). This means that the computational cost increases with the number of clusters. Therefore, it is expected that in problems with many clusters, the computational cost of GRASP-II is high. The highest computational cost of GRASP-II occurred for the instance with 1000 objects and 50 clusters, when the method required about 25 minutes to complete the run. For additional analysis, Table 1 from the appendix of this paper reports the objective function values of the solutions found by the heuristics and their computational time.

As GRASP-II generated the best results for most of the assessed problems, the next experiments were performed only for this metaheuristic.

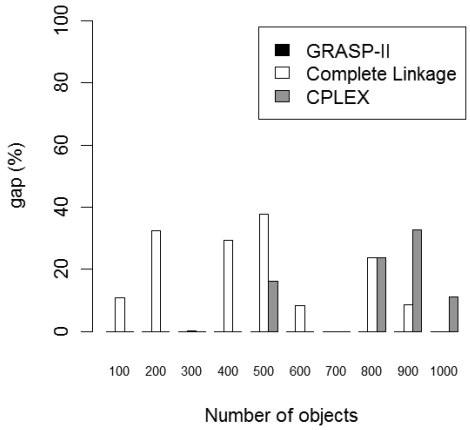
6.2 Comparison of GRASP-II, *Complete Linkage* and CPLEX

In order to assess the quality of solutions obtained by GRASP-II, we compare them with the solutions obtained by the heuristic from literature, the *Complete Linkage* (Hansen & Delattre, 1978). The results of this comparison can be observed in Figure 4. In Figure 4, the displayed graphs show the gaps obtained by the *Complete Linkage* and GRASP-II heuristics with relation to the best feasible solution found among the three methods: *Complete Linkage*, GRASP-II and CPLEX. GRASP-II had a mean gap equals to 0.3%, a standard deviation of 1.1% and a maximum gap equals to 5.6%. *Complete Linkage* obtained a mean gap of 8.4%, a standard deviation of 11.3% and a maximum gap of 44.0%. In addition, according to the 60 assessed problems, 7 of them obtained the same results using *Complete Linkage* and GRASP-II. For only 9 instances, the *Complete Linkage* results were better than the results from GRASP-II.

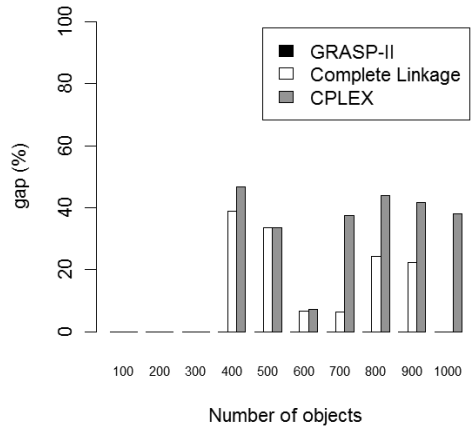
Analyzing the results obtained by CPLEX, only 12 optimal solutions were found by it. For these instances, GRASP-II also found the optimal solutions, whereas *Complete Linkage* obtained the optimal solution for four of these instances. Regarding the other instances, for only 3 of them GRASP-II obtained worse solutions than CPLEX, whereas *Complete Linkage* obtained 16 inferior results. For more details about these solutions, the authors recommend analyzing the values reported in Table 2 of the appendix of this paper. This table presents the results obtained by the heuristics, the bounds obtained by CPLEX, and the execution time of each tested method.

According to these results, we can conclude that GRASP-II had an excellent performance, despite the higher computational time in some test cases. Thus, one can consider this metaheuristic as a better alternative than *Complete Linkage*, a classical heuristic from literature.

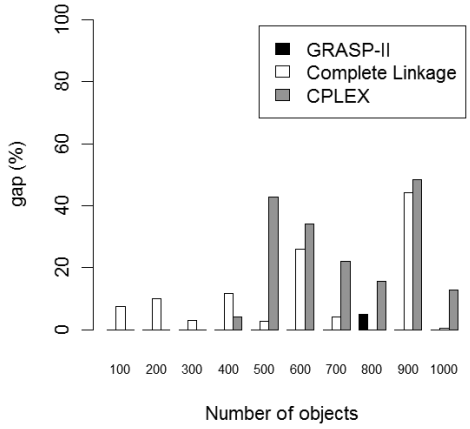
Next, other criteria are evaluated with respect to the results of *Complete Linkage*, CPLEX and of the best heuristic proposed in this paper, GRASP-II. For this assessment we used an external evaluation criterion, consistent with the classification of data sets.



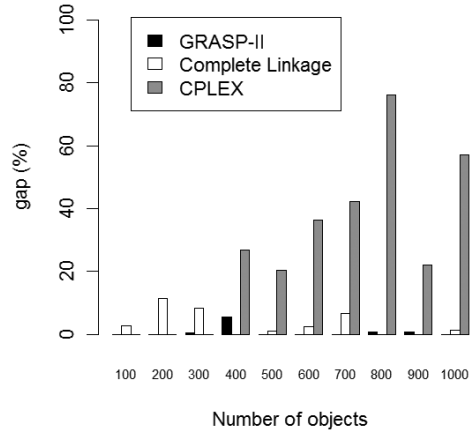
(a) 3 clusters



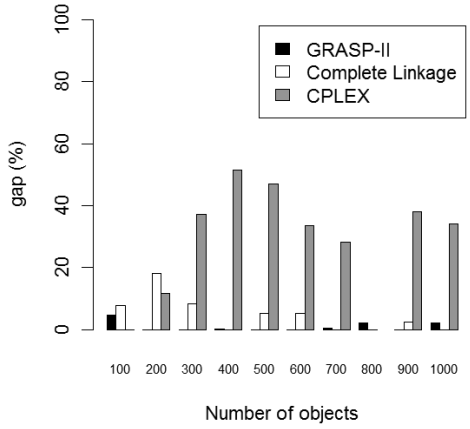
(b) 4 clusters



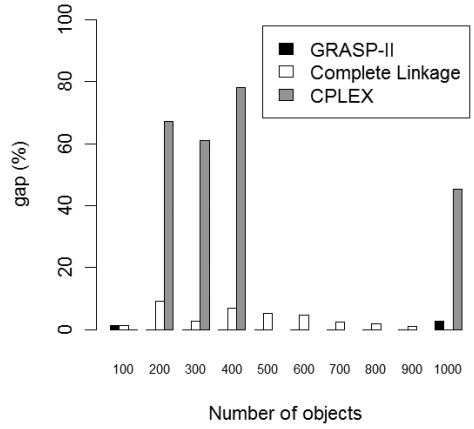
(c) 5 clusters



(d) 10 clusters



(e) 20 clusters



(f) 50 clusters

Figure 4 – Graphs with the results achieved by GRASP-II, *Complete Linkage* and CPLEX.

6.3 Assessment of the solutions according to NMI

To evaluate the results of GRASP-II with respect to the real classification of the used data sets, an external evaluation criterion for data clustering was used, the *Normalized Mutual Information* (NMI) (Danon *et al.*, 2005). The use of this measure is inspired on the study performed by Lancichinetti & Fortunato (2009). The authors compare partitions from data sets found by clustering algorithms taking the known labels of the data sets into account. For such, first consider a partition as a vector containing the labels (the cluster number) at the corresponding position of the object. For example, if the i-th object belongs to the k-th cluster, then the i-th position of its label vector is k. Having this definition, let $\pi^{(1)}$ and $\pi^{(2)}$ be partitions whose label vectors are represented, respectively, as $p^{(1)}$ and $p^{(2)}$. The NMI, the normalized form of the Mutual Information measure, that estimates the dependence of two random variables used in this paper is presented in Equation (5).

$$NMI = \frac{I(X, Y)}{\sqrt{H(X)H(Y)}} \tag{5}$$

where X and Y are the random variables that describe $p^{(1)}$ and $p^{(2)}$, respectively,

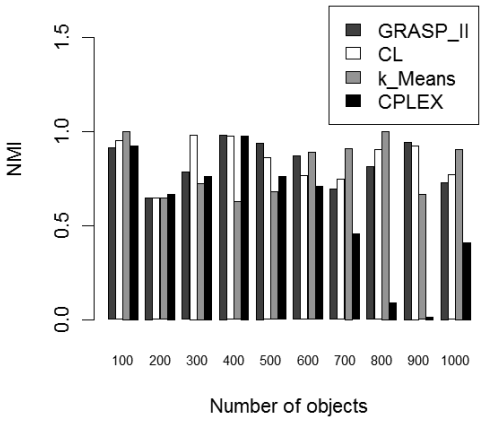
$$H(X) = \sum_{h=1}^{k^1} |C_h^{p^{(1)}}| \log \frac{|C_h^{p^{(1)}}|}{n}$$

and

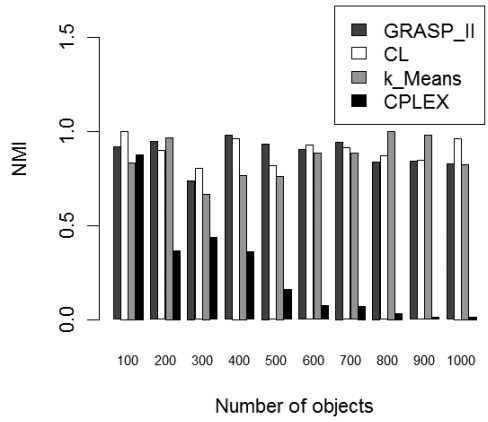
$$I(X, Y) = \sum_{h=1}^{k^1} \sum_{i=1}^{k^2} |C_h^{p^{(1)}} \cap C_i^{p^{(2)}}| \log \frac{n|C_h^{p^{(1)}} \cap C_i^{p^{(2)}}|}{|C_h^{p^{(1)}}|\,|C_i^{p^{(2)}}|}\,.$$

The closer to 1, the better is the clustering according to the original labels. There are other versions for the normalization of NMI, however, the one presented is better for comparing partitions which are not guaranteed for having balanced clusters. The case of the data sets from our experiments. The graphs in Figure 5 show the results achieved by the GRASP-II, *Complete Linkage*, *C-Means* and CPLEX methods, according to NMI.

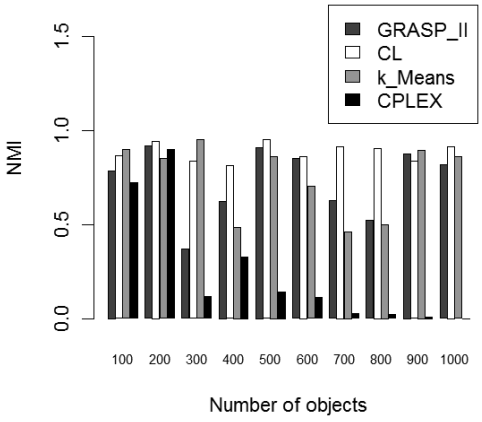
For only 14 instances the *Complete Linkage* obtained lower results than the other heuristics. GRASP-II, *C-Means* and CPLEX obtained best solutions for 9, 4 and 1, respectively. Even for the 12 instances for which CPLEX and GRASP-II obtained optimal solutions, the *Complete Linkage* still had better NMI mean values. The mean NMI of GRASP-II and of *C-Means* were 0.7 and 0.6, respectively, with standard deviations of 0.2, for both, whereas for the *Complete Linkage*, the mean NMI was 0.9, with standard deviation of 0.1. CPLEX showed the worst results in this experiment, with a mean NMI of 0.3, and a standard deviation of 0.3. It should be noticed that *Complete Linkage* achieved better results than GRASP-II in 83% of the problems, while *C-Means* and CPLEX obtained for 92% and 95%, respectively. Even though GRASP-II has worse results than *Complete Linkage*, its results were higher than 0.7 for 58% of the instances. These data can be found in Table 3 of the appendix of this paper.



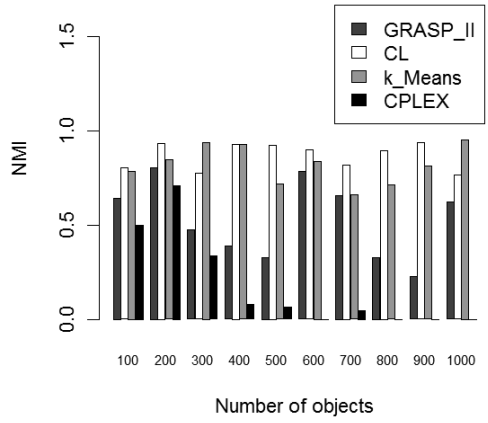
(a) 3 clusters



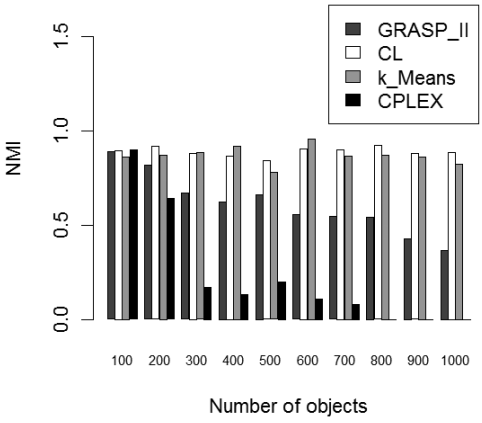
(b) 4 clusters



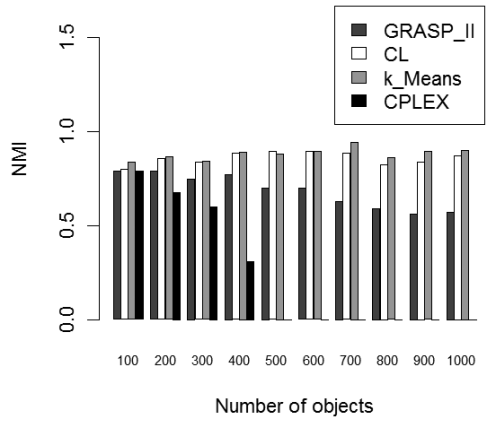
(c) 5 clusters



(d) 10 clusters



(e) 20 clusters



(f) 50 clusters

Figure 5 – Graphs with the NMI of the partitions found by the following methods: GRASP-II, *Complete Linkage* (CL), *C-Means* and CPLEX.

To sum up, it must be highlighted that GRASP-II was the solution method that obtained the best results among all the tested methods. In other words, this metaheuristic found better solutions for the studied integer problem. Furthermore, it is possible that if we find partitions with different number of clusters for a same data set, the NMI results of the proposed heuristic may improve. Moreover, possibly, more robust solutions could be achieved by the heuristics regarding the external evaluation criterion, NMI.

7 FINAL REMARKS

In this paper we studied the data clustering problem based on formulation which aims at the minimization of the largest diameter of a partition. To solve it approximately, four heuristics were proposed and assessed: two greedy heuristics (CH and GHLS) and two GRASP metaheuristics (GRASP-I and GRASP-II). In the first experiment, the performance of the methods was comparatively evaluated. The results showed that the solutions obtained by GRASP-II were superior to the other proposed methods. However, as expected, this method was computationally the most expensive. In the second experiment, the main purpose was to compare the objective function value obtained by GRASP-II with respect to the solutions of the *Complete Linkage* method and to the optimal solutions found by the optimization software CPLEX. GRASP-II proved to be very efficient in 85% of the cases obtaining better results than *Complete Linkage*. Moreover, it found the optimal solutions for the twelve problems for which CPLEX provided the optimal solutions. In the third and last experiment, we performed a suitability test using an external validation criterion for data clustering, the Normalized Mutual Information (NMI). In this experiment, the partitions found by GRASP-II, CPLEX, *Complete Linkage* and *C-Means* were validated with the original partition of the data set. In this case, it was observed that, for the studied data sets, *Complete Linkage* obtained solutions closer to the real classification than GRASP-II and *C-Means*.

The results of this study indicate that, for a considerable large number of instances, the MMD problem can be solved efficiently by the GRASP-II. Moreover, by the first two experiments, GRASP-II achieved the best results in this paper. The improvement we aimed is with regard to the internal validation criterion, that is, the objective function of the MMD problem. Thus, the proposed metaheuristic is characterized for being highly efficient. Nevertheless, even though the results obtained by the proposed metaheuristic according to the external validity criterion had good quality, they were lower than those of the *Complete Linkage*. These values were closer to the results obtained by *C-Means* and CPLEX in the problems for which it found the feasible solution. These results may indicate that the MMD problem perhaps is not the most appropriate formulation for the data clustering problem.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous referees for their helpful comments which significantly improved the quality of this paper. This research was funded by the Brazilian FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo) and CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico).

REFERENCES

[1] BEZDEK JC. 1981. Pattern recognition with fuzzy objective function algorithms. New York: Plenum.

[2] BOGINSKI V, BUTENKO B & PARDALOS PM. 2006. Mining market data: A network approach. *Computers & Operations Research*, **33**: 3171–3184.

[3] BRUSCO MJ & STAHL S. 2005. Branch and Bound applications in combinatorial data analysis. New York: Springer-Verlag.

[4] CANO J, CORDÓN O, HERRERA F & SÁNCHEZ L. 2002. A GRASP algorithm for clustering. *Lecture Notes in Computer Science Springer*, 214–223.

[5] DANON L, DUCH J, ARENAS A & DÍAZ-GUILERA A. 2005. Comparing community structure identification. Statistical mechanics, 9008, 09008.

[6] DUDA RO. 2001. Pattern classification. John Wiley & Sons.

[7] FEO TA & RESENDE MGC. 1995. Greedy randomized adaptive search procedures. *Global Optimization*, **6**: 109–133.

[8] HANSEN P & DELATTRE M. 1978. Complete-link cluster analysis by graph coloring. *American Statistical Association*, **73**: 362, 397–403.

[9] HANSEN P & JAUMARD B. 1997. Cluster analysis and mathematical programming. *Mathematical Programming*, **79**: 191–215.

[10] HANSEN P & MLADENOVIC N. 2001. J-Means: a new local search heuristic for minimum sum of squares clustering. *Pattern Recognition*, **34**: 405–413.

[11] HIGHAM DJ, KALNA G & VASS JK. 2007. Spectral analysis of two-signed microarray expression data. *Mathematical Medicine and Biology*, **24**: 131–148.

[12] HUTTENHOWER C, FLAMHOLZ AI, LANDIS JN, SAHI S, MYERS CL, OLSZEWSKI KL, HIBBS MA, SIEMERS NO, TROYANSKAYA OG & COLLER HA. 2007. Nearest neighbor networks: clustering expression data based on gene neighborhoods. *BMC Bioinformatics*, **8**: 250.

[13] IBM ILOG. 2010. CPLEX 12.2.0 reference manual.

[14] IHAKA R & GENTLEMAN R. 1993. R Development Core Team. GNU General Public License.

[15] JAIN AK & DUBES RC. 1988. Algorithms for Clustering Data. Prentice-Hall advanced reference series. Prentice-Hall, Inc., Upper Saddle River, NJ.

[16] JAIN AK, MURTY MN & FLYNN PJ. 1999. Data clustering: a review. *ACM Computing Sureys*, ACM Press, **31**: 264–323.

[17] KAWAJI H, TAKENAKA Y & MATSUDA H. 2004. Graph-based clustering for finding distant relationships in a large set of protein sequences. *Bioinformatics*, **20**(2): 243–252.

[18] KRAUSE A, STOYE J & VINGRON M. 2005. Large scale hierarchical clustering of protein sequences. *BMC Bioinformatics*, **6**: 15.

[19] LANCICHINETTI A & FORTUNATO S. 2009. Community detection algorithms: a comparative analysis. *Physical Review A*, **80**: 056117-[11 pages].

[20] MARINAKIS Y, MARINAKI M, DOUMPOS M, MATSATSINIS N & ZOPOUNIDIS C. 2008. A hybrid stochastic genetic-GRASP algorithm for clustering analysis. *Operational Research*, **8**: 22–46.

[21] MINGOTI SA. 2007. Análise de dados através de métodos de estatística multivariada, uma abordagem aplicada. Belo Horizonte: UFMG.

[22] NASCIMENTO MCV. 2010. Metaheurísticas para o problema de agrupamento de dados em grafo. Tese de Doutorado, Universidade de São Paulo.

[23] NASCIMENTO MCV, TOLEDO FMB & CARVALHO ACPLF. 2010. Investigation of a new GRASP-based clustering algorithm applied to biological data. *Computers & Operations Research*, **37**: 1381–1388.

[24] RAO MR. 1971. Cluster analysis and mathematical programming. *Journal of the American Statistical Association*, **66**: 622–626.

[25] RESENDE MGC & RIBEIRO CC. 2010. Greedy randomized adaptive search procedures: Advances, hybridizations, and applications. In: M. Gendreau and J.-Y. Potvin, editors, Handbook of Metaheuristics. Springer-Verlag, 2nd edition.

[26] ROMANOWSKI CJ, NAGI R & SUDIT M. 2006. Data mining in an engineering design environment: or applications from graph matching. *Computers & OR*, **33**: 3150–3160.

[27] ROMESBURG HC. 2004. Cluster Analysis for Researchers. Lulu Pres (re-print).

[28] WU Z & LEAHY R. 1993. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **15**: 1101–1113.

APPENDIX A: NUMERICAL RESULTS

In this Appendix, we present the numerical results obtained in the experiments. The values reported in Tables 1, 2 and 3 correspond to the data from the graphs displayed in Figures 3, 4 and 5.

Table 1 presents the results obtained by the four algorithms proposed in this paper. The first column shows the name of the used data sets. The names of these data indicate, respectively, the number of objects and its number of *clusters*. For example the data set 100_3 has 100 objects and 3 clusters. In the second to fifth columns, the results of the heuristics are presented, that is, their execution time in seconds (Time) and objective function value (Z). The best results are highlighted in bold.

Table 1 – Proposed method results.

Data Set	CH		GHLS		GRASP-I		GRASP-II	
	Time	Z	Time	Z	Time	Z	Time	Z
100_3	0.031	0.603	0.031	**0.448**	0.296	**0.448**	0.437	**0.448**
100_4	0.032	0.590	0.031	**0.412**	0.234	**0.412**	0.467	**0.412**
100_5	0.015	1.276	0.016	0.968	0.280	**0.901**	0.640	**0.901**
100_10	0.031	1.732	0.031	1.462	0.266	**1.340**	1.124	**1.340**
100_20	0.032	2.046	0.015	1.333	2.122	**1.295**	2.932	**1.295**
100_50	0.094	**1.414**	1.373	1.455	125.675	1.455	126.345	1.455
200_3	0.234	0.086	0.530	0.081	0.951	0.067	2.012	**0.065**
200_4	0.093	1.538	0.764	1.483	1.451	**1.254**	3.837	**1.254**
200_5	0.063	0.396	0.110	0.381	1.092	0.347	5.398	**0.271**
200_10	0.109	0.566	0.234	0.484	1.528	0.420	8.268	**0.390**
200_20	0.187	2.315	0.296	1.932	4.196	1.710	17.146	**1.459**
200_50	0.343	2.439	1.779	1.487	131.118	1.487	141.056	**1.347**
300_3	0.452	0.608	0.796	**0.594**	4.400	**0.594**	6.535	**0.594**
300_4	0.437	0.277	0.515	**0.277**	2.684	**0.277**	8.112	**0.277**
300_5	0.203	1.001	0.296	**0.971**	1.310	**0.971**	5.055	**0.971**
300_10	0.234	1.057	0.374	1.024	3.323	0.989	15.974	**0.954**
300_20	0.328	1.110	0.578	0.985	9.392	0.913	54.585	**0.865**
300_50	0.671	2.151	2.324	1.767	142.725	1.534	175.423	**1.401**
400_3	0.514	0.376	1.123	0.351	8.252	**0.269**	24.804	**0.269**
400_4	0.281	0.591	3.963	0.537	12.496	**0.392**	34.367	**0.392**
400_5	0.374	0.543	0.546	0.489	3.790	0.477	64.896	**0.460**
400_10	0.343	0.850	0.514	0.817	11.966	0.770	52.323	**0.732**
400_20	0.577	1.288	0.890	1.151	15.085	1.112	86.206	**1.037**
400_50	1.139	1.910	3.182	1.590	159.729	1.512	272.581	**1.269**
500_3	18.470	0.137	28.143	**0.111**	35.896	**0.111**	43.758	**0.111**
500_4	0.609	0.301	6.848	0.274	14.601	0.268	54.008	**0.223**
500_5	1.325	0.200	4.337	0.140	31.076	**0.138**	81.869	**0.138**
500_10	0.609	0.993	1.529	0.950	19.016	0.894	61.137	**0.874**
500_20	0.952	0.899	1.357	0.831	22.105	0.800	92.961	**0.706**
500_50	2.106	1.685	4.336	1.530	173.224	1.422	1519.606	**1.244**
600_3	52.151	0.012	50.310	0.012	57.658	**0.012**	66.253	**0.012**
600_4	0.608	0.594	11.919	0.577	25.554	**0.547**	97.969	**0.547**
600_5	1.763	0.230	7.971	0.197	34.367	0.175	119.184	**0.173**
600_10	0.826	0.810	1.701	0.775	32.089	0.770	515.896	**0.730**
600_20	1.514	1.147	2.403	1.079	37.346	1.037	282.159	**0.972**
600_50	3.946	1.749	8.549	1.636	198.683	1.558	617.763	**1.347**
700_3	35.287	0.007	34.975	**0.007**	38.111	**0.007**	46.550	**0.007**
700_4	50.341	0.139	50.982	**0.109**	78.453	**0.109**	132.882	**0.109**
700_5	1.030	0.382	2.496	0.348	26.910	0.345	134.504	**0.313**

Table 1 (continuation) – Proposed method results.

Data Set	CH		GHLS		GRASP-I		GRASP-II	
	Time	Z	Time	Z	Time	Z	Time	Z
700_10	1.342	0.538	1.919	0.504	19.469	0.479	118.374	**0.426**
700_20	3.213	0.456	5.070	0.414	52.603	0.409	239.492	**0.395**
700_50	5.274	1.404	9.781	1.294	230.710	1.269	674.627	**1.124**
800_3	143.193	**0.042**	138.514	**0.042**	143.209	**0.042**	155.861	**0.042**
800_4	165.159	**0.082**	175.766	**0.082**	188.933	**0.082**	245.358	**0.082**
800_5	4.618	0.333	7.270	0.319	55.536	**0.311**	192.209	**0.311**
800_10	2.621	0.738	3.978	0.693	24.118	0.667	157.155	**0.646**
800_20	5.086	0.988	10.608	0.932	103.382	0.926	672.147	**0.903**
800_50	6.067	0.843	11.231	0.795	233.487	0.761	711.161	**0.689**
900_3	56.223	0.112	61.558	**0.104**	75.317	**0.104**	113.989	**0.104**
900_4	291.863	**0.036**	292.034	**0.036**	371.828	**0.036**	503.711	**0.036**
900_5	65.333	0.170	69.872	0.167	136.798	**0.118**	408.614	**0.118**
900_10	2.464	0.335	5.118	0.327	65.457	0.325	274.126	**0.322**
900_20	3.198	0.658	5.397	0.630	71.651	0.613	1004.287	**0.591**
900_50	8.268	0.843	14.633	0.795	280.521	0.759	1300.893	**0.712**
1000_3	243.331	**0.009**	245.733	**0.009**	280.241	**0.009**	289.117	**0.009**
1000_4	274.015	0.055	271.754	**0.055**	301.752	**0.055**	334.420	**0.055**
1000_5	7.987	0.443	35.770	0.442	77.268	0.436	249.710	**0.434**
1000_10	112.945	0.079	117.422	0.071	208.230	0.071	456.896	**0.070**
1000_20	4.712	0.421	8.300	0.406	108.047	0.395	447.504	**0.389**
1000_50	8.175	0.722	16.270	0.670	360.581	0.660	1499.497	**0.629**

Table 2 presents the results of GRASP-II, *Complete Linkage* and CPLEX. The columns GRASP-II and *Complete Linkage* exhibit the computational time in seconds (Time), the objective function value (Z) and the gap of each solution regarding the best solution obtained. The best objective functions and gaps are highlighted in bold. The CPLEX column shows the value of the objective function of the node with the best expectation, lower bound (LB), the value of best integer solution, higher upper limit (UB) and the computational time in seconds (Time). When the LB coincides with the UB, then the solution obtained is optimal. In this case, these results are highlighted in bold. For CPLEX is imposed a time limit of 3 hours (10.800 seconds).

Table 2 – GRASP-II, *Complete Linkage* and CPLEX results.

Data	GRASP-II			*Complete Linkage*			Best	CPLEX		
Set	Time	Z	Gap %	Time	Z	Gap %	Solution	LB	UB	Time
100_3	0.437	**0.448**	**0.00**	0.749	0.497	10.94	0.448	0.448	0.448	9
100_4	0.467	**0.412**	**0.00**	0.702	0.412	**0.00**	**0.412**	0.412	0.412	24
100_5	0.64	**0.901**	**0.00**	0.843	0.968	7.44	0.901	**0.901**	**0.901**	49
100_10	1.124	**1.340**	**0.00**	1.217	1.377	2.76	1.340	0.822	1.340	10800
100_20	2.932	**1.295**	**0.00**	1.887	1.333	2.93	1.295	0.000	1.237	10800
100_50	126.345	**1.455**	**0.00**	5.335	**1.455**	**0.00**	1.455	0.000	1.437	10800
200_3	2.012	**0.065**	**0.00**	3.135	0.086	30.30	**0.065**	**0.065**	**0.065**	462
200_4	3.837	**1.254**	**0.00**	3.744	**1.254**	**0.00**	1.254	**1.254**	**1.254**	517
200_5	5.398	**0.271**	**0.00**	2.808	0.298	9.96	0.271	**0.271**	**0.271**	1912
200_10	8.268	**0.390**	**0.00**	3.947	0.435	11.54	0.390	0.344	0.390	10800
200_20	17.146	**1.459**	**0.00**	4.914	1.726	18.30	1.459	0.000	1.632	10800
200_50	141.056	**1.347**	**0.00**	14.274	1.471	9.21	1.347	0.000	2.253	10801
300_3	6.535	**0.594**	**0.00**	7.769	0.595	0.17	0.594	**0.594**	**0.594**	681
300_4	8.112	**0.277**	**0.00**	11.934	**0.277**	**0.00**	0.277	**0.277**	**0.277**	688
300_5	5.055	**0.971**	**0.00**	11.84	1.001	3.09	0.971	**0.971**	**0.971**	5031
300_10	15.974	**0.954**	**0.00**	18.002	1.028	7.76	0.954	0.785	0.949	10800
300_20	54.585	**0.865**	**0.00**	13.993	0.938	8.44	0.865	0.000	1.186	10802
300_50	175.423	**1.401**	**0.00**	44.74	1.44	2.78	1.401	0.000	2.257	11002
400_3	24.804	**0.269**	**0.00**	15.803	0.348	29.37	0.269	**0.269**	**0.269**	1421
400_4	34.367	**0.392**	**0.00**	13.947	0.544	38.78	0.392	0.225	0.575	10801
400_5	64.896	**0.46**	**0.00**	15.007	0.514	11.74	0.46	0.267	0.479	10800
400_10	52.323	0.732	5.63	11.482	**0.693**	**0.00**	0.693	0.000	0.879	10801
400_20	86.206	1.037	0.29	10.842	**1.034**	**0.00**	1.034	0.000	1.565	10803
400_50	272.581	**1.269**	**0.00**	58.204	1.356	6.86	1.269	0.000	2.260	10800
500_3	43.758	**0.111**	**0.00**	22.636	0.153	37.84	0.111	0.110	0.129	10800
500_4	54.008	**0.223**	**0.00**	22.56	0.298	33.63	0.223	0.181	0.298	10800
500_5	81.869	**0.138**	**0.00**	21.438	0.142	2.90	0.138	0.097	0.197	10800
500_10	61.137	**0.874**	**0.00**	21.199	0.884	1.14	0.874	0.102	1.053	10802
500_20	92.961	**0.706**	**0.00**	18.437	0.744	5.38	0.706	0.000	1.038	10803
500_50	1519.606	**1.244**	**0.00**	82.914	1.311	5.39	1.244	0.000	Inf	10806
600_3	66.253	**0.012**	**0.00**	35.521	0.013	8.33	0.012	**0.012**	**0.012**	7444
600_4	97.969	**0.547**	**0.00**	30.514	0.584	6.76	0.547	0.375	0.586	10801
600_5	119.184	**0.173**	**0.00**	31.606	0.218	26.01	0.173	0.105	0.232	10801
600_10	515.896	**0.73**	**0.00**	27.487	0.748	2.47	0.73	0.000	0.996	10802
600_20	282.159	**0.972**	**0.00**	25.834	1.022	5.14	0.972	0.000	1.297	10803
600_50	617.763	**1.347**	**0.00**	54.741	1.409	4.60	1.347	0.000	Inf	10807
700_3	46.55	**0.007**	**0.00**	50.404	**0.007**	**0.00**	0.007	**0.007**	**0.007**	10801
700_4	132.882	**0.109**	**0.00**	47.081	0.116	6.42	0.109	0.088	0.150	10800
700_5	134.504	**0.313**	**0.00**	51.09	0.326	4.15	0.313	0.108	0.382	10800
700_10	118.374	**0.426**	**0.00**	40.966	0.454	6.57	0.426	0.000	0.606	10802
700_20	239.492	0.395	0.51	35.849	**0.393**	**0.00**	0.393	0.000	0.504	10805

Table 2 (continuation) – GRASP-II, *Complete Linkage* and CPLEX results.

Data	GRASP-II			*Complete Linkage*			Best	CPLEX		
Set	Time	Z	Gap %	Time	Z	Gap %	Solution	LB	UB	Time
700_50	674.627	**1.124**	**0.00**	72.213	1.153	2.58	1.124	0.000	Inf	10810
800_3	155.861	**0.042**	**0.00**	66.098	0.052	23.81	0.042	0.042	0.052	10801
800_4	245.358	**0.082**	**0.00**	61.449	0.102	24.39	0.082	0.054	0.118	10802
800_5	192.209	0.311	5.07	63.433	**0.296**	**0.00**	0.296	0.094	0.342	10800
800_10	157.155	0.646	0.78	60.979	**0.641**	**0.00**	0.641	0.000	1.125	10804
800_20	672.147	0.903	2.27	108.666	**0.883**	**0.00**	0.883	0.000	Inf	10808
800_50	711.161	**0.689**	**0.00**	47.415	0.702	1.89	0.689	0.000	2.202	10808
900_3	113.989	**0.104**	**0.00**	98.885	0.113	8.65	0.104	0.103	0.138	10803
900_4	503.711	**0.036**	**0.00**	79.785	0.044	22.22	0.036	0.031	0.051	10800
900_5	408.614	**0.118**	**0.00**	79.059	0.17	44.07	0.118	0.000	0.175	10802
900_10	274.126	0.322	0.94	69.631	0.319	**0.00**	**0.319**	0.000	0.389	10804
900_20	1004.287	**0.591**	**0.00**	67.666	0.605	2.37	0.591	0.000	0.816	10806
900_50	1300.893	**0.712**	**0.00**	124.647	0.72	1.12	0.712	0.000	Inf	820
1000_3	289.117	**0.009**	**0.00**	115.173	**0.009**	**0.00**	0.009	0.008	0.010	10802
1000_4	334.42	**0.055**	**0.00**	106.926	**0.055**	**0.00**	0.055	0.043	0.076	10802
1000_5	249.71	**0.434**	**0.00**	99.2	0.436	0.46	0.434	0.000	0.490	10802
1000_10	456.896	**0.07**	**0.00**	89.038	0.071	1.43	0.07	0.000	0.110	10804
1000_20	447.504	0.389	2.10	84.401	**0.381**	**0.00**	0.381	0.000	0.511	10808
1000_50	1499.497	0.629	2.78	80.931	**0.612**	**0.00**	0.612	0.000	0.890	1008

Table 3 presents the values of NMI achieved by the solutions found by GRASP-II, *Complete Linkage*, *C-Means* and CPLEX. Again, the best results are marked in bold.

Table 3 – Normalized Mutual Information (NMI).

Data Set	NMI				Best NMI
	GRASP-II	CL	C-Means	CPLEX	
100_3	0.912	**0.951**	0.782	0.923	0.951
100_4	0.917	**1.000**	0.591	0.877	1.000
100_5	0.786	**0.865**	0.707	0.721	0.865
100_10	0.643	**0.803**	0.765	0.500	0.803
100_20	0.888	0.895	0.587	**0.898**	0.898
100_50	0.788	**0.797**	0.743	0.789	0.797
200_3	0.647	0.646	**0.810**	0.667	0.810
200_4	**0.946**	0.898	0.797	0.366	0.946
200_5	0.918	**0.942**	0.664	0.899	0.942
200_10	0.804	**0.934**	0.541	0.710	0.934
200_20	0.818	**0.916**	0.374	0.640	0.916
200_50	0.789	**0.855**	0.548	0.676	0.855
300_3	0.783	**0.981**	0.920	0.762	0.981
300_4	0.738	0.802	**0.831**	0.435	0.831
300_5	0.370	**0.836**	0.769	0.117	0.836
300_10	0.475	**0.774**	0.467	0.335	0.774
300_20	0.670	**0.881**	0.425	0.172	0.881
300_50	0.749	**0.837**	0.434	0.601	0.837

Table 3 (continuation) – Normalized Mutual Information (NMI).

Data Set	NMI				Best NMI
	GRASP-II	CL	C-Means	CPLEX	
400_3	**0.981**	0.975	0.907	0.977	0.981
400_4	**0.979**	0.961	0.855	0.359	0.979
400_5	0.625	**0.812**	0.683	0.328	0.812
400_10	0.388	**0.929**	0.561	0.078	0.929
400_20	0.623	**0.866**	0.489	0.134	0.866
400_50	0.771	**0.883**	0.352	0.309	0.883
500_3	**0.938**	0.860	0.874	0.759	0.938
500_4	**0.934**	0.817	0.809	0.160	0.934
500_5	0.910	**0.951**	0.724	0.143	0.951
500_10	0.328	**0.924**	0.505	0.064	0.924
500_20	0.662	**0.841**	0.501	0.201	0.841
500_50	0.700	**0.894**	0.343	0.000	0.894
600_3	**0.869**	0.765	0.752	0.707	0.869
600_4	0.903	**0.926**	0.765	0.075	0.926
600_5	0.851	**0.861**	0.762	0.113	0.861
600_10	0.784	**0.899**	0.618	0.000	0.899
600_20	0.556	**0.905**	0.415	0.110	0.905
600_50	0.698	**0.892**	0.357	0.000	0.892
700_3	0.693	0.745	**0.776**	0.456	0.776
700_4	**0.944**	0.911	0.804	0.073	0.944
700_5	0.627	**0.914**	0.651	0.026	0.914
700_10	0.658	**0.819**	0.505	0.048	0.819
700_20	0.545	**0.901**	0.584	0.082	0.901
700_50	0.629	**0.883**	0.239	0.000	0.883
800_3	0.812	**0.903**	0.865	0.091	0.903
800_4	0.838	**0.869**	0.804	0.034	0.869
800_5	0.521	**0.904**	0.777	0.023	0.904
800_10	0.330	**0.896**	0.524	0.000	0.896
800_20	0.544	**0.922**	0.451	0.000	0.922
800_50	0.590	**0.823**	0.383	0.000	0.823
900_3	**0.943**	0.924	0.915	0.015	0.943
900_4	0.840	**0.845**	0.728	0.012	0.845
900_5	**0.877**	0.836	0.709	0.009	0.877
900_10	0.226	**0.935**	0.527	0.000	0.935
900_20	0.429	**0.878**	0.481	0.000	0.878
900_50	0.563	**0.838**	0.316	0.000	0.838
1000_3	0.730	0.769	**0.790**	0.410	0.790
1000_4	0.827	**0.961**	0.796	0.015	0.961
1000_5	0.817	**0.915**	0.741	0.000	0.915
1000_10	0.621	**0.764**	0.534	0.000	0.764
1000_20	0.367	**0.886**	0.495	0.000	0.886
1000_50	0.571	**0.872**	0.329	0.000	0.872