2009 Ninth International Conference on Intelligent Systems Design and Applications

Comparison Among Methods for k Estimation in k-means

Murilo C. Naldi André Fontana Ricardo J. G. B. Campello
Computer Science Department
University of São Paulo (USP) at São Carlos
São Carlos, Brazil
murilocn@icmc.usp.br, fon@grad.icmc.usp.br, campello@icmc.usp.br

***Abstract*—One of the most influential algorithms in data mining, k-means, is broadly used in practical tasks for its simplicity, computational efficiency and effectiveness in high dimensional problems. However, k-means has two major drawbacks, which are the need to choose the number of clusters, k, and the sensibility to the initial prototypes' position. In this work, systematic, evolutionary and order heuristics used to suppress these drawbacks are compared. 27 variants of 4 algorithmic approaches are used to partition 324 synthetic data sets and the obtained results are compared.**

I. Introduction

Clustering is one of the main tasks in data mining and consists of the organization of a data set in clusters, according to their similarities [1], [2]. This task has applications in diverse areas, such as market segmentation [3], bioinformatics [4], text mining [5], intrusion detection [6], image processing [7], taxonomy [8], and others.

Clustering algorithms can be divided into two main categories: partitional and hierarchical. Considering the data set $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$, composed by vectors $\mathbf{x}_j \in \Re^n$ (described for n attributes or features), an exclusive partition is a collection $\mathbf{C} = \{C_1, C_2, ..., C_k\}$ of k clusters or subsets C_i in which $C_1 \cup C_2 \cup ... \cup C_k = \mathbf{X}$, $C_i \neq \emptyset$ and $C_i \cap C_l = \emptyset$ for $i \neq l$. Non-exclusive partitions enable objects to belong or have some degree of membership to distinct clusters, while the aplication of hierarchical algorithms results in a hierarchical nested sequence of partitions.

Among the huge variety of clustering algorithms in the literature [2], k-means deserves attention for two main reasons [9]: (i) it is considered one of the most influential algorithms in data mining; and (ii) it is simple and scalable, having linear asymptotic execution time in relation to any parameter. However, it presents two major drawbacks [9]: the need for the number of clusters k as an input and sensibility to the initial position of their respective prototypes. These issues can make its practical usage difficult, as the way the data set is distributed is unknown for most clustering applications. For this reason, methods were developed to suppress these drawbacks. The most simple one consists of systematic repetitions of k-means with increasing or randomly draw values of k in an interval defined by the user. The resulting partitions are evaluated and the best is chosen. However, sweeping incrementally across k values can be inefficient for datasets with large numbers of clusters and random draws can be inefficient for large intervals. Another method consists of using algorithms that executes the fine tuning of partitions obtained from k-means. Evaluating the obtained partitions enables these algorithms to estimate the data set's approximate number of clusters. This is the case of the algorithms Bisecting k-means [10], which generates a hierarchical partition from the iterative division of the dataset by k-means, and X-means [11], which uses k-means to guide the dataset's division and, then, refines the resulting partition.

In addition to the methods described, hybridization of k-means with other general proposal meta-heuristics were adapted to the clustering problem [12]. A meta-heuristic that deserves attention is the evolutionary algorithms (EAs), broadly used in clustering problems [13]. The main interest of this work are EAs that use k-means as a local search operator, to refine the global search made by the evolutionary method. There is not much research developed with evolutionary operators for a variable number of clusters [13]. In particular, the Evolutionary Algorithm for Clustering (EAC) was developed aiming to evolve partitions with variable k using elimination, division and agglomeration of clusters systematically refined by k-means. The computational efficiency of EAC was improved in [14] with the usage of guided mutation operators, self-adjustment of application rates, among other features, which gave origin to the Fast Evolutionary Algorithm for Clustering (F-EAC).

In [15], a new codification and changes in F-EAC data structures reduced the computational time considerably in relation to the original version of the algorithm, published in [14]. In the present paper, an experimental methodology is used to compare and estimate this algorithm with others also based on k-means and capable of estimating the number of clusters k, known as: ordered and random systematic executions of k-means; variations of Bisecting k-means, variations of X-means; and variations of F-EAC that incur in stochastic local search procedures. All algorithms are compared in terms of the obtained partition quality and computational performance for the 324 data sets.

The other sections of this article are organized as follows: in Section II, algorithms of interest for this work based on k-means are described; in Section III the methodology

978-0-7695-3872-3/09 $26.00 © 2009 IEEE
DOI 10.1109/ISDA.2009.78

and the experimental results comparing these algorithms are presented; the conclusions are reported in Section IV.

II. k-MEANS BASED ALGORITHMS

The k-means and its variants have been under investigation for half a century [16], especially due to their simplicity, scalability and the fact that they can easily be modified to deal with data streams and large data sets [9]. The k-means version used in all implemented algorithms in this work is described in detail in Algorithm 1[1], in which $\bar{\mathbf{x}}_i$ is the centroid of cluster C_i and $d(\mathbf{x}_j, \bar{\mathbf{x}}_i)$ is the Euclidean distance between $\bar{\mathbf{x}}_i$ and the object $\mathbf{x}_j \in C_i$. The method converges when there are no differences between values of $\bar{\mathbf{x}}_i$ between two consecutive iteractions and is limited to a maximum number of t iteractions, given *a priori*. In this work, this limit is set to $t = 5$ for all implemented algorithms, as empirical evidence suggests that this value is usually sufficient for k-means to find satisfactory solutions [17]. The computational complexity of k-means, in terms of its potentially critical variables, is $O(N \cdot k \cdot n)$ [9]. Following this section, brief descriptions of algorithms capable of determining k will be made.

Algorithm 1 k-means

1: select initial cluster prototypes with attributes in the interval [0,1];
2: **repeat**
3: **for all** $\mathbf{x}_j \in \mathbf{X}$ **do**
4: **for all** $\bar{\mathbf{x}}_i$ **do**
5: compute the dissimilarity $d(\mathbf{x}_j, \bar{\mathbf{x}}_i)$;
6: **end for**
7: assign object $\mathbf{x}_j$ to cluster C_i for which $i = \arg\min_{l=1}^{k} d(\mathbf{x}_j, \bar{\mathbf{x}}_l)$;
8: **end for**
9: **for all** $\bar{\mathbf{x}}_i$ **do**
10: update $\bar{\mathbf{x}}_i$ as the centroid of cluster C_i;
11: **end for**
12: **until** convergence is attained or the number of iterations exceeds a given limit t.

A. Bisecting k-means

The algorithm Bisecting k-means [10] consists of a hierarchical variant of the k-means algorithm. In each interaction, it selects a cluster previously generated and split into two with k-means, producing a hierarchy. The algorithm works as described in Algorithm 2, in which π_a is the current partition, $inter$ is the number of repetitions of Step 5 (not necessarily bigger than 1 [18], which is the case here) and k_{max} is the maximum number of clusters established by the user.

[1]Any other version could be adopted, as long the same version is used in all algorithms investigated here.

Algorithm 2 Bisecting k-means

1: initialize π_a as one cluster with all objects of the data set;
2: **repeat**
3: select a cluster $C_s \in \pi_a$;
4: **for** $i = 1$ to $inter$ **do**
5: split C_s in C'_s and C''_s by k-means and store the result;
6: **end for**
7: select the split made in Step 5 which produces the biggest similarity between objects and their centroids;
8: apply division selected in Step 7 to π_a;
9: **until** one partition with k_{max} clusters is obtained.
10: **return** π_a

The cluster to be split, C_s, can be chosen in many ways (Step 3): from its size (e.g., diameter or no. of objects) or similarity between objects and their centroids (e.g. volume). The complexity of the algorithm varies according to the method used to select C_s. If $k_{max} << N$ and the selection is made from the clusters' number of objects, the algorithm's complexity can be linear [10]. In general, however, the complexity will be quadratic in N at least, that is $O(N^2 \cdot k_{max} \cdot n)$. In this work, C_s is chosen by its diameter, calculated in two ways:

(a) the diameter is calculated as two times the major distance between the centroid and one of the cluster's objects;
(b) the diameter is calculated as two times the mean of the distances between the centroid and all cluster's objects.

In this work, Bisecting k-means was modified to evaluate the need to split the clusters, enabling the algorithm to stop its execution and estimate a value $k < k_{max}$. Cluster validation criteria were used to compare the quality of the solution before dividing it with the quality after the division. The first criterion is the well known Bayesian Information Criterion (BIC - e.g. see [19]). The second is a more computationally efficient version of the popular Average Silhouette Width Criterion [1]. The version used here is called Simplified Silhouette [20].

Once the validation criteria is defined, the following two stopping criteria can be used:

(i) evaluate the split of the cluster with the major diameter. If there is no improvement, the split is not done and execution is finished;
(ii) evaluate the split of all clusters in decreasing order. If there is no improvement in any split, execution is finished.

The combination of the methods to calculate diameter ((a) or (b)) with the stopping criteria ((i) or (ii)) and the method

of evaluation (BIC or Silhouette) results in 8 versions of the Bisecting k-means algorithm studied in the present paper.

B. X-means

The algorithm X-means was proposed by Pelleg & Moore [11] and consists of partitioning the data set into a number of clusters in an interval $[k_{min}, k_{max}]$, defined by the user, using k-means. A simplified version of X-means is described in Algorithm 3, in which π_a is the current partition, π_r is the resulting partition, p is the cluster proportion to be kept in Step 13 and $|\cdot|$ denotes set cardinality.

Algorithm 3 Algorithm X-means

1: apply k-means to create π_a with k_{min} clusters;
2: evaluate π_a;
3: $\pi_r := \pi_a$;
4: **repeat**
5: **for all** $C_i \in \pi_a$ **do**
6: split C_i in C_i' and C_i'' by k-means;
7: evaluate the improvement in π_a resulted from Step 6;
8: **end for**
9: **if** at least one evaluation made in Step 7 improves π_a **then**
10: drop all splits which made the cluster quality worse, evaluated in Step 7;
11: keep the best splits evaluated in Step 7 in which $|\pi_a| \leq k_{max}$;
12: **else**
13: keep the proportion p of the splits having better improvement evaluated in Step 7 in which $|\pi_a| \leq k_{max}$;
14: **end if**
15: k-means is used to fine tune π_a;
16: evaluate π_a;
17: **if** π_a is better evaluated in Step 16 then π_r **then**
18: $\pi_r := \pi_a$
19: **end if**
20: **until** $|\pi_a| = k_{max}$
21: **return** π_r

The BIC criterion is used in [11] to evaluate partitions and clusters, in Steps 2, 7 and 16. Similarly to the algorithm Bisecting k-means (see Section II-A), the X-means was also adapted in this work to use a Simplified Silhouette [20] as evaluation criterion. However, unlike the BIC criterion, which can be calculated locally for the split cluster solely (as the split does not affect the other clusters), Silhouette is calculated considering the complete partition before and after the split of each cluster.

Both versions evaluated by BIC and Silhouette were implemented in Matlab language, which enables the comparison of the computational time with other algorithms under investigation in this work, also implemented using the same language. In addition to these versions, implemented as described in the original paper [11][2], another two versions of the algorithm were considered to compare the quality of the obtained partition. One of them is the Pelleg's C version available at http://www.cs.cmu.edu/~dpelleg/kmeans.html. The code can be obtained by personal correspondence with the author. Another version was developed in Java and distributed in the developer's version of Weka Software (version 3.5), available at http://www.cs.waikato.ac.nz/ml/weka/. There are some differences between these implementations and the original paper [11], commented in the source code and discussed in the author's mailing list (see the URL above). All versions use $p = 0.5$ in Step 13.

C. Systematic (repetitive) Methods

The determination of the number of clusters k systematically is generally a procedure consisting of two steps. Firstly, the algorithm is executed multiple times, having distinct prototypes initialization, with the number of clusters varying in a predefined interval. Then, a relative validation criterion is used to estimate the quality of the generated partitions. In thi work, the Simplified Silhouette is used [20].

Two systematic approaches are studied: the Ordered Multiple Runs of k-means (OMRk) and Multiple Runs of k-means (MRk). OMRk consists of executing the k-means algorithm increasing the value of the number of clusters $k = 2, \cdots, k_{max}$. For each k value, n_p partitions must be created with different initial prototypes. The partition π_r with the highest validation value is chosen among the obtained solutions. A common practice consists of assuming $k_{max} = \sqrt{N}$ [22], [23], which will be used here. Previous experiences and studies also suggest that the usage of $n_p = 10$ or $n_p = 20$ to be enough to find partitions of reasonable quality for a wide variety of data sets [15], [24] and, therefore, these values will be adopted here.

Algorithm 4 presents an example of this method, as implemented here, in which π_a is the current partition and k_{max} is the maximum number of clusters permitted for the resulting partitions.

MRk consists of executing k-means multiple times with random k values drawn between 2 and k_{max}. The generated partitions are evaluated with Simplified Silhouette criterion and the best is returned by the algorithm, halted after a stopping criterion $\mathsf{S_C}$ is satisfied. Possible choices for the practical use of this criterion are the interruption of the algorithm after a defined number of k-means executions or after a determined time limit. The first option will be used here, with the number of k-means executions equal to those executed by OMRk, which will be $n_p \cdot (k_{max} - 2 + 1)$ with n_p assuming 10 or 20. The MRk is ilustrated in Algorithm 5.

[2]Except for the BIC implementation, whose log-likelihood function follows the classical description [21] instead of the approximation adopted in [11].

Algorithm 4 Ordered Multiple Runs of k-means

1: **for** $k = 2, ..., k_{max}$ **do**
2: **for** $i = 1, ..., n_p$ **do**
3: create a partition with k clusters with random prototypes;
4: apply *k-means*;
5: evaluate the resulting partition π_a;
6: **if** π_a has the best validation value **then**
7: $\pi_r := \pi_a$;
8: **end if**
9: **end for**
10: **end for**
11: **return** π_r;

Algorithm 5 Multiple Runs of k-means

1: **repeat**
2: randomly select $k \in \{2, ..., k_{max}\}$;
3: create a partition with k clusters with random prototypes;
4: apply *k-means*;
5: evaluate the resulting partition π_a;
6: **if** π_a has the best validation value **then**
7: $\pi_r := \pi_a$;
8: **end if**
9: **until** stopping criterion $\mathsf{S_C}$ is satisfied.
10: **return** π_r;

The complexity of OMRk and MRk is estimated as $O(N \cdot k_{max}^2 \cdot n)$. More details about these algorithms, including a detailed asymptotic analysis, can be found in [15].

D. F-EAC

The Fast Evolutionary Algorithm for Clustering (F-EAC) [14], an improvement of the well established EAC [13], [20], [25], was developed to evolve partitions generated by the use of k-means and evolutionary operators. These partitions are represented by individuals, that in turn are codified by genotypes, whose set is called population. Algorithm 6 presents the main F-EAC steps, in which g is the current generation, P_g is the current population, $|P|$ is the population size and $\mathsf{S_C}$ is the stopping criterion.

The Simplified Silhouette [20] is used here once more, now as the fitness function (Step 6). An elitist strategy keeps the best solution (Step 9) [26]. The other solutions are chosen by a selection operator (e.g., proportional selection such as the roulette selection [26] − Step 10).

Some possible stopping criteria $\mathsf{S_C}$ for F-EAC, applied in Step 17, can be: the definition of a maximum number of generations, a threshold for population diversity, and others [27]. In this work, $\mathsf{S_C}$ is satisfied if the best obtained fitness value keeps constant for $g_{\mathsf{S_C}}$ consecutive generations. Different values for $g_{\mathsf{S_C}}$ will be evaluated.

Algorithm 6 F-EAC.

1: $g \leftarrow 1$;
2: initialize randomly a population P_g;
3: **repeat**
4: **for** $i = 1, ..., |P|$ **do**
5: apply the k-means algorithm to each genotype;
6: evaluate each genotype according to the fitness function;
7: **end for**
8: **if** $\mathsf{S_C}$ is not satisfied **then**
9: apply elitist strategy;
10: select genotypes from P_g;
11: **for all** selected genotypes **do**
12: select which clusters will be mutated by proportional selection;
13: apply the mutation operators in the selected clusters to create new genotypes;
14: **end for**
15: copy the new genotype to the next population P_{g+1} and increments g;
16: **end if**
17: **until** $\mathsf{S_C}$ is satisfied
18: **return** P_g;

One parameter of F-EAC is the population size $|P|$. Empirical evidence suggests that this type of algorithm is robust for distinct value choices for this parameter [14], [15], [25] and values such as $|P| = 10$ enable the algorithm to obtain good partitions in reasonable computational time. This value will be adopted in the present work.

The F-EAC uses two cluster oriented mutation operators. The first eliminates one or more clusters, adding its objects into the clusters with the closest centroids. The second splits one or more clusters into two new clusters each. The proportion of application between the two operators is adjusted dynamically based on the performance each obtained in the previous generation. One simple way of doing this, adopted in this work, is considering the performance of the operators individually for each genotype. If the usage of one operator generated a child with a fitness higher than its father, this operator will be used in the mutation of this child afterwards. Otherwise, the other operator will be used. If the genotype belongs to the initial population or was selected by elitism, it has 50% of chance of being mutated by each operator.

The complexity of the F-EAC is estimated as $O(N \cdot \hat{k}_{max} \cdot n)$ [15]. For a more detailed description of the algorithm, its data structure, evolutionary operators and complexity, see [14], [15].

E. Evolutionary local search methods

Besides the F-EAC, another two variants of this algorithm that incur in stochastic local search procedures will be evaluated in this work. These variants use deterministic

selection methods based on ranks: the (μ, λ) and the $(\mu+\lambda)$ [28]. The (μ, λ) method consists of generating λ children solution by applying evolutionary operators in μ parent solutions. After that, the μ best solutions are selected from the resulted children solutions. The other method, $(\mu + \lambda)$, selects the μ best solutions from the union of the parent with the children solutions. Besides deterministic selection, the F-EAC variants uses $\mu = 1$, with the objective of carrying out an evolutionary search without the need of a population, which incurs in a local search strategy of the hill-climbing stochastic type. Different values of λ are taken into consideration in this work. The initial solutions are generated randomly with $k \in \{2, ..., k_{max}\}$, as done for each individual of F-EAC's population.

III. Experiments

The main objective of the experiments presented here is to compare the algorithms described in Section II in terms of quality of the resulted partitions and computational performance. To achieve it, a set of 324 data sets was artificially generated as described in [29], although with more objects ($N = 1000$). This set consists of 108 data set types, each of them resulting from a combination of the following characteristics: number of attributes $n = 2, ..., 10$; number of clusters $k = 4, 6, 8, 10$; and three methods to balance the objects between the clusters. Three distinct data sets were generated for each of the 108 types described, resulting in 324 data sets in total, with attributes normalized in the unitary interval ($[0, 1]$). The quality of the obtained partitions is measured based on the ideal partitions, using the well known Jaccard external criterion (e.g. see [19]). The algorithms described were compared in 27 versions[3]:

1) Bisecting k-means (Section II-A) with the diameter calculated by method (a), using stop criterion (i) and partition evaluated by Silhouette.
2) Bisecting k-means with the diameter by method (a), stopping criterion (ii) and evaluation by Silhouette.
3) Bisecting k-means with the diameter by method (b), stopping criterion (i) and evaluation by Silhouette.
4) Bisecting k-means with the diameter by method (b), stopping criterion (ii) and evaluation by Silhouette.
5) Bisecting k-means with the diameter by method (a), stopping criterion (i) and evaluation by BIC.
6) Bisecting k-means with the diameter by method (a), stopping criterion (ii) and evaluation by BIC.
7) Bisecting k-means with the diameter by method (b), stopping criterion (i) and evaluation by BIC.
8) Bisecting k-means with the diameter by method (b), stopping criterion (ii) and evaluation by BIC.
9) OMRk (Section II-C) with the number of distinct initializations $n_p = 10$ and evaluation by Silhouette.
10) OMRk with $n_p = 20$ and evaluation by Silhouette.
11) MRk (Section II-C) with the number of distinct initializations set as $n_p = 10$ and evaluation by Silhouette.
12) MRk with $n_p = 20$ and evaluation by Silhouette.
13) Local Search (Section II-E): stopping criterion with $g_{\mathsf{S_C}} = 3$ and selection $(1, \lambda)$ with $\lambda = 5$.
14) Local Search: $g_{\mathsf{S_C}} = 5$, selection $(1, \lambda)$ with $\lambda = 5$.
15) Local Search: $g_{\mathsf{S_C}} = 3$, selection $(1, \lambda)$ with $\lambda = 10$.
16) Local Search: $g_{\mathsf{S_C}} = 5$, selection $(1, \lambda)$ with $\lambda = 10$.
17) Local Search: $g_{\mathsf{S_C}} = 3$, selection $(1 + \lambda)$ with $\lambda = 4$.
18) Local Search: $g_{\mathsf{S_C}} = 5$, selection $(1 + \lambda)$ with $\lambda = 4$.
19) Local Search: $g_{\mathsf{S_C}} = 3$, selection $(1 + \lambda)$ with $\lambda = 9$.
20) Local Search: $g_{\mathsf{S_C}} = 5$, selection $(1 + \lambda)$ with $\lambda = 9$.
21) F-EAC with $|P| = 10$ and $g_{\mathsf{S_C}} = 1$ (Section II-D).
22) F-EAC with $|P| = 10$ and $g_{\mathsf{S_C}} = 3$.
23) F-EAC with $|P| = 10$ and $g_{\mathsf{S_C}} = 5$.
24) X-means evaluated by BIC (Section II-B).
25) X-means evaluated by Silhouette.
26) X-means Pelleg's version (source code in C).
27) X-means distributed in Weka (source code in Java).

The implemented algorithms were executed 30 times for each of the 324 data sets, storing their running time and the external Jaccard criterion values of the obtained partitions (related to the ideal partitions of the artificial data). Due to differences between platforms, algorithms 26 (in C) and 27 (in Java) were not compared with the others (in Matlab) in terms of computational time. All experiments involving algorithms implemented in Matlab (version 7.6.0 R2008a) have the same k-means (described in Algorithm 1) and were executed in the same computer: processor Intel Core 2 E8400 with 3 Ghz, 4Gb RAM, operating system Ubuntu 8.04 64 bits Kernel Linux 2.6.24-23-generic.

All algorithms were executed with the number of clusters limited between $k_{min} = 2$ and $k_{max} = \sqrt{N}$ [22], [23], [25]. For the evolutionary algorithms (F-EAC and Evolutionary Local Search) this interval is applied only to the initial solutions, as these algorithms can increase or decrease freely the number of clusters in the partition(s) during the evolutionary process.

The mean values of the Jaccard external criterion obtained for each of the algorithms are presented in Table I.

Table I
Mean values obtained by the compared algorithms for Jaccard criterion.

12	**11**	**10**	**9**	**16**	**23**	**20**	**15**	**22**
0.9639	0.9636	0.9600	0.9562	0.9522	0.9495	0.9468	0.9463	0.9417
19	**14**	**18**	**13**	**21**	**17**	**27**	**7**	**5**
0.9387	0.9371	0.9274	0.9270	0.9197	0.9107	0.8714	0.8593	0.8555
6	**8**	**2**	**4**	**24**	**25**	**26**	**3**	**1**
0.8505	0.8475	0.8387	0.8387	0.8188	0.7530	0.7230	0.7162	0.7110

To determine statistical difference between the Jaccard values obtained for each algorithm, a hypothesis test was used. The test ANalysis Of VAriance (ANOVA) [30] as-

[3]Note that selections $(1, \lambda_1)$ and $(1+\lambda_2)$ are applied to the same number of candidate solutions if $\lambda_2 = \lambda_1 - 1$, which was done here.

sumes that the compared samples are drawn from populations with normal distribution and that the tested random variables have similar variances [31]. As these requirements are not ensured here, the no parametric test of Friedman [32] was applied with $\alpha = 5\%$.

Since the null hypothesis was rejected, which indicates that the compared means are significantly distinct, post-hoc tests were applied to find which differences have a statistical significance. Conclusions involving various means based on comparisons of two algorithms may not be as precise as they should, as the test level of significance is distorted. In fact, the larger the number of comparisons made, the higher the probability to reject the test by chance [30]. Thus, the Nemenyi's post-hoc test is adopted to maintain the significance of the test, which is 95% for the experiments made in this work.

The significant differences between the mean of the Jaccard ranks obtained by Friedman's test are presented in Table II, in descending order. The symbol in position ij of this table indicates the relation between the means resulting from the i-th and the j-th algorithms. If this symbol is $\bigtriangledown$, the mean of the i-th algorithm is statistically lower than the j-th, if it is $\bigtriangleup$ this mean is higher and $\bigcirc$ indicates that there is no statistical difference between the compared means. It is important to note that, unlike Table I, Table II considers the means of the ranks of Jaccard values and not the mean of the Jaccard values themselves.

Table II
STATISTICAL DIFFERENCE BETWEEN THE MEAN RANKS OF JACCARD VALUES OBTAINED BY TH ALGORITHMS UNDER INVESTIGATION.

	12	11	10	9	16	20	15	23	19	14	22	27	18	13	17	21	7	5	4	2	6	8	24	3	1	26	25
12	○	○	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△
11	○	○	○	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△
10	▽	○	○	○	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△
9	▽	▽	○	○	○	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△
16	▽	▽	▽	○	○	○	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△
20	▽	▽	▽	▽	○	○	○	○	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△
15	▽	▽	▽	▽	▽	○	○	○	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△
23	▽	▽	▽	▽	▽	○	○	○	○	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△
19	▽	▽	▽	▽	▽	▽	○	○	○	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△
14	▽	▽	▽	▽	▽	▽	▽	○	○	○	○	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△
22	▽	▽	▽	▽	▽	▽	▽	▽	○	○	○	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△
27	▽	▽	▽	▽	▽	▽	▽	▽	○	○	○	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△	△
18	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	○	○	△	△	△	△	△	△	△	△	△	△	△	△	△	△
13	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	○	○	△	△	△	△	△	△	△	△	△	△	△	△	△	△
17	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	○	○	△	△	△	△	△	△	△	△	△	△	△	△
21	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	○	○	△	△	△	△	△	△	△	△	△	△	△	△
7	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	○	○	○	○	△	△	△	△	△	△	△	△
5	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	○	○	○	○	○	△	△	△	△	△	△	△
4	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	○	○	○	○	○	○	△	△	△	△	△	△
2	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	○	○	○	○	○	○	△	△	△	△	△	△
6	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	○	○	○	○	○	△	△	△	△	△	△
8	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	○	○	○	○	△	△	△	△	△	△
24	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	○	△	△	△	△
3	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	○	○	△	△
1	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	○	○	○	△
26	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	○	○	△
25	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	▽	○

The results presented in Tables I and II show that the highest mean values of Jaccard were obtained with the algorithms based on systematic multiple execution of k-means (algorithms 9 to 12), followed by the evolutionary algorithms (EAs 13 to 23), with no statistical difference between algorithms 9 and 16. Among the EAs, the largest values of g_{Sc} and λ resulted in the highest Jaccard means.

When a fixed value of g_{Sc} is considered, no hegemony of performance can be observed, in terms of mean Jaccard values, between the F-EAC and its evolutionary local search variants. It is important to cite that these algorithms distinguish between themselves only by the type of selection and by the usage of a population of solutions. In what concerns the population usage, previous experiments, including experiments with real data sets from text mining and bioinformatics, indicates that the guided and cluster oriented mutation operators of F-EAC are aggressive and effective at the same time, which allows the algorithm to present good results even for small population sizes like $|P| = 4$ or $|P| = 5$ [15], [20]. In fact, extensive experimental analysis indicates that these algorithms are robust to the choice of $|P|$ [14], [15], [25]. As the usage of one unique solution is the particular case in which $|P| = 1$, it is not surprising that there are no expressive differences in the obtained results, especially considering the fact that the data sets used here are well behaved (volumetric clusters with normal distributions). The above observations are valid apart from the value used for the λ parameter, which is clearly not critical.

Generally, algorithms X-means and Bisecting k-means resulted in the lowest means for the Jaccard index compared to the other algorithms, which can be observed by the test used. The Bisecting k-means present better results than most X-means implementations. The exceptions are the X-means algorithm distributed with Weka (no. 27) and the Bisecting k-means versions implemented with stopping criterion (i) and evaluation by Silhouette (no. 1 and 3). The first showed a high mean of Jaccard ranks, resembling some EAs (namely, 14 and 22), although the mean of the Jaccard values, presented in Table I, is visually much lower in comparison to these EAs. Algorithms 1 and 3, in turn, resulted in the worst means found between the studied algorithms.

In addition to the Jaccard values, the mean values of the execution times were compared for the algorithms implemented in Matlab, presented in ascending order in Table III. To determine statistical significance between the obtained execution times, the same methodology used before with the Friedman test is used again. The results are presented in Table IV, in a similar way as done in Table II. However, the symbol $\bigtriangledown$ in position ij indicates that the mean of the i-th algorithm is higher than the mean of the j-th algorithm. The symbol $\bigtriangleup$ indicates the opposite.

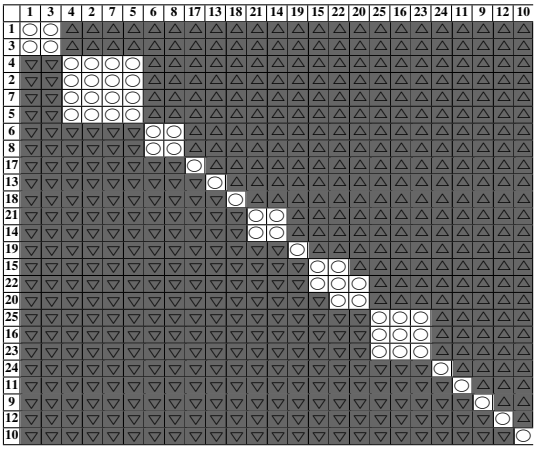
Table III
MEAN EXECUTION TIMES OF THE COMPARED ALGORITHMS.

1	**3**	**4**	**2**	**7**	**5**	**6**	**8**	**17**
0.2210	0.2249	0.4811	0.4834	0.4914	0.4930	1.3153	1.3394	1.7381

13	**18**	**21**	**14**	**19**	**15**	**22**	**20**	**25**
2.0934	2.3146	2.7049	2.8228	3.4297	4.1392	4.1517	4.4671	5.0549

23	**16**	**24**	**11**	**9**	**12**	**10**
5.4596	5.5563	7.8515	11.3113	11.4646	22.6472	22.9463

Table IV
MEAN RANKS OF THE EXECUTION TIMES FROM THE COMPARED ALGORITHMS.



Analyzing the results presented in Tables III and IV, it can be concluded that the versions of the Bisecting k-means algorithm were the fastest of the considered algorithms. However, these execution times are associated to poor Jaccard values (see tables I and II). Although the EAs (algorithms 13 to 23) execution time means higher than those resulted from Bisecting k-means, these means are lower than the obtained by other studied algorithms. The mean times of the EAs increase together with the values of g_{SC}, λ and, consequently, with the mean Jaccard value obtained. The Friedman test suggests statistical differences between the mean ranks of these algorithms.

The implemented X-means algorithms presented execution time means which were higher than those obtained by most of the EAs. The only exceptions were algorithms 16 and 23, which resulted in time means closer to the X-means algorithm of no. 25. However, EAs 16 and 23 obtained high Jaccard mean values, similar or statistically equivalent to systematic algorithms based on multiple executions of k-means (OMRk and MRk). This fact does not occur with any version of X-means.

As expected, the systematic algorithms based on multiple executions of k-means, from number 9 to 12, were the algorithms with highest execution times. Although they are the algorithms with the highest Jaccard means, the mean times obtained by these algorithms are more than 10 times the means of some EAs and can be one hundred times larger than some versions of the Bisecting k-means algorithm. In this context, it is important to mention that, despite the large quantity and diversity of the data sets used in this work, these data sets have well behaved distributions. In previous work, with data sets of higher complexity, algorithms of the family (F-)EAC showed to be more efficient than systematic algorithms based on multiple executions of k-means, not only in terms of computational time, but also in terms of mean quality of the obtained partitions [15], [20]. Unlike the present paper, however, these results were observed in experiments with a reduced amount of data sets.

IV. CONCLUSION

In this work, various versions of algorithms capable of estimating the number of clusters and the initial prototypes to be used by the algorithm k-means were compared to unsupervised data set partitioning. In general, systematic methods of multiple executions of k-means produced partitions of better quality in relation to the ideal partitions (known *a priori*), but with the highest associated computational costs. The evolutionary algorithm F-EAC and variants based on stochastic local searching were capable of obtaining partitions with quality similar to (in some cases, with no statistical relevance) those obtained by systematic methods based on multiple executions of k-means, in considerably less computational time. In addition, the quality of the partitions obtained by these evolutionary algorithms was significantly higher than those obtained by versions of Bisecting k-means and X-means investigated in this work. Although the versions of the Bisecting k-means showed to be the fastest between the investigated algorithms, they produced the worst partitions in terms of mean quality. Comparisons of the algorithms involving different classes of data sets (other than those considered in the present work), possibly real ones, constitute an interesting subject that deserves further investigation.

ACKNOWLEDGMENTS

The authors acknowledge the Brazilian Research Agencies CAPES, CNPq, and FAPESP for financial support.

REFERENCES

[1] L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley, 1990.

[2] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM Computing Surveys*, vol. 31, no. 3, pp. 264–323, 1999. [Online]. Available: citeseer.ist.psu.edu/jain99data.html

[3] J. P. Bigus, *Data Mining with Neural Networks: Solving Business Problems from Application Development to Decision Support*. McGraw-Hill, 1996.

[4] P. Baldi and S. Brunak, *Bioinformatics: The Machine Learning Approach*, ser. Adaptative Computation and Machine Learning. MIT Press, 1998.

[5] R. Feldman and J. Sanger, *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*. Cambridge University Press, 2006.

[6] J. B. D. Cabrera, C. Gutiérrez, and R. K. Mehra, "Ensemble methods for anomaly detection and distributed intrusion detection in mobile ad-hoc networks," *Inf. Fusion*, vol. 9, no. 1, pp. 96–119, 2008.

[7] S. Bandyopadhyay and U. Maulik, "Genetic clustering for automatic evolution of clusters and application to image classification," *Pattern Recognition*, vol. 35, pp. 1197–1208, 2002.

[8] P. Sneath and R. Sokal, *Numerical Taxonomy*. Freeman, 1973.

[9] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J. Hand, and D. Steinberg, "Top 10 algorithms in data mining," *Knowl. Inf. Syst.*, vol. 14, no. 1, pp. 1–37, 2007.

[10] M. Steinbach, G. Karypis, and V. Kumar, "A comparison of document clustering techniques," University of Minnesota, Tech. Rep. 34, 2000.

[11] D. Pelleg and A. Moore, "X-means:a extending k-means with efficient estimation of the number of clusters," in *Proceedings of the Seventeenth International Conference on Machine Learning*, 2000, pp. 727–734.

[12] V. J. Rayward-Smith, "Metaheuristics for clustering in kdd," *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 2380–2387, 2005.

[13] E. Hruschka, R. J. G. B. Campello, A. A. Freitas, and A. C. Ponce Leon F. de Carvalho, "A survey of evolutionary algorithms for clustering," *IEEE Trans. Systems, Man, and Cybernetics, Part C*, vol. 39, no. 2, pp. 133–155, 2009.

[14] V. S. Alves, R. J. G. B. Campello, and E. R. Hruschka, "Towards a fast evolutionary algorithm for clustering," *IEEE Congress on Evolutionary Computation*, pp. 1776–1783, 2006.

[15] M. C. Naldi, R. J. G. B. Campello, E. R. Hruschka, and A. C. P. L. F. Carvalho, "Efficiency issues of evolutionary k-means." *Applied Soft Computing. Submited*, 2009.

[16] D. Steinley, "K-means clustering: A half-century synthesis," *British Journal of Mathematical and Statistical Psychology*, vol. 59, pp. 1–34(34), May 2006. [Online]. Available: http://www.ingentaconnect.com/content/bpsoc/bjmsp/2006/00000059/00000001/art00001

[17] M. R. Anderberg, *Cluster Analysis for Applications*. Academic Press Inc., 1973.

[18] P. A. Pantel, "Clustering by committee," Ph.D. dissertation, Faculty of Graduate Studies and Research − University of Alberta, 2003.

[19] B. Mirkin, *Clustering for Data Mining: A Data Recovery Approach.* Chapman & Hal, 2005.

[20] E. R. Hruschka, R. J. Campello, and L. N. de Castro, "Evolving clusters in gene-expression data," *Information Sciences*, vol. 176, no. 13, pp. 1898–1927, 2006.

[21] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[22] N. Pal and J. Bezdek, "On cluster validity for the fuzzy c-means model," *IEEE Trans. Fuzzy Systems*, vol. 3, no. 3, pp. 370–379, 1995.

[23] M. Pakhira, S. Bandyopadhyay, and U. Maulik, "A study of some fuzzy cluster validity indices, genetic clustering and application to pixel classification." *Fuzzy Sets Systems*, vol. 155, no. 2, pp. 191–214, 2005.

[24] D. J. Hand and W. J. Krzanowski, "Optimising k-means clustering results with standard software packages," *Computational Statistics & Data Analysis*, vol. 49, no. 4, pp. 969 – 973, 2005. [Online]. Available: http://www.sciencedirect.com/science/article/B6V8V-4CX6R65-1/2/ec12c83506474d2e3d666947e5912714

[25] R. J. G. B. Campello, E. R. Hruschka, and V. S. Alves, "On the efficiency of evolutionary fuzzy clustering," *Journal of Heuristics*, vol. 15, no. 1, pp. 43–75, 2009.

[26] M. Mitchell, *An Introduction to Genetic Algorithms*. MIT Press, 1998.

[27] E. Falkenauer, *Genetic Algorithms and Grouping Problems.* John Wiley & Sons, 1998.

[28] T. Bäck, D. B. Fogel, and Z. Michalewicz, *Evolutionary Computation 1: Basic Algorithms and Operators*. Institute of physics publishing, Bristol and Philadelphia, 2000.

[29] G. W. Milligan and M. C. Cooper, "An examination of procedures for determining the number of clusters in a data set," *Psychometrika*, vol. 50, pp. 159–179, 1985.

[30] R. E. Walpole, R. Myers, and S. L. Myers, *Probability & Statistics for Engineers & Scientists*. Macmillan, 2006.

[31] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, 2006.

[32] M. Hollander and D. A. Wolfe, *Nonparametric Statistical Methods*. Wiley-Interscience, 1999.