A Survey of Evolutionary Algorithms for Clustering

Eduardo R. Hruschka, *Member, IEEE*, Ricardo J. G. B. Campello, *Member, IEEE,* Alex A. Freitas, *Member, IEEE*, André C. P. L. F. de Carvalho, *Member, IEEE*

***Abstract* — This paper presents a survey of evolutionary algorithms designed for clustering tasks. It tries to reflect the profile of this area by focusing more on those subjects that have been given more importance in the literature. In this context, most of the paper is devoted to partitional algorithms that look for hard clusterings of data, though overlapping (i.e., soft and fuzzy) approaches are also covered in the manuscript. The paper is original in what concerns two main aspects. First, it provides an up-to-date overview that is fully devoted to evolutionary algorithms for clustering, is not limited to any particular kind of evolutionary approach, and comprises advanced topics, like multi-objective and ensemble-based evolutionary clustering. Second, it provides a taxonomy that highlights some very important aspects in the context of evolutionary data clustering, namely, fixed or variable number of clusters, cluster-oriented or non-oriented operators, context-sensitive or context-insensitive operators, guided or unguided operators, binary, integer or real encodings, centroid-based, medoid-based, label-based, tree-based or graph-based representations, among others. A number of references is provided that describe applications of evolutionary algorithms for clustering in different domains, such as image processing, computer security, and bioinformatics. The paper ends by addressing some important issues and open questions that can be subject of future research.**

***Index Terms* — evolutionary algorithms, clustering, applications.**

I. INTRODUCTION

Clustering is a task whose goal is to determine a finite set of categories (clusters) to describe a data set according to similarities among its objects [75][40]. The applicability of clustering is manifold, ranging from market segmentation [17] and image processing [72] through document categorization and web mining [102]. An application field that has shown to be particularly promising for clustering techniques is bioinformatics [7][13][129]. Indeed, the importance of clustering gene-expression data measured with the aid of *microarray* and other related technologies has grown fast and persistently over the past recent years [74][60].

Clustering techniques can be broadly divided into three main types [72]: overlapping (so-called non-exclusive), partitional, and hierarchical. The last two are related to each other in that a hierarchical clustering is a nested sequence of partitional clusterings, each of which represents a hard partition of the data set into a different number of mutually disjoint subsets. A hard partition of a data set $\mathbf{X}=\{\mathbf{x}_1,\mathbf{x}_2, ...,\mathbf{x}_N\}$, where $\mathbf{x}_j$ (j = 1, ..., N) stands for an n-dimensional feature or attribute vector, is a collection $\mathbf{C}=\{\mathbf{C}_1,\mathbf{C}_2, ...,\mathbf{C}_k\}$ of k non-overlapping data subsets $\mathbf{C}_i\neq\emptyset$ (non-null clusters) such that $\mathbf{C}_1 \cup \mathbf{C}_2 \cup...\cup \mathbf{C}_k = \mathbf{X}$ and $\mathbf{C}_i \cap \mathbf{C}_j = \emptyset$ for $i \neq j$. If the condition of mutual disjunction ($\mathbf{C}_i \cap \mathbf{C}_j = \emptyset$ for $i \neq j$) is relaxed, then the corresponding data partitions are said to be of overlapping type. Overlapping algorithms produce data partitions that can be *soft* (each object fully belongs to one or more clusters) [40] or *fuzzy* (each object belongs to one or more clusters to different degrees) [118][64].

In spite of the type of algorithm (partitional, hierarchical or overlapping), the main goal of clustering is maximizing both the homogeneity within each cluster and the heterogeneity among different clusters [72][3]. In other words, objects that belong to the same cluster should be more similar to each other than objects that belong to different clusters. The problem of measuring similarity is usually tackled indirectly, i.e., distance measures are used for quantifying the degree of dissimilarity among objects, in such a way that more similar objects have lower dissimilarity values [73]. Several dissimilarity measures can be employed for clustering tasks [72][132]. Each measure has its bias and comes with its own advantages and drawbacks. Therefore, each one may be more or less suitable to a given analysis or application scenario. Indeed, it is well-known that some measures are more suitable for gene clustering in bioinformatics [74], whereas other measures are more appropriate for text clustering and document categorization [114], for instance.

Clustering is deemed one of the most difficult and challenging problems in machine learning, particularly due to its unsupervised nature. The unsupervised nature of the problem implies that its structural characteristics are not known, except if there is some sort of domain knowledge available in advance. Specifically, the spatial distribution of the data in terms of the number, volumes, densities, shapes, and orientations of clusters (if any), are unknown [47]. These adversities may be potentialized even further by an eventual need for dealing with data objects described by attributes of distinct natures (binary, discrete, continuous, and categorical), conditions (complete and partially missing) and scales (ordinal and nominal) [72][73].

From an optimization perspective, clustering can be formally considered as a particular kind of NP-hard grouping

E. R. Hruschka, R. J. G. B. Campello, and A. C. P. L. F. de Carvalho are with the Department of Computer Sciences of the University of São Paulo (USP) at São Carlos, SP, Brazil. E-mails: {erh;campello;andre}@icmc.usp.br.

A.A. Freitas is with the Computer Science Department of the University of Kent at Canterbury, Kent, UK, E-mail: A.A.Freitas@kent.ac.uk.

The authors acknowledge the Brazilian Research Agencies CNPq and FAPESP for their financial support to this work.

problem [43]. This has stimulated the search for efficient approximation algorithms, including not only the use of *ad hoc* heuristics for particular classes or instances of problems, but also the use of general-purpose metaheuristics (e.g. see [116]). Particularly, evolutionary algorithms are metaheuristics widely believed to be effective on NP-hard problems, being able to provide near-optimal solutions to such problems in reasonable time. Under this assumption, a large number of evolutionary algorithms for solving clustering problems have been proposed in the literature. These algorithms are based on the optimization of some objective function (i.e., the so-called fitness function) that guides the evolutionary search.

This paper presents a survey of evolutionary algorithms designed for clustering tasks. It tries to reflect the profile of this area by focusing more on those subjects that have been given more importance in the literature. In this context, most of the paper is devoted to partitional algorithms that look for hard data clusterings, though overlapping approaches are also covered in the manuscript. It is important to stress that comprehensive surveys on clustering have been previously published, such as the outstanding papers by Jain et al. [73], Jiang et al. [74], and Xu and Wunsch II [132], just to mention a few. Nevertheless, to the best of the authors' knowledge, none has been fully devoted to evolutionary approaches. It is worth mentioning, however, that reviews on similar subjects have been previously published. The authors themselves have previously published overviews on related topics. For instance, in [109] the authors provide an overview of Genetic Algorithms (GAs) for clustering, but only a small subset of the existing evolutionary approaches (namely, GAs) is discussed in that reference. In [50], in its turn, the author provides an extensive review of evolutionary algorithms for data mining applications, but the work focuses on specific evolutionary approaches (GAs and Genetic Programming) and is mainly intended for classification tasks, clustering being just slightly touched in a peripheral section. Three previous monographs [23][43][119] have also partially approached some of the issues raised in the present manuscript. In particular, Cole [23] reviewed a number of genetic algorithms for clustering published until 1997, whereas [119] provided a review of evolutionary algorithms for clustering that is more recent, yet much more concise. In contrast, Falkenauer [43] describes in details a high-level paradigm (meta-heuristic) that can be adapted to deal with grouping problems broadly defined, thus being useful for several applications – e.g., bin packing, economies of scale, conceptual clustering, and equal piles. However, data partitioning problems like those examined in the present paper are not the primary focus of Falkenauer's book [43], which has been published in 1998.

Bearing the previous remarks in mind, it can be stated that the present paper is original in the following two main aspects: (i) It provides an up-to-date overview that is fully devoted to evolutionary algorithms for clustering, is not limited to any particular kind of evolutionary approach, and comprises advanced topics, like multi-objective and ensemble-based evolutionary clustering; and (ii) It provides a taxonomy that allows the reader to identify every work surveyed with respect to some very important aspects in the context of evolutionary data clustering, such as:

- *Fixed* or *variable* number of clusters;
- *Cluster-oriented* or *non-oriented* operators;
- *Context-sensitive* or *context-insensitive* operators;
- *Guided* or *unguided* operators;
- *Binary*, *integer* or *real* encodings;
- *Centroid-based*, *medoid-based*, *label-based*, *tree-based* or *graph-based* representations.

By *cluster-oriented* operators, it is meant here operators that are task dependent, such as operators that copy, split, merge, and eliminate clusters of data objects, in contrast to conventional evolutionary operators that just exchange or switch bits without any regard to their task-dependent meaning. *Guided* operators are those operators that are guided by some kind of information about the quality of individual clusters, about the quality of the overall data partition, or about their performance on previous applications, such as operators that are more likely to be applied to poor quality clusters and operators whose probability of application is proportional to its success (or failure) in previous generations. Finally, *context-sensitivity* will hereafter refer to the original concept as defined by Falkenauer [43], which is limited to crossover operators. In brief, a crossover operator is context-sensitive if: (i) it is cluster-oriented; and (ii) two (possibly different) chromosomes encoding the same clustering solution do not generate a different offspring solution when they are crossed-over. As a consequence, when the number of clusters, k, is fixed in advance, it can be asserted that two chromosomes encoding different clustering solutions with the same k must not produce solutions with a number of clusters other than k as a result of crossover. Of course, *context-sensitivity* is more stringent than *cluster-orientation*.

The remainder of this paper is organized as follows. Section II presents a survey of evolutionary algorithms for hard partitional clustering, whereas Section III presents a review of evolutionary algorithms for overlapping clustering. Section IV discusses evolutionary algorithms for multi-objective clustering and clustering ensembles. A number of references that describe applications of evolutionary algorithms for clustering in different domains is provided in Section V. Finally, the material presented throughout the paper is summarized in Section VI, which also addresses some important issues for future research.

II. HARD PARTITIONAL CLUSTERING

As mentioned in the introduction, a hard partition of a data set $\mathbf{X}$ is a collection of k non-overlapping clusters of these data. The number of clusters, k, usually must be provided in advance by the user. In some cases, however, it can be estimated automatically by the clustering algorithm. Section

II.A describes evolutionary algorithms for which k is assumed to be fixed *a priori*, whereas Section II.B addresses algorithms capable of estimating k during the evolutionary search.

A. Algorithms with Fixed Number of Clusters

Several papers address evolutionary algorithms to solve clustering problems for which the number of clusters (k) is known or set up *a priori* (e.g., Bandyopadhyay and Maulik [10]; Estivill-Castro and Murray [39]; Fränti et al. [48]; Kivijärvi et al. [79]; Krishna and Murty [83]; Krovi [84]; Bezdek et al. [14]; Kuncheva and Bezdek [85]; Lu et al. [95][94]; Lucasius et al. [96]; Maulik and Bandyopadhyay [100]; Merz and Zell [103]; Murthy and Chowdhury [107]; Scheunders [121]; Sheng and Liu [122]). Cole [23] reviews and empirically assesses a number of such genetic algorithms for clustering published up to 1997.

It is intuitive to think of algorithms that assume a fixed number of clusters (k) as being particularly suitable for applications in which there is information regarding the value of k. For instance, domain knowledge may be available that suggests a reasonable value – or a small interval of values – for k. Having such information in hand, algorithms described in this section can be potentially applied for tackling the corresponding clustering problem. Alternatively, the reader may think about using conventional clustering algorithms for fixed k, such as k-means [101][72], EM (Expectation Maximization) [34][61], and SOM (Self-Organized Maps) [17][62] algorithms. However, these prototype-based algorithms are quite sensitive to initialization of prototypes[1] and may get stuck at sub-optimal solutions. This is a well-known problem, which becomes more evident for more complex data sets[2]. A common approach to alleviate this problem involves running the algorithm repeatedly for several different prototype initializations. Nevertheless, note that one can only guarantee that the best clustering solution for a fixed value of k would be found if all possible initial configurations of prototypes were evaluated. Of course, this approach is not computationally feasible in practice, especially for large data sets and large k. Running the algorithm only for a limited set of initial prototypes, in turn, may be either inefficient or not computationally attractive, depending on the number of prototype initializations to be performed.

For this reason, other approaches have been investigated. Among them, evolutionary algorithms have shown to be promising alternatives. Evolutionary algorithms essentially evolve clustering solutions through operators that use probabilistic rules to process data partitions sampled from the search space [43]. Roughly speaking, more fitted partitions have higher probabilities of being sampled. Thus, the evolutionary search is biased towards more promising clustering solutions and tends to perform a more computationally efficient exploration of the search space than traditional randomized approaches (e.g., multiple runs of k-means). Besides, traditional randomized approaches do not make use of the information on the quality of previously assessed partitions to generate potentially better partitions. For this reason, these algorithms tend to be less efficient (in a probabilistic sense) than an evolutionary search.

In spite of the theoretical advantages (in terms of computational efficiency) of evolving clustering solutions, much effort has also been undertaken towards showing that evolutionary algorithms can provide partitions of better quality than those found by traditional algorithms. In fact, this may be possible provided that the parallel nature of the evolutionary algorithms allows them to handle multiple solutions, possibly guided by different distance measures and different fitness evaluation functions.

This section reviews a significant part of the literature on evolutionary algorithms for partitioning a data set into k clusters. Potential advantages and drawbacks of each algorithm are analyzed under the light of their corresponding encoding schemes, operators, fitness functions, and initialization procedures.

1) Encoding Schemes: Several encoding schemes have been proposed in the literature. In order to explain them, let us consider a simple pedagogical data set (Table I) formed by 10 objects $\mathbf{x}_i$ (i = 1, 2,…,10) with two attributes each (n = 2), denoted a_1 and a_2. Such objects have been arbitrarily grouped into three clusters ($\mathbf{C}_1$, $\mathbf{C}_2$, and $\mathbf{C}_3$). These clusters are depicted in Fig. 1 and are used to illustrate how partitions can be encoded to be processed by an evolutionary search. Aiming at summarizing common encodings found in the literature, we first here categorize them into three types: *binary*, *integer*, and *real*.

TABLE I. PEDAGOGICAL DATA SET.

Object ($\mathbf{x}_i$)	a_1	a_2	Cluster - $\mathbf{C}_j$
$\mathbf{x}_1$	1	1	Cluster 1 ($\mathbf{C}_1$)
$\mathbf{x}_2$	1	2	Cluster 1 ($\mathbf{C}_1$)
$\mathbf{x}_3$	2	1	Cluster 1 ($\mathbf{C}_1$)
$\mathbf{x}_4$	2	2	Cluster 1 ($\mathbf{C}_1$)
$\mathbf{x}_5$	10	1	Cluster 2 ($\mathbf{C}_2$)
$\mathbf{x}_6$	10	2	Cluster 2 ($\mathbf{C}_2$)
$\mathbf{x}_7$	11	1	Cluster 2 ($\mathbf{C}_2$)
$\mathbf{x}_8$	11	2	Cluster 2 ($\mathbf{C}_2$)
$\mathbf{x}_9$	5	5	Cluster 3 ($\mathbf{C}_3$)
$\mathbf{x}_{10}$	5	6	Cluster 3 ($\mathbf{C}_3$)

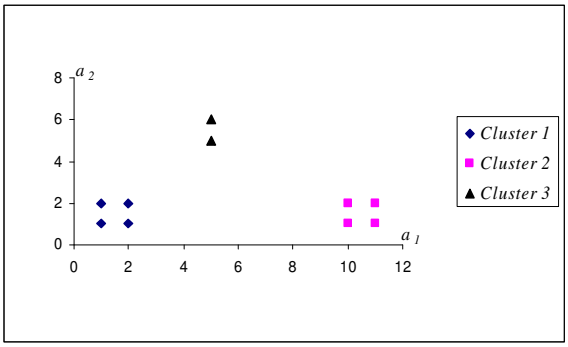

Fig. 1. Pedagogical data set (see Table I).

[1] We here define a *prototype* as a particular feature vector that represents a given cluster. For instance, prototypes can be centroids, medoids, or any other vector computed from the data partition and that represents a cluster (as in the case of typical fuzzy clustering algorithms).

[2] Complexity here refers to the number of different local minima and the variance of their objective function values, which are usually strongly related to the number n of data attributes and the number k of clusters.

a) *Binary encoding*: In a *binary encoding*, each clustering solution (partition) is usually represented as a binary string of length N, where N is the number of data set objects. Each position of the binary string corresponds to a particular object, i.e., the ith position (gene) represents the ith object. The value of the ith gene is 1 if the ith object is a prototype and zero otherwise. For example, the partition depicted in Fig. 1 can be encoded by means of the string [1000100010], in which objects 1, 5, and 9 are cluster prototypes. Clearly, such an encoding scheme inexorably leads to a *medoid-based representation*, i.e., a prototype-based representation in which the cluster prototypes necessarily coincide with objects from the data set. The partition encoded into a given genotype[3] can be derived by the nearest prototype rule – taking into account the proximities between objects and prototypes – in such a way that the ith object is assigned to the cluster represented by the closer (i.e. the most similar) prototype. Kuncheva and Bezdek [85] make use of this encoding approach, which allows the evolutionary search to be performed by means of those classical GA operators originally developed to manipulate binary genotypes [54][105]. However, the use of such classical operators usually suffers from serious drawbacks in the specific context of evolutionary clustering, as will be further discussed in Section II.A.2.a.

There is an alternative way to represent a given data partition using a *binary encoding*. It is the use of a $k \times N$ matrix in which the rows represent clusters and the columns represent objects. In this case, if the jth object belongs to the ith cluster, then 1 is assigned to the ith element of the jth column of the genotype, whereas the other elements of the same column receive 0. For example, using this representation, the partition depicted in Fig. 1 would be encoded as [14]:

1	1	1	1	0	0	0	0	0	0
0	0	0	0	1	1	1	1	0	0
0	0	0	0	0	0	0	0	1	1

This *matrix-based binary encoding* scheme has the clear disadvantage of requiring $O(k \cdot N)$ memory space, against $O(N)$ space of the usual *string-based binary encoding* scheme formerly described. On the other hand, the time it requires to recover the data partition from a given genotype is $O(k \cdot N)$ – both in the average and worst cases – against $O(k \cdot n \cdot N)$ for the string-based scheme (due to the nearest prototype rule computations)[4]. This computational saving is relevant only for data sets with many attributes. When the number of attributes n is not large, the advantage of the matrix-based scheme reduces to the possibility of extending it to handle soft partitions, by allowing multiple elements of a given column to be non-null. Soft partitional clustering is discussed in Section III.

b) *Integer encoding*: There are two ways of representing clustering solutions by means of *integer encoding*. In the first one, a genotype is an integer vector of N positions, where N is the number of data set objects. Each position corresponds to a particular object, i.e., the ith position (gene) represents the ith data set object. Provided that a genotype represents a partition formed by k clusters, each gene has a value over the alphabet $\{1, 2, 3, \ldots, k\}$. These values define the cluster labels, thus leading to a *label-based representation*. For example, the integer vector [1111222233] represents the clusters depicted in Fig. 1. This encoding scheme is adopted in [84][107][83][95][94]. In particular, only partitions formed by two clusters are addressed in [84], thus allowing the use of a binary representation for which each gene has a value over the alphabet $\{0, 1\}$.

This *integer encoding* scheme is naturally redundant, i.e., the encoding is *one-to-many*. In fact, there are $k!$ different genotypes that represent the same solution. For example, there are 3! different genotypes that correspond to the same clustering solution represented in Fig. 1, namely: [1111222233], [1111333322], [2222111133], [2222333311], [3333111122], and [3333222211]. Thus, the size of the search space to be explored by the genetic algorithm is much larger than the original space of solutions. Depending on the employed operators, this augmented space may reduce the efficiency of the genetic algorithm. An alternative to solve this problem is the use of a *renumbering procedure* [43].

Another way of representing a partition by means of an *integer encoding* scheme involves using an array of k elements to provide a *medoid-based representation* of the data set. In this case, each array element represents the index of the object $\mathbf{x}_i$, $i = 1, 2, \ldots, N$ (with respect to the order the objects appear in the data set) corresponding to the prototype of a given cluster. As an example, the array [1 5 9] can represent a partition in which objects 1, 5, and 9 are the cluster prototypes (medoids) of the data given in Table I. Taking into account these prototypes and assuming a nearest prototype rule for assigning objects to clusters, the partition depicted in Fig. 1 can be recovered. Lucasius et al. [96], for instance, make use of this approach. This representation scheme is also adopted, for instance, in [39] and [122].

Conceptually speaking, representing medoids by means of an integer array of k elements, as previously discussed, is usually more computationally efficient than using the *string-based binary encoding* scheme

[3] The terms *genotype*, *chromosome* and *individual* usually have the same meaning in the literature on evolutionary algorithms and will be freely interchanged in this paper.

[4] Actually, the nearest neighbor search can be performed in asymptotic logarithmic time by exploiting the Delaunay triangulation [81], which is the dual of the Voronoi diagram – e.g., see [98]. However, to the best of our knowledge this idea has not been explored in the context of evolutionary algorithms for clustering.

described in Section II.A.1.a. However, it must be noticed that such an *integer encoding* scheme may be redundant if unordered genotypes are allowed, in which case the solutions [1 5 9], [1 9 5], [5 1 9], [5 9 1], [9 1 5], and [9 5 1] encode the same partition depicted in Fig. 1. In such a case, a renumbering procedure should be used in order to avoid potential redundancy problems.

When comparing the two different *integer encoding* schemes discussed in this section, one has to take into account some different aspects that may be of interest. Considering space complexity issues, the *integer encoding* is $O(N)$ when a *label-based representation* is used, whereas it is $O(k)$ when a *medoid-based representation* is adopted. Thus, in principle, one may conclude that the latter is more advantageous than the former (since k is typically much lower than N). However, this is not necessarily true. Actually, the suitability of each of the aforementioned encoding schemes is highly dependent upon the fitness function used to guide the evolutionary search, as well as upon the evolutionary operators that manipulate the clustering solutions being evolved – as it will become evident in the following sections. In brief, the *label-based* encoding does not require any additional processing to make available the information on the membership of each object to its corresponding cluster. Such information may be necessary for computing cluster statistics, which, by their turn, can be needed for computing the fitness function and/or for guiding the application of evolutionary operators. It is easy to see that, contrarily to the *label-based* encoding, the *medoid-based* encoding requires further processing in order to recover the clusters encoded into the genotype. Consequently, depending on the computational cost involved in cluster recovering, a particular encoding may become more (or less) suitable for a given clustering problem.

c) *Real encoding*: In *real encoding* the genotypes are made up of real numbers that represent the coordinates of the cluster prototypes. This means that, unlike the *integer encoding* scheme discussed in Section II.A.1.b, *real encoding* is necessarily associated with a *prototype-based representation* of partitions. However, unlike the *string-based binary encoding* scheme discussed in Section II.A.1.a, *real encoding* does not necessarily leads to a *medoid-based representation*. Instead, it may also be (and in fact usually is) associated with a *centroid-based representation* of the partitions, as discussed in the sequel.

If genotype i encodes k clusters in an n dimensional space, $\Re^n$, then its length is $n \cdot k$. Thus, the first n positions represent the n coordinates of the first cluster prototype, the next n positions represent the coordinates of the second cluster prototype, and so forth. To illustrate this, the genotype [1.5 1.5 10.5 1.5 5.0 5.5] encodes the prototypes (1.5, 1.5), (10.5, 1.5), and (5.0, 5.5) of clusters $\mathbf{C}_1$, $\mathbf{C}_2$, and $\mathbf{C}_3$ in Table I, respectively. Given the genotype, the corresponding clusters can be recovered by the nearest prototype rule, in such a way that the ith object is assigned to the cluster represented by the most similar prototype.

The genotype representation adopted in references [121][100][103][10] follows a *real encoding* scheme in which the prototype locations are not restricted to the positions of the objects. This representation, named *centroid-based representation*, is also adopted by Fränti et al. [48] and Kivijärvi et al. [79]. These authors, however, additionally encode into the genotype a partitioning table that describes, for each object, the index of the cluster to which the object belongs.

Alternatively, one could encode the real-valued coordinates of a set of k medoids. In order to do so, it is only necessary to enforce the constraint that the prototype locations coincide with positions of objects in the data set. In the pedagogical example of Table I, the coordinates of a set of objects – e.g. $\{\mathbf{x}_1, \mathbf{x}_5, \mathbf{x}_9\}$ – can be represented by the genotype [1 1 10 1 5 5]. These medoids allow recovering the clusters depicted in Fig. 1 by using by the nearest prototype rule as well.

The potential advantages and drawbacks of the *real encoding* schemes are fundamentally the same as the *integer medoid-based encoding* scheme discussed in Section II.A.1.b, with the caveat that the former demands $O(n \cdot k)$ memory space in order to represent a given genotype, whereas the latter demands only $O(k)$ space. Possibly for this reason, the use of a *real medoid-based encoding* scheme has not been reported in any work surveyed in the present paper.

2) Operators: A number of crossover and mutation operators for clustering problems have been investigated. In order to aid the perception of common features shared by these operators, we address them according to the encoding schemes for which they have been designed.

a) *Crossover*: Falkenauer [43] addresses several drawbacks of *traditional* genetic algorithms when they are applied to tackle grouping tasks. As far as crossover operators are concerned, an important problem to be considered involves the *context-insensitivity* concept. Formally, *context-insensitivity* means that [43] "the schemata defined on the genes of the simple chromosome do not convey useful information that could be exploited by the implicit sampling process carried out by a clustering genetic algorithm". In the following we illustrate, by means of pedagogical examples, the context-insensitivity problem. Then, having such examples in mind, we analyze crossover operators frequently described in the literature.

Let us assume that genotypes [1111222233] and [1111333322] – encoded under the *label-based integer encoding* discussed in Section II.A.1.b – are recombined under the standard one-point crossover, as depicted in Fig. 2 (bold type refers to the exchanged genetic information).

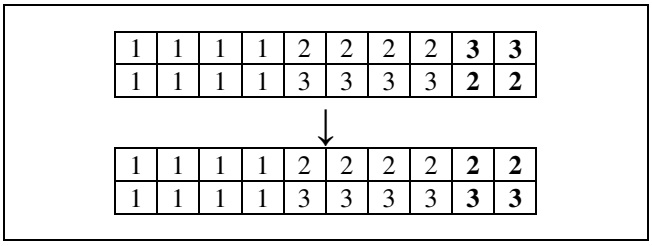

Fig. 2. Example 1 – standard one-point crossover: equal parents generating different offspring.

In this case, the resulting genotypes (offspring) should be equal to their parents, since the parents represent the same solution to the clustering problem – depicted in Fig. 1. However, one can observe that the offspring represent clustering solutions different from the ones encoded into their parents. Moreover, assuming that the number of clusters has been fixed in advance as k=3, invalid solutions formed by two clusters have been derived from the application of crossover. In this particular case, the context-insensitivity problem could have been avoided by using a renumbering procedure before crossover (with the caveat that such a procedure would incorporate an additional computational burden to the algorithm). However, although the use of a renumbering procedure can ameliorate a clustering genetic algorithm based on traditional operators, it does not offer any guarantees that invalid clustering solutions will not be produced by the application of crossover. To illustrate this point, let us apply the standard one-point crossover to the genotypes [1122223333] and [1111111123], as displayed in Fig. 3 (bold type refers to the exchanged genetic information).

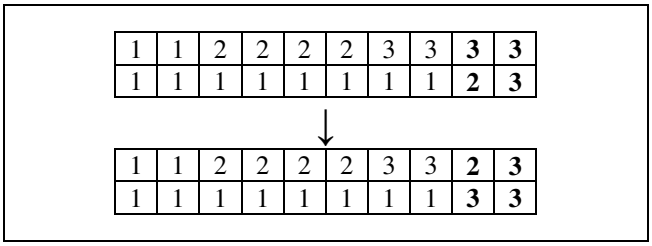

Fig. 3. Example 2 – invalid solution (for fixed k) under standard one-point crossover.

Note that the second child (from top to botton) represents an invalid partition – formed by two clusters. It is not hard to see that similar problems may also occur under the standard two-point crossover.

The application of traditional recombination operators under a *real encoding* scheme may also originate problems similar to those just described. For instance, let us consider the genotypes [1.5 1.5 10.5 1.5 5.0 5.5] and [10.5 1.5 5.0 5.5 1.5 1.5] in which the first two positions represent the two dimensions of the first cluster center, the next two positions represent those of the second cluster center, and so forth. These genotypes represent the same clustering solution (depicted in Fig. 1), the only difference relying on the order in which the clusters are encoded. Accordingly, the offspring resulting from crossing over such genotypes should ideally encode the partition depicted in Fig. 1 as well. However, this may not happen. For the sake of illustration, let us consider that the previous genotypes are recombined under the standard one-point crossover, as illustrated in Fig. 4 (bold type refers to the exchanged genetic information).

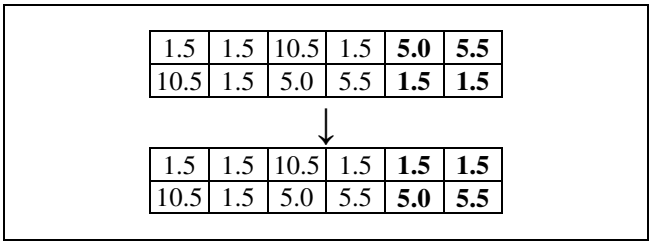

Fig. 4. Example 3 – standard one-point crossover under a real encoding scheme.

In this case, both children represent partitions formed by two clusters. Assuming that the genetic search should ideally provide partitions formed by three clusters (as encoded into their parents), invalid solutions would have been found. Although it is easy to see that similar problems may be faced when the *binary encoding* is employed, context-insensitivity is not so readily detected when using the *integer medoid-based encoding* scheme discussed in Section II.A.1.b, particularly when it is assumed that the k indexes of the selected objects are kept ordered in the array (e.g., by using a renumbering procedure). For illustration purposes, consider the following two medoid-based representation individuals: [1 2 3] and [3 4 5] - each representing three clusters. After doing a single-point crossover between the second and the third genes, the resulting offspring are [1 2 5] and [3 4 3]. This second child has a repeated medoid, thus encoding just two clusters. Context-insensitivity can also conceptually take place under the *integer medoid-based encoding* scheme if the classic definition of a data *set* $\mathbf{X}=\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$ is relaxed, allowing the presence of equal objects, i.e., if it is assumed that $\mathbf{X}$ is a multiset. As an example, let us consider two genotypes, [3 17 25] and [7 13 19], encoding three clusters each. Also, let us consider that objects 13 and 25 in the data set are equal to each other. A single point crossover of these genotypes could produce the offspring [3 17 19] and [7 13 25]. Note that, since objects 13 and 25 are equal to one another, the second child has only two different medoids, which result in a partition with one empty cluster. Of course, empty clusters could be avoided by enforcing the objects closer to multiple identical medoids to be shared among the corresponding clusters. Note, however, that such a procedure is rather unnatural from a conceptual viewpoint, since equal objects are conceptually supposed to belong to the same cluster. It is rather evident that this empty cluster situation becomes more frequent if a *centroid-based real encoding* is used, since different centroids do not necessarily represent different partitions. This situation is not unrealistic for real-world scenarios

and designing procedures to circumvent it is not a trivial task.

To summarize, when traditional genetic operators are employed in clustering problems, they usually just manipulate gene values without taking into account their connections with other genes. It is important to notice that the interconnections among gene values should constitute the genuine optimization goal in clustering problems. For this reason, it is worth giving particular attention to the development of genetic operators specially designed for clustering problems, that is, genetic operators that are cluster-oriented. Nevertheless, a word of caution is in order. Alternatively to the use of cluster-oriented operators, one could claim that there are simple ways of dealing with invalid solutions such as those illustrated in previous examples. Simply putting, invalid solutions found during the evolutionary search could be simply discarded and substituted by valid ones. For instance, the crossover operator could be repeatedly applied until a valid solution has been found, or the invalid child can be set to one of the parents at random. Besides, an invalid solution could be replaced by the best genotype found so far in the evolutionary search. Despite the simplicity involved in these and other related approaches, plenty of computational resources may be wasted to figure out and/or fix invalid solutions. The real impact of such computational burden into the efficiency of the evolutionary search relies on several factors that are hard to theoretically assess, such as the overall design of the evolutionary algorithm and the application in hand. Therefore, it is more conservative not to make any sharp claims concerning the generalization of the efficacy provided by cluster-oriented operators for any algorithm and/or application. Instead, we here focus on conceptual features shared by several operators frequently found in the literature. Aimed at facilitating the visualization of those features, we address them according to the encoding schemes for which they have been designed, as detailed next.

Considering the *string-based binary encoding* described in Section II.A.1.a, Kuncheva and Bezdek [85] adopted a uniform crossover in which the parent genotypes swap their *i*th bits. Under the framework of a variable number of clusters, this operator would be context-sensitive. However, this operator may generate invalid offspring for fixed k, which is a presumed condition of Kuncheva and Bezdek's work, thus being technically context-insensitive in this specific application scenario[5]. Even so, the operator can be deemed cluster-oriented, inasmuch as any exchange of medoids can be interpreted as an exchange of clusters. An object-oriented operator that randomly moves objects among clusters was used in [14].

Several crossover operators based on *integer encoding* have been proposed for evolutionary algorithms. In [84][107], a single-point crossover that is not cluster-oriented (and, accordingly, is context-insensitive) is used. Figures 2 and 3 illustrate potential problems faced by such an operator. The cluster-oriented crossover operator adopted in [96][122] modifies genotypes in such a way that new genotypes are generated by randomly scrambling the medoids encoded into their parents. Flip mutation, in which a medoid is replaced with another randomly chosen medoid, takes place both during and after crossover. Similarly, Estivill-Castro and Murray [39] proposed a context-sensitive crossover operator that is based on exchanging medoids from the parents. The set of medoids encoded into the offspring is built iteratively, adding one medoid at a time, until k different medoids have been represented. In [39][96][122], invalid solutions may be generated if $\mathbf{X}$ is assumed to be a multiset. In other words, the crossover operators adopted in these papers may be context-insensitive depending on the data set, though they are cluster-oriented. Krishna and Murty [83] addressed the use of an evolutionary algorithm for clustering that does not make use of any crossover operator, although the authors call their approach a *genetic algorithm*. Strictly speaking, such algorithm is better categorized as an evolutionary algorithm, in which the k-means algorithm is used as a search operator in lieu of crossover. Lu et al. [95][94] also proposed a related algorithm inspired by the work of Krishna and Murty [83].

Let us now consider operators developed for *real encoding* schemes. In [121], a single-point crossover that allows exchanging the clusters centers (centroids) of a pair of genotypes is used. This operator is not context-sensitive, as already illustrated in Fig. 4, though it can be considered to be cluster-oriented if an exchange of centroids is interpreted as an exchange of clusters. The operator used in [100][10] is also based on exchanging information contained in the centroids. More precisely, for genotypes of length l, where $l = n \cdot k$, a crossover point is randomly drawn within the range $[1, l-1]$. Then, the portions lying to the right of the genotypes under consideration are exchanged to produce offspring. By doing so, not only centroids are exchanged, but also parts of them can be modified, thus making the operator to be neither context-sensitive nor cluster-oriented. For instance, let us consider that the single-point crossover proposed by Bandyopadhyay and Maulik [10][100] is applied to the genotypes displayed in Fig. 5 (bold type refers to the exchanged genetic information).

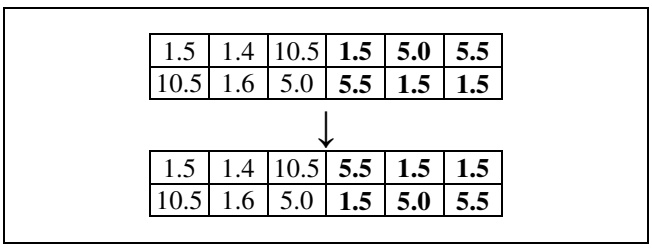

Fig. 5. Example 4 – One-point crossover used by Bandyopadhyay and Maulik [10][100].

[5] Note that an operator may be context-insensitive in a fixed number of clusters scenario yet context-sensitive in a variable number of clusters scenario.

Although the parent genotypes in Fig. 5 are not equal to one another, they represent the same data partition (Fig. 1). In spite of this, the first genotype of the offspring has two "cluster centers" – (1.5, 1.4) and (1.5, 1.5) – positioned in the data density region of *Cluster 1* and one "cluster center" positioned in a region of low data density not represented in the parent-genotypes, i.e., in (10.5, 5.5). Similarly, the second offspring encodes the "cluster center" (5.0, 1.5), which lies in a low density data region not represented in either of the parent genotypes. This example suggests that such a crossover operator may generate clustering solutions significantly different from those encoded into their parents, thus having an evolutionary role more closely related to mutation than to crossover.

Fränti et al. [48] assessed five crossover operators that fundamentally select k centroids from two parents. The *random crossover* operator randomly chooses $k/2$ centroids from each of the two parents. Duplicate centroids are replaced by means of repeated draws. In the operator named *centroid distance*, the clusters are initially sorted according to their distances from the grand mean (overall centroid) of the data set. Then, they are divided into two subsets, namely: *central clusters* and *remote clusters*. The *central clusters* are those closer to the centroid of the data set, whereas the *remote clusters* are the remaining ones. An offspring is created by taking the *central clusters* from parent *A* and the *remote clusters* from parent *B*. In *pairwise crossover*, clusters encoded into different parents are paired according to the similarities of their centroids. An offspring is then generated by randomly taking one centroid from each pair of clusters. In the *largest partitions* operator, M centroids are selected by a greedy heuristic based on the assumption that larger clusters are more important than smaller ones. Finally, the authors evaluate the *pairwise nearest neighbor* crossover operator that considers that the $2k$ centroids from parents *A* and *B* can be clustered into k clusters that will form the offspring. The crossover operators just described have been designed to manipulate clusters, thus being cluster-oriented. Nonetheless, the *centroid distance* operator may be viewed as a variant of the single point crossover, which is not context-sensitive. Based upon an experimental evaluation, the authors argue that the *pairwise nearest neighbor* operator is the best choice among the assessed variants. Kivijärvi et al. [79] have used the same crossover operators described by Fränti et al. [48], but these authors also employed an additional single point crossover operator. Merz and Zell [103] used two recombination operators. The first one is uniform crossover, which randomly copies centroids from the parents. The second crossover operator basically replaces some centroids in parent *A* with centroids from parent *B* using the nearest neighbor concept. Even though they are cluster-oriented, these operators are affected by context-insensitivity problems similar to those already discussed.

b) Mutation: Following Falkenauer´s typology [43], mutation operators can be categorized as being *object-oriented* or *group-oriented*. The latter class is particularly interesting for clustering problems, since it encompasses operators designed to work with clusters rather than with objects. Kuncheva and Bezdek [85] mutate genotypes by alternating bits of the *string-based binary encoding*. This corresponds to either deleting existing prototypes or inserting new prototypes. Since each prototype allows recovering a cluster, the operator is conceptually cluster-oriented, with the caveat that invalid solutions may be generated when k is fixed *a priori*. The object-oriented mutation operator described in [14] randomly assigns an object to a different cluster. Considering *integer encoding*, Murthy and Chowdhury [107] used a mutation operator that randomly changes the gene value (cluster label) of some randomly selected objects. This object-oriented operator may generate invalid solutions when the number of clusters is fixed. Krishna and Murty [83] applied a mutation operator that changes a gene value depending on the distances of the cluster centroids from the corresponding object. In particular, the probability of changing a gene value to a given cluster label is higher if the centroid of the corresponding cluster is closer to the object. This object-oriented mutation can be considered a guided operator, but it may yield empty clusters. Lu et al. [95][94] adopted an object-oriented mutation operator similar to the one developed by Krishna and Murty [83]. The genetic algorithm developed by Lucasius et al. [96] randomly selects a medoid that can be replaced with an object from the data set – according to a predetermined probability. Similar approaches are adopted in [122][39]. The mutation approaches described in [96][122][39] can be considered as cluster-oriented inasmuch as medoids allow recovering clusters. The application of such approaches, however, may result in invalid solutions if $\mathbf{X}$ is assumed to be a multiset and k is fixed.

For *real encoding* schemes, some papers [10][100][121] describe mutation operators aimed at slightly modifying the centroids encoded into a given genotype. As a consequence, these mutations tend to change the membership of some objects in relation to the clusters represented by the genotype. In other words, the underlying philosophy of these operators is not constrained to create new clusters or to eliminate existing ones. Hence, conceptually speaking, such operators are better categorized as object-oriented, for it is expected that the result of the mutation will be changing the membership of a subset of objects from some particular clusters. More precisely, Scheunders [121] proposed to randomly add a value equal to either -1 or +1 to a randomly chosen component of the centroid of a given cluster. Maulik and Bandyopadhyay [100], in turn, proposed to mutate clusters centers by the following procedure: a number δ in the range [0, 1] is drawn with uniform distribution. This number is then used to change the value v of a given gene to $(1 \pm 2\delta)v$ when $v \neq 0$, and $\pm 2\delta$ when $v=0$. Signs "+" and "–" occur with equal probability. In [10], the authors adopted a

conceptually similar mutation scheme. However, differently from the approach proposed in [100], the mutation operator in [10] provides perturbation in a maximum range to genotypes either when they are less fitted in the population or when all the genotypes have the same fitness value. Alternatively to slightly changing cluster prototypes by some *perturbation* in their component values, some papers [48][79][103] describe mutation operators that replace prototypes with objects from the data set. Such operators are more likely to create or eliminate clusters. From this point of view, they can be considered as cluster-oriented. More specifically, in the work by Fränti et al. [48] a prototype is replaced with a randomly chosen object from the data set. The same mutation operator is used by Kivijärvi et al. [79]. Two mutation operators are used by Merz and Zell [103]. The first operator assigns a selected object from the data set to substitute a randomly chosen prototype. The second operator randomly selects two clusters $\mathbf{C}_i$ and $\mathbf{C}_j$. Next, the object belonging to $\mathbf{C}_i$ with the maximum distance from its prototype is chosen to replace the prototype of $\mathbf{C}_j$, such that this can be considered a guided operator. Krovi [84] does not report the mutation operator used.

3) Fitness Function: Many clustering validity criteria can be used for assessing partitions containing a given number (k) of clusters (e.g., see [72], [40], [75]). Several of these criteria can be adapted to be used as fitness functions to evolve data partitions. Thus, the fitness functions used by the evolutionary algorithms described in the literature, and here surveyed, represent only a subset of the possible fitness functions that can be used in practice.

Krovi [84] proposes a fitness function to assess partitions formed by only two clusters. This function is based both on the average distance from the objects to their respective cluster centroids and on the distance between cluster centroids. In particular, the ratio of the distance between centroids and average intra-cluster distances is used to evaluate the fitness of a genotype.

Lucasius et al. [96] suggest minimizing the distances from the k medoids encoded into the genotype to the objects of the corresponding clusters. More precisely, the authors propose to minimize the sum of distances between the objects of the data set $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ and the medoids from the set $\{\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_k\} \subset \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$. To this end, they define the criterion F:

$$F = \sum_{i=1}^{N} d(\mathbf{x}_i, \mathbf{m}) \qquad (1)$$

where $\mathbf{m}$ represents the closest medoid to object $\mathbf{x}_i$, i.e., $d(\mathbf{x}_i,\mathbf{m}) = \min_{j\in\{1,\ldots,k\}} d(\mathbf{x}_i,\mathbf{m}_j)$, and d denotes a distance measure. This criterion is essentially the well-known sum of within-cluster distances applied to a *medoid-based representation.* The genetic algorithms developed by Estivill-Castro and Murray [39] and Sheng and Liu [122] are also aimed at minimizing F in Equation (1).

Some authors (Murthy and Chowdhury [107]; Maulik and Bandyopadhyay [100]; Bandyopadhyay and Maulik [10]; Merz and Zell [103]) propose to minimize the sum of squared Euclidean distances of the objects from their respective cluster means. Formally, the fitness function $f(\mathbf{C}_1,\mathbf{C}_2,\ldots,\mathbf{C}_k)$ adopted by these authors is:

$$f(\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_k) = \sum_{j=1}^{k} \sum_{\mathbf{x}_i \in \mathbf{C}_j} \left\| \mathbf{x}_i - \mathbf{z}_j \right\|^2 \qquad (2)$$

where $\{\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_k\}$ is the set of k clusters encoded into the genotype, $\mathbf{x}_i$ is a data set object, and $\mathbf{z}_j$ is the mean vector of cluster $\mathbf{C}_j$. This criterion is essentially a *centroid-based* version of (1). The fitness functions used in the genetic algorithms described by Fränti et al. [48], Kivijärvi et al. [79], Krishna and Murty [83], and Lu et al. [95][94] are aimed at minimizing some measure of the *distortion* of the clusters, which is equivalent to minimize $f(\mathbf{C}_1,\mathbf{C}_2,\ldots,\mathbf{C}_k)$ in (2). For instance, the distortion d can be a measure of the intra-cluster diversity defined as:

$$d = \frac{f(\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_k)}{N \cdot n} \qquad (3)$$

where N and n are the numbers of objects and attributes of the data set, respectively.

The fitness function adopted by Scheunders [121] also relates to $f(\mathbf{C}_1,\mathbf{C}_2,\ldots,\mathbf{C}_k)$ defined in (2), as follows:

$$f'(\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_k) = \frac{N}{f(\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_k)} \qquad (4)$$

Kuncheva and Bezdek [85] use a genetic algorithm for minimizing the well-known J_m criterion:

$$J_m = \sum_{i=1}^{k} \sum_{j=1}^{N} \mu_{ij}^m \left\| \mathbf{x}_j - \mathbf{v}_i \right\|^2 \qquad (5)$$

where $\mathbf{x}_j$ is the jth data set object, $\mathbf{v}_i$ is the prototype of the ith cluster $\mathbf{C}_i$, μ_{ij} denotes the membership of object $\mathbf{x}_j$ to cluster $\mathbf{C}_i$, and m ($m \geq 1$) is a user-defined parameter. Clearly, J_m reduces to the sum of within-cluster distances in (2) when dealing with hard partitions, in which case μ_{ij} is such that $\mu_{ij} \in \{0, 1\}$. A fitness function based on J_m – equation (5) – has also been used in [14].

It is important to stress that these previous criteria all make sense if and only if the number of clusters k is fixed, since, for fixed k, minimizing the intra-cluster distances implies maximizing the inter-cluster distances as well. However, that does not hold when k is variable. Indeed, it is straightforward to see that one can arbitrarily minimize the sum of intra-cluster distances by increasing the number k of clusters, making it equal to zero in the limit by assigning each object to an individual cluster ($k = N$ singletons).

In general, the previous projects used fitness functions based on the distance between objects and either clusters' centroids or medoids. Although these types of functions are still widely used, they usually have some clear disadvantages. E.g., trying to minimize the Euclidean distance between objects and its nearest cluster's centroid usually is biased towards the discovery of spherical clusters, which clearly will be inappropriate in many applications where the natural clusters for the data are not spherical. In other words, many clustering algorithms impose a specific type of structure (like

spherical clusters) to the data, rather than discovering clusters of arbitrary shape in a more data-driven way.

In the last decade or so, an alternative to such distance-based clustering validity measures, consisting of density-based clustering validity criteria, has been increasingly used in data mining. In such criteria, a cluster is essentially a group of objects in the same dense region in the data space, and the goal of a density-based clustering algorithm is to find high-density regions (each region corresponding to a cluster) that are separated by low-density regions. Density-based clustering methods usually have the advantage of being flexible enough to discover clusters of arbitrary shape [38].

An evolutionary algorithm – more precisely, an Estimation of Distribution Algorithm – using a density-based fitness function is described in [31]. In this algorithm, the fitness function is essentially computed as the average density over the clusters represented by an individual, where the density of a cluster is simply the number of objects in the cluster divided by the size of the region defining that cluster.

4) Selection: Proportional selection [6] has been used by several authors (e.g., Krovi [84]; Lucasius et al. [96]; Murthy and Chowdhury [107]; Estivill-Castro and Murray [39]; Fränti et al. [48]; Maulik and Bandyopadhyay [100]; Kivijärvi et al. [79]; Bandyopadhyay and Maulik [10]; Krishna and Murty [83]; Lu et al. [95][94]). Additionally to proportional selection, *elitist* variants [105] for selecting genotypes are also investigated in the papers by Murthy and Chowdhury [107], Fränti et al.[48], and Kivijärvi et al. [79].

Not much is said about the selection procedure used by Scheunders [121]. The author only reports that all genotypes are pairwise compared and the most fitted genotype of each pair is copied into the other. This description suggests that Scheunders [121] uses a variant of the so-called deterministic tournament selection [6]. Sheng and Liu [122] use two-fold tournament selection to choose $P/2$ parent pairs for reproduction, where P is the population size.

Kuncheva and Bezdek [85] adopt an elitist strategy in which the parents and the children are pooled and the best genotypes survive, standing as the new parents. Similarly, Merz and Zell [103] derive a new population by selecting the best genotypes out of the pool of parents and children. These selection methods can be viewed as variants of the so-called $(\mu+\lambda)$ selection procedure used in *evolution strategies* [6][30].

The advantages and disadvantages of the existing selection mechanisms are well-known in the evolutionary computation literature. As far as we know, there is no strong evidence that the relative behavior of these mechanisms is much different when assessed in the particular context of evolutionary clustering.

5) Initial Population: In the papers by Krovi [84], Murthy and Chowdhury [107], Krishna and Murty [83], Bezdek et al. [14], and Lu et al. [95][94], the initial population of the genetic algorithm is generated from random assignments of objects to clusters. Such an initialization strategy usually results in unfavorable initial partitions, since the initial clusters are likely to be mixed up to a high degree. It constitutes, however, an effective approach to test the algorithms against hard evaluation scenarios.

Lucasius et al. [96] suggest randomly selecting a subset of objects to be the medoids of the initial population, when prior knowledge is not available. Similarly, in the papers by Kuncheva and Bezdek [85], Estivill-Castro and Murray [39], Maulik and Bandyopadhyay [100], Merz and Zell [103], Sheng and Liu [122], and Bandyopadhyay and Maulik [10], an initialization strategy is adopted that randomly chooses data set objects to be initial prototypes of the clusters. In the papers by Scheunders [121], Fränti et al. [48], and Kivijärvi et al. [79], the initial prototypes of the clusters are also randomly generated, but not restricted to the data positions.

The random initialization of prototypes represents a good trade-off between simplicity, computational demand, and effectiveness. For this reason, this sort of strategy is very popular in practice, both in evolutionary and non-evolutionary clustering applications.

6) Local Search by k-means: Algorithms endowed with mechanisms that both globally explore and locally exploit the search space are well-known under the heading of hybrid or memetic evolutionary algorithms. The combination of evolutionary search with the k-means algorithm has led to a variety of hybrid evolutionary algorithms for clustering that can be considered to belong to this class. For instance, in the approach by Krishna and Murty [83], a one step k-means operator is used in substitution of crossover. The authors suggest that the resulting algorithm can avoid expensive computations required by the usual crossover operators. A similar approach is followed by Lu et al. [95].

Kivijärvi et al. [79], Fränti et al. [48], and Badyopadhyay and Maulik [10] apply the k-means algorithm for fine-tuning partitions found by genetic operators. Scheunders [121] applies k-means to all genotypes of each generation, aiming at leading the corresponding partitions into local optima. Sheng and Liu [122] adapted k-means to work as a local search heuristic for the problem of finding k medoids.

In contrast to hybrid evolutionary algorithms that employ k-means as an additional operator for a fine-tuning of the evolving partitions, the memetic algorithm proposed by Merz and Zell [103] has been designed for searching in the space of locally optimal solutions - instead of searching in the space of all candidate solutions.

B. Algorithms with Variable Number of Clusters

The evolutionary algorithms addressed in Section II.A were designed to optimize partitions for fixed (user-defined) values of k. As discussed in that section, evolutionary algorithms that search for k clusters (k defined beforehand) can be particularly suitable for applications in which there is information available (e.g., domain knowledge) regarding k. The evolutionary algorithms for clustering described in this section go even further by searching for both the *best* number of clusters (k^*) and their corresponding partitions in regions of the space where they are more likely to be found. These algorithms have been designed with the underlying assumption that the *best* number of clusters, k^*, is unknown *a priori*. In other words, it is presumed that the number of clusters is inherent to the data set and that estimates for k^* are not

available. One may hypothesize that evolutionary algorithms for fixed k (addressed in Section II.A) could be potentially used for tackling such a problem. In order to do so, repeated runs of the evolutionary algorithm might be performed for different values of k, and the obtained clustering solutions could be comparatively assessed by some measure that reflects the partition quality[6]. This approach may eventually provide good results in applications for which the cardinality of the set formed by the possible values of k is low. In practice, this situation may take place when there is a strong suspect that a small interval of values contains k^*. In other situations, clustering algorithms that explore partitions with different numbers of clusters are often preferred.

Evolutionary algorithms aimed at optimizing the number of clusters (k) and the corresponding partitions are described in the works by Cole [23], Cowgill et al. [26], Bandyopadhyay and Maulik [12][11], Hruschka and Ebecken [65], Casillas et al. [21], Hruschka et al. [69][70][68], Ma et al. [97], Alves et al. [2], Tseng and Yang [128], Naldi and de Carvalho [108], Handl and Knowles [59], and Pan and Cheng [113]. Falkenauer [43] describes a high-level paradigm (metaheuristic) that can be adapted to deal with grouping problems broadly defined, showing that it is useful for several applications – e.g., bin packing, economies of scale, conceptual clustering, and equal piles. Data partitioning problems like those examined in this paper are not the primary focus of Falkenauer's book [43]. Nevertheless, it is worth mentioning that, in order to pave the way for the proposed paradigm, the author investigates, among other issues, key aspects of some genetic algorithms designed for data partitioning problems until 1998. Most importantly, the concepts underlying such a paradigm allow delving into important features of evolutionary algorithms for data partitioning problems. Therefore, Falkenauer's work permeates several discussions hereafter performed.

1) Encoding Schemes: Most of the encoding schemes used in evolutionary algorithms capable of estimating the number of clusters (k) are similar to the encoding schemes employed in evolutionary algorithms for which k is assumed to be known or set a priori. Thus, in this section, we complement the material already explored in Section II.A.1, trying to avoid unnecessary redundancies as much as possible.

Cole [23] adopts the *label-based integer encoding* described in Section II.A.1.b, in which a genotype is an integer vector of N positions, each of which is associated with a data object and takes a value (cluster label) over the alphabet $\{1,2,3,\ldots,k\}$ In this case, k can be interpreted as the maximum number of clusters represented in each individual. This encoding scheme has also been used by Cowgill et al. [26], Hruschka and Ebecken [65], Hruschka et al. [69][70][68], Naldi and de Carvalho [108], and Alves et al. [2], but some of these authors additionally suggest storing the number of clusters (k) in the genotype. In this case, k represents the fixed number of clusters of the individual, but different individuals can have different values of k, so that the population as a whole represents candidate solutions with different numbers of clusters. A discussion on the potential advantages and drawbacks of such an *integer encoding* scheme can be found in Section II.A.1.b.

Ma et al. [97] proposed an evolutionary algorithm for clustering, named EvoCluster, which encodes a partition in such a way that each gene represents one *cluster* and contains the labels of the objects grouped into it. Thus, a genotype encoding k clusters ($\mathbf{C}_1$, $\mathbf{C}_2$, …, $\mathbf{C}_k$) of a data set with N objects is formed by k genes, each of which stores n_i labels ($n_1 + n_2 + \ldots + n_k = N$). Ma et al. [97] claim that this encoding scheme represents an advantageous alternative over other different approaches. In particular, they argue that the *label-based integer encoding* is not very scalable, since the length of each genotype is exactly the number of objects of the data set. Although this assertion is persuasive at a first glance, it is worth noticing that the amount of information that must be stored (and handled) in both encoding schemes described above is essentially the same, that is, N object labels (EvoCluster's encoding) or N cluster labels (*label-based* encoding). Then, the scalability of EvoCluster in terms of memory requirement does not really benefit from its encoding scheme. Actually, the encoding scheme does not seem to be a crucial aspect regarding the practical usefulness of an algorithm when handling large data sets. In the end, the data set itself must be handled somehow (e.g. using efficient data structures for external memory management) and its dimensionality is necessarily larger than that of any encoding scheme.

A very different kind of integer encoding is used in [119][120]. In this work an individual represents a set of axis-aligned hyper-rectangular rules, each rule consisting of n genes – where n is the number of attributes in the data being mined. In each rule, the ith gene, $i = 1,\ldots,n$, encodes two fields, namely, the lower (l_i) and upper (u_i) bounds defining the boundary of the rule in the ith dimension (attribute). For instance, in a 2-dimensional problem, a 2-rule individual would look like:

Rule 1 (Cluster 1)				**Rule 2 (Cluster 2)**			
2	5	3	9	1	6	2	4
l_1	u_1	l_2	u_2	l_1	u_1	l_2	u_2

The integer numbers encoded in the fields l_i and u_i represent indices of intervals produced during a previous quantization stage (before the evolutionary algorithm is run). Hence, this encoding is both integer-based and grid-based. Note that there is an implicit conjunction operator linking the boundaries of all the dimensions of a rule. Hence, in this particular example, the first rule can be read as:

IF (interval_2 ≤ attribute_1 ≤ interval_5)
AND (interval_3 ≤ attribute_2 ≤ interval_9)
THEN (object belongs to Cluster 1) .

One advantage of this kind of representation is that axis-aligned hyper-rectangular rules can usually be easily interpreted by the user. The algorithm ensures that the rules represented in an individual are disjoint, which arguably

[6] Indexes for measuring quality of data partitions with variable k will be further discussed in Section II.B.3.

further facilitates the rules' interpretation. The grid-based representation also helps to reduce the size of the search space – by comparison with a real-valued representation based on original (not quantized) attribute values. In addition, the quantization is performed by an elaborated statistical procedure, which aggregates together objects mapping into the same grid cell. Note that, although each rule has a fixed length (d genes), different individuals can contain different number of rules, so that the algorithm considers a variable number of clusters along its search.

In the work by Bandyopadhyay and Maulik [12], genotypes are made up of real numbers that represent the coordinates of the cluster centers. If genotype i encodes k clusters in n dimensional space $\Re^n$, then its length is $n \cdot k$. This encoding scheme has been examined in Section II.A.1.c under the term *real encoding*. An important difference in the present context is that this encoding scheme leads to variable length genotypes, once k is no longer assumed to be constant. In [11], the authors use a slightly different encoding scheme that allows working with constant length genotypes. Basically, the number of clusters encoded by a given genotype is assumed to lie in the range $[k_{min}, k_{max}]$ – where k_{min} and k_{max} are the minimum and maximum number of clusters, respectively – and a "don't care" symbol (#) is used to fill in genotypes whose k is less than k_{max}. An evident disadvantage of this scheme is that it demands estimates of k_{min} and k_{max}. On the one hand, a too wide interval $[k_{min}, k_{max}]$ causes a waste of memory and processing time. On the other hand, a too narrow interval increases the probability of leaving k^* out, thus becoming unreachable. Tseng and Yang [128] propose to generate a set of m small clusters to be used as building blocks for the genetic search. Such initial clusters are obtained using a nearest neighbor based strategy. Partitions are encoded into m-length binary strings (genotypes) that represent subsets of the clusters initially generated. In particular, if cluster i is encoded into a given genotype, then the value of the ith position of the corresponding string (the ith gene) will be 1; otherwise, it will be 0. This is a particular kind of *binary encoding* that has the disadvantage of limiting the genetic search to combinations of the building blocks initially generated. Pan and Cheng [113] also adopt a binary encoding scheme based on a maximum number of clusters that is determined *a priori*. Each position of the string corresponds to an initial cluster center. Thus, the total number of 1s in the string corresponds to the number of clusters encoded into the genotype.

Casillas et al. [21] adopt as encoding scheme a binary vector with $(N-1)$ elements. These elements represent the $(N-1)$ edges of a Minimum Spanning Tree (MST) [24] in which the nodes represent the N data set objects and the edges correspond to proximity indexes between objects. In this representation, the value 0 means that the corresponding edge remains, whereas the value 1 means that the corresponding edge is eliminated. The number of elements with value 1 is equal to $(k-1)$, where k is the number of clusters. This is an example of a *tree-based representation*. Specifically, it corresponds to a *tree-based binary encoding* scheme. A disadvantage of this scheme is that it demands the highly intensive computation of an MST for a complete graph with N vertices, which may become prohibitive for large N.

Handl and Knowles [59] employ a *graph-based representation* in which a genotype is an integer vector of N positions, that is, a *graph-based integer encoding* scheme. Each position of the genotype corresponds to an object, i.e., the ith position (gene) represents the ith data set object. Genes can take values from the set $\{1, 2, \ldots, N\}$. A value j assigned to a gene i means that there is a link between objects i and j and that these are placed into the same cluster. The partition encoded into the genotype is recovered by identifying all connected components of the graph. This encoding scheme is particularly suitable in the context of the evolutionary algorithm for multi-objective clustering proposed by the authors.

2) Operators: Several crossover and mutation operators have been proposed for clustering problems in which the number of clusters is unknown in advance. In this section, we elaborate on the main properties of crossover and mutation operators commonly found in the corresponding literature. Analogously to what we have done in Section II.A.2, we discriminate the operators according to the encoding schemes for which they have been designed.

a) Crossover: Considering *integer encoding*, Cole [23] uses edge-based crossover operators. Two objects are considered to be connected by an edge if they are in the same cluster. These crossover operators construct children by combining the edges of their parents, considering the set of intersections of the clusters. They manipulate clusters encoded in the parent genotypes, in such a way that context-sensitivity is kept. Cowgill et al. [26] adopt uniform crossover in the early generations and two-point crossover in later generations. These operators are not cluster-oriented (see Section II.A.2.a). The context-sensitive crossover operator proposed in [65] is inspired by Falkenauer's work [43]. It combines clustering solutions coming from different genotypes. More precisely, the operator works in the following way. First, two genotypes ($\mathbf{G_1}$ and $\mathbf{G_2}$) are selected. Then, assuming that $\mathbf{G_1}$ represents k_1 clusters, $c \in \{1, 2, \ldots, k_1\}$ clusters randomly chosen are copied into $\mathbf{G_2}$. The unchanged clusters of $\mathbf{G_2}$ are maintained and the changed ones have their unaffected objects allocated to the corresponding nearest clusters (according to their centroids). In this way, an offspring $\mathbf{G_3}$ is obtained. The same procedure is employed to get an offspring $\mathbf{G_4}$, but now considering that the changed clusters of $\mathbf{G_2}$ are copied into $\mathbf{G_1}$. This crossover operator is also used in the genetic algorithms for clustering described in [69][70][68]. A similar crossover operator is also used by the EvoCluster algorithm [97]. Such a crossover operator, however, can be probabilistically guided by information concerning the quality of the individual clusters in a given partition.

Tseng and Yang [128] use two-point crossover for their *binary encoding* scheme based on building blocks,

discussed in Section II.B.1. In such an encoding scheme, a two-point crossover exchanges sets of clusters (i.e., it is cluster-oriented) and can be context-sensitive in a variable number of clusters scenario. Similarly, Pan and Cheng [113] adopt a one-point crossover that manipulates cluster centers.

Bandyopadhyay and Maulik [12] use a two-point crossover that allows exchanging real-valued cluster prototypes from a pair of genotypes. In [11], a single-point crossover is used. As discussed in Section II.A.2.a, single-point and two-point crossover operators are not context-sensitive when applied to genotypes under *real encoding*.

Casillas et al. [21] use a one-point crossover that manipulates the edges of a Minimum Spanning Tree (MST), in which the nodes represent the data set objects and the edges correspond to proximity indexes between them. The adopted operator can split and merge clusters and it is context-sensitive.

Handl and Knowles [59] use the standard uniform crossover operator. Under the *graph-based representation* employed by the authors, uniform crossover implements merging and splitting operations on individual clusters, while maintaining the remainder of the partition, thus being cluster-oriented.

The evolutionary algorithms for clustering presented in [70][68][2] do not make use of crossover operators.

b) *Mutation*: Cole [23] uses three mutation operators designed for *integer encoding*. The *split* cluster-oriented operator probabilistically selects a cluster from a particular partition and moves objects from that cluster into a new cluster. The *merge* cluster-oriented operator moves all the objects from one cluster to another pre-existing cluster. Finally, the *move* object-oriented operator shifts objects between clusters already encoded on a genotype. These operators are inspired by Falkenauer's work [43]. In the algorithm described in [26], the mutation process is applied to each genotype resulting from the application of the crossover operator. Elements of each genotype are randomly altered according to low probabilities. A single mutation randomly changes the gene value (cluster label) of a randomly selected object, thus being object-oriented. Two cluster-oriented operators for mutation inspired by Falkenauer's work [43] are used in [65][69]. The first operator works only on genotypes that encode more than two clusters. It eliminates a randomly chosen cluster, placing its objects into the nearest remaining clusters (according to their centroids). The second mutation operator splits a randomly selected cluster, which must be formed by at least two objects to be eligible for this operator, into two new ones. The first cluster is formed by the objects closer to the original centroid, whereas the other cluster is formed by those objects closer to the farthest object from the centroid. Similar mutation operators are adopted in [70][68]. The main difference from the approach used in [65][69] is that the slightly modified mutation operators described in [70][68] are allowed to act on more than one cluster encoded into the genotype. The EvoCluster algorithm [97] has six mutation operators that also split, merge, and eliminate groups. These operators can be viewed essentially as modified versions of those in [65][69][70][68]. Differently from the operators found in [65][69] and analogously to their modified versions in [70][68], EvoCluster's mutation operators can be simultaneously applied to multiple clusters of the same partition. In addition, they can be probabilistically guided by information concerning the quality of the individual clusters in a given partition (*guided operators*). In fact, the guided application of evolutionary operators has been shown – from a statistical perspective – to be able to significantly speed up convergence of the evolutionary search for clustering [2][7]. The same holds with respect to the simultaneous application of the mutation operators to multiple clusters of the same partition, as shown in [70][68]. The guided mutation operators described in [2] go even further by including an additional mechanism that also helps improving mutation performance. It consists of the use of a self-adjusting procedure that automatically controls the rates of application of the individual mutation operators based upon their relative success/failure averaged over past generations.

Bandyopadhyay and Maulik [12][11] propose to mutate clusters centers by the following procedure: A number δ in the range [0,1] is generated with uniform distribution. This number is then used to change the value v of a given gene to $(1 \pm 2\delta)v$ when $v \neq 0$, and $\pm 2\delta$ when $v=0$. Signs "+" and "–" occur with equal probability. Since the genes encode coordinates of cluster prototypes, these operators are conceptually neither cluster-oriented nor object-oriented.

In [128], loci of the binary genotype are chosen according to a given probability and their values are changed either from 0 to 1 or vice-versa. Conceptually speaking, the initial building clusters encoded into the genotypes may be inserted or not inserted into the offspring, thus making the mutation operator be cluster-oriented. Analogously, the number of clusters and, consequently, the cluster centers, are randomly changed by the mutation operator reported in [113].

Casillas et al. [21] change bits using a low mutation probability. Under the representation adopted, this operator can split and merge clusters, thus being cluster-oriented.

Handl and Knowles [59] use a nearest-neighbor based

[7] In [2], however, all the operators are guided in the sense that bad clusters are more likely to be modified, whereas in Evocluster [97] the crossover operator is more likely to affect good clusters.

mutation operator. In particular, each object can be probabilistically linked to one of its L nearest neighbors. Such a procedure may affect clusters, objects only, or neither. For this reason, this mutation operator cannot be conceptually and strictly categorized as cluster-oriented or object-oriented.

3) Fitness Function: In principle, any relative clustering validity criterion (e.g. see Jain and Dubes [72]; Milligan and Cooper [104]; Halkidi et al. [55]; Handl et al. [60]) that is non-monotonic with the number of clusters can be potentially used as a fitness function for an evolutionary algorithm designed to optimize the number of clusters. Such criteria have been extensively investigated and, despite the well-known fact that their particular features make their performance problem dependent [112], some of them have shown satisfactory results in several different application scenarios. In the sequel, a number of relative validity criteria that have been used as fitness functions for evolutionary clustering algorithms are reviewed.

Cole [23], Cowgill et al. [26], and Casillas et al. [21] use as fitness function the Calinski and Harabasz Variance Ratio Criterion (VRC) [18], which is defined as:

$$VRC = \frac{\text{trace } \mathbf{B}/(k-1)}{\text{trace } \mathbf{W}/(N-k)} \tag{6}$$

where **B** and **W** are the between-cluster and the pooled within-cluster sums of squares (covariance) matrices, respectively. The terms N and k are the total number of objects and the number of clusters in the partition, respectively.

Tseng and Yang [128] define the fitness function of a given genotype G as:

$$Fitness(G) = \sum_{i=1}^{k} D_{\text{inter}}(C_i) \cdot w - D_{\text{intra}}(C_i) \tag{7}$$

where $D_{intra}(C_i)$ is the intra-distance of cluster C_i, $D_{inter}(C_i)$ is the inter-distance between C_i and the set of all other clusters, and w is a user-defined parameter.

Bandyopadhyay and Maulik [12] propose the following validity index $I(k)$ for computing the fitness of a genotype that represents k clusters:

$$I(k) = \left(\frac{1}{k} \cdot \frac{E_1}{E_j} \cdot D_j \right)^p \tag{8}$$

where p is any real number larger than or equal to 1. Terms E_j and D_j are given by equations (9) and (10), respectively:

$$E_j = \sum_{j=1}^{k} \sum_{i=1}^{N} \mu_{ji} \left\| \mathbf{x}_i - \mathbf{z}_j \right\| \tag{9}$$

$$D_j = \max_{l,m=1}^{k} \left\| \mathbf{z}_l - \mathbf{z}_m \right\| \tag{10}$$

where N is the total number of objects in the data set, $[\mu_{ji}]_{k \times N}$ is a partition matrix for the data set $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, and $\mathbf{z}_m$ is the center of the mth cluster. The authors report some experiments in which $I(k)$ provides better results than the Davis-Bouldin [28] and Dunn's [36] indexes commonly used as relative validity criteria for clustering. Nevertheless, in a later work [11], the authors turn to use a fitness function based on the Davis-Bouldin (DB) index. A variant of the $I(k)$ defined in (8) is also adopted as the fitness function in [113]. Besides using the Calinski and Harabasz's VRC [18] defined in (6), Cole [23] also evaluates the minimization of the DB index as a fitness function. The DB index for the partitioning of N objects into k clusters is defined as [28]:

$$DB = \frac{1}{k} \sum_{c=1}^{k} R_c \tag{11}$$

in which the index for the cth cluster, R_c, is given by:

$$R_C = \max_{j \neq c} \{R_{j,c}\} \tag{12}$$

with $R_{j,c}$ denoting the measure of within-to-between cluster spread for all pairs of clusters (j, c), that is:

$$R_{j,c} = \frac{e_j + e_c}{m_{j,c}} \tag{13}$$

where e_j is the within cluster variation for the jth cluster and $m_{j,c}$ is the distance between the centers of the jth and cth clusters.

In [65][69], the silhouette proposed by Kaufman and Rousseeuw [75] is employed for computing the fitness of a given genotype. In order to define the silhouette, let us consider an object $\mathbf{x}_i$ belonging to cluster **A**. So, the average dissimilarity of $\mathbf{x}_i$ to all other objects of **A** is denoted by $a(\mathbf{x}_i)$. Now let us take into account cluster **C**. The average dissimilarity of $\mathbf{x}_i$ to all objects of **C** will be named $d(\mathbf{x}_i,\mathbf{C})$. After computing $d(\mathbf{x}_i,\mathbf{C})$ for all clusters $\mathbf{C} \neq \mathbf{A}$, the smallest one is selected, i.e., $b(\mathbf{x}_i) = \min d(\mathbf{x}_i,\mathbf{C})$, $\mathbf{C} \neq \mathbf{A}$. This value represents the dissimilarity of $\mathbf{x}_i$ to its neighbor cluster, and the silhouette $s(\mathbf{x}_i)$ is given by:

$$s(\mathbf{x}_i) = \frac{b(\mathbf{x}_i) - a(\mathbf{x}_i)}{\max\{a(\mathbf{x}_i), b(\mathbf{x}_i)\}} \tag{14}$$

It is easy to verify that $-1 \leq s(\mathbf{x}_i) \leq 1$. Thus, the higher $s(\mathbf{x}_i)$, the better the assignment of object $\mathbf{x}_i$ to a given cluster. In addition, if $s(\mathbf{x}_i)$ is equal to 0, then it is not clear whether the object should have been assigned to its current cluster or to a neighboring one [40]. Finally, if cluster **A** is a singleton, then $s(\mathbf{x}_i)$ is not defined and the most neutral choice is to set $s(\mathbf{x}_i) = 0$ [75]. The silhouette criterion is given by the average of $s(\mathbf{x}_i)$ over $i = 1,2,\ldots,N$. Besides assessing the silhouette developed by Kaufman and Rousseeuw [75] as a fitness function, Hruschka et al. [69][70][68] also proposed additional validity indexes to guide the genetic search. One of them is a simplified version of the silhouette [69][70][68], which is also used by Alves et al. [2]. This criterion is based on the computation of distances between objects and the mean vectors of the clusters. More specifically, the term $a(\mathbf{x}_i)$ of Equation (14) becomes the dissimilarity of object $\mathbf{x}_i$ to the centroid of its cluster (**A**). Similarly, instead of computing $d(\mathbf{x}_i,\mathbf{C})$ as the average dissimilarity of $\mathbf{x}_i$ to all objects of **C**, $\mathbf{C} \neq \mathbf{A}$, only the distance between $\mathbf{x}_i$ and the centroid of **C** must be computed, thus reducing the computational complexity of the index from $O(N^2)$ to $O(N)$. Alternatively to the original and simplified versions of the silhouette, Hruschka et al. [69] showed that the fitness function can also be taken as the average of $b(\mathbf{x}_i)/(a(\mathbf{x}_i)+\varepsilon)$ over $i = 1,2,\ldots,N$. The term ε is

necessary to compute $s(\mathbf{x}_i)$ when $a(\mathbf{x}_i)$ is 0, i.e., when all objects from cluster **A** are equal to one another. This modified objective function seems to be more sensitive to slight changes in $a(\mathbf{x}_i)$ and $b(\mathbf{x}_i)$, which in turn might correspond to significant changes in the clustering solution, with the price that the criterion is no longer bounded within the interval [−1, +1].

In the recent paper by Ma et al. [97], the authors propose to assess the fitness of each genotype by means of an algorithm with two main steps[8]. In brief, the first step is aimed at discovering statistically significant association patterns in the partition encoded into the genotype. To this end, some objects from different clusters are randomly selected to form a training set for pattern discovery. In the step 2 of the fitness computation, the reclassification accuracy of the objects not selected in step 1 is evaluated and the final fitness value is calculated based on this accuracy. Ma et al. [97] claim that this fitness function works for both fixed and variable number of clusters, though only experiments involving a set of user-defined interesting values for k were reported. An interesting characteristic of this function is that, the larger the number of clusters, the smaller the accuracy tends to be (because the classification problem becomes harder), and so this fitness function implicitly tends to favor the discovery of smaller numbers of clusters. Ma et al. [97] claim that their fitness function has been conceived to deal with noisy and missing data as well as to distinguish between relevant and irrelevant features for the clustering process. In this sense, it is worth taking the opportunity to make the following important remarks:

(i) Evolutionary clustering algorithms that are not capable of automatically handling incomplete data sets can benefit from a number of imputation techniques (e.g., [127][110][77]), as a preprocessing procedure. In addition, if the proportion of missing values is low, just the known values may be enough for computing unbiased pairwise (dis)similarity measures.

(ii) Evolutionary clustering algorithms that are not capable of automatically distinguishing between relevant and irrelevant features for the clustering process can benefit from a number of feature selection techniques (e.g. see [78], [90] and references therein), as a preprocessing procedure. Some of those techniques are additionally endowed with the ability to remove redundant features (which may still impact the clustering process even after removal of the irrelevant features).

4) Selection: Proportional selection has been used by several authors (e.g., Cole [23]; Cowgill et al. [26]; Tseng and Yang [128]; Bandyopadhyay and Maulik [12][11]; Casillas et al. [21], Hruschka and Ebecken [65]; Hruschka et al. [69][70][68]; Ma et al. [97]; Alves et al. [2]; Naldi and de Carvalho [108]). Alves et al. [2] also mention the use of a (μ+λ)-like deterministic/elitist selection. The evolutionary algorithm for multi-objective clustering proposed by Handl and Knowles [59] is based on the PESA-II algorithm [25], whose selection principles rely on the interface between two populations: an internal population that explores new solutions by standard processes of reproduction and variation, and an external population that exploits good solutions by elitism. Pan and Cheng [113] adopt a selection procedure based on Tabu search [53].

As previously mentioned in Section II.A.4, the advantages and disadvantages of traditional selection mechanisms are well-known in the evolutionary computation literature and, as far as we know, there is no strong evidence that the relative behavior of these mechanisms is much different when assessed in the particular context of evolutionary clustering.

5) Initial Population: In the papers by Cole [23], Cowgill et al. [26], Hruschka and Ebecken [65], Hruschka et al. [69][70][68], Ma et al. [97], Naldi and de Carvalho [108], and Alves et al. [2], the initial population for the algorithm is generated from random assignments of objects to clusters. As previously noticed in Section II.A.5, such an initialization strategy usually results in unfavorable initial partitions, since the initial clusters are likely to be mixed up to a high degree. It constitutes, however, an effective approach to test the algorithms against tough evaluation scenarios.

In Tseng and Yang [128], the population of binary genotypes is randomly generated in such a way that the number of 1's in each individual is uniformly distributed within [1, m], where m is the number of clusters initially generated. Initial data partitions are randomly generated in the algorithm proposed by Pan and Cheng [113]. Bandyopadhyay and Maulik [12][11] randomly select objects from the data set to be the initial prototypes of the clusters to be evolved by their genetic algorithm. Handl and Knowles [59] employ minimum spanning trees and the k-means algorithm to generate initial clustering solutions. Casillas et al. [21] do not describe the procedure they used to generate the initial population.

6) Local Search by k-means: Some evolutionary algorithms designed for estimating the number of clusters make use of the k-means clustering algorithm as a local search procedure (e.g. [69][70][2][68][108]). From this particular viewpoint, k-means fundamentally performs a fine-tuning of some rough partitions obtained by the evolutionary search, thus speeding up its convergence. In a broader view, a synergy between k-means and evolutionary operators can be achieved. On the one hand, k-means minimizes the variances of the clusters achieved by the evolutionary algorithm operators, thus yielding to more compact clusters. On the other hand, evolutionary algorithms can lessen the two main drawbacks of k-means, namely: (i) it may get stuck at sub-optimal centroids; and (ii) the user has to specify the number of clusters (k). Since the evolutionary operators can eliminate, split, and merge clusters through an evolutionary search, they are able to evolve better partitions in terms of both the number of clusters and centroids. These partitions may provide better initial centroids for k-means, thus reducing the probability of getting stuck at sub-optimal solutions.

[8] The intermediate sub-steps and the corresponding formulae have been omitted here for the sake of compactness. Please, refer to [97] for further details.

III. OVERLAPPING CLUSTERING

Recall from the introductory section that a *hard* partition of a data set $\mathbf{X}=\{\mathbf{x}_1,\mathbf{x}_2, ...,\mathbf{x}_N\}$ is a collection $\mathbf{C}=\{\mathbf{C}_1,\mathbf{C}_2, ...,\mathbf{C}_k\}$ of k non-overlapping data subsets (clusters) such that $\mathbf{C}_1 \cup \mathbf{C}_2 \cup ... \cup \mathbf{C}_k = \mathbf{X}$ and $\mathbf{C}_i \cap \mathbf{C}_j = \varnothing$ for $i \neq j$. If the condition of mutual disjunction ($\mathbf{C}_i \cap \mathbf{C}_j = \varnothing$ for $i \neq j$) is relaxed, then the partitions and the corresponding algorithms are said to be of overlapping type. Overlapping algorithms produce data partitions that can be *soft* (each object fully belongs to one or more clusters – partial membership is not allowed) [40] or *fuzzy* (each object belongs to one or more clusters to different degrees) [118][64].

There are only a few works in the literature devoted to evolutionary algorithms that search for soft partitions. At a high level of abstraction, it can be asserted that these algorithms use the same kind of encoding scheme, which has been termed *cluster description-based representation* in [50]. Following the lines of [50], a genotype under *cluster description-based representation* explicitly represents the parameters necessary to precisely specify each cluster. The exact nature of these parameters depends on the shape of clusters to be produced, which could be, e.g., boxes, spheres, ellipsoids, etc. In any case, each genotype contains k sets of parameters, where k is the number of clusters, and each set of parameters determines the position, shape and size of its corresponding cluster. This way, different clusters may cover common areas of the data space and, in such a case, any object relying on these areas will be considered to belong non-exclusively to the corresponding overlapping clusters. Besides, the cluster description is non-exhaustive in the sense that some objects may not be within any cluster. This kind of representation is used, e.g., in [123][52][45][46].

A. Fuzzy Clustering

When a fuzzy clustering algorithm is applied to a data set with N objects, the final result is a partition of the data into a certain number k of *fuzzy clusters*, such that:

$$\begin{cases} P = \left[\mu_{ij}\right]_{k\times N} \\ \mu_{ij} \in \left[0,1\right] \end{cases} \tag{15}$$

where P is a $k \times N$ *fuzzy partition matrix* whose element μ_{ij} represents the fuzzy membership of the jth object to the ith fuzzy cluster. When μ_{ij} is limited to the extreme values of its feasibility interval, i.e., $\mu_{ij} \in \{0,1\}$, then P degenerates to a soft partition. Besides, if the additional constraint $\Sigma_i \mu_{ij} = 1$ is imposed to every column j of the matrix, then P degenerates to a standard hard partition. The representation of a data partition in terms of a hard partition matrix corresponds precisely to the *matrix-based binary encoding* scheme described in Section II.A.1.a.

A fuzzy partition matrix provides additional information about the data that is not available in its soft or hard counterparts. In fact, the fuzzy membership values μ_{ij} can help discover more sophisticated relations between the corresponding data objects and disclosed clusters [132]. In addition, in contrast to their Boolean relatives, the continuous membership values of fuzzy partitions are particularly appropriate to describe boundaries between ambiguous or blurred clusters that are not clearly separated from each other. Owing to these desired properties, the applicability of fuzzy clustering is broad in scope and includes areas such as pattern classification, image segmentation, document categorization, data visualization, and dynamic systems identification, just to mention a few [16][64][5][32].

Most of the research on evolutionary algorithms for overlapping clustering has focused on algorithms that evolve fuzzy partitions of data. In this context, many authors have proposed evolutionary algorithms to solve fuzzy clustering problems for which the number of clusters is known or set in advance by the user [56][57][80][15][134][130][37][58][91]. However, as previously discussed in the introductory section, the optimal number of clusters is usually unknown in advance. For this reason, more recent papers have proposed to optimize both the number of clusters and the corresponding fuzzy partitions by some form of evolutionary search [115][89][99][111][67][1][19][44].

Regardless of the fixed or variable nature of the number of clusters, the evolutionary algorithms for fuzzy clustering are mostly based on extensions – to the fuzzy domain – of the fundamental ideas discussed in Section II for hard partitional clustering. This is in conformity with the fact that most fuzzy clustering algorithms are based on generalizations of traditional algorithms for hard clustering, as it is the case of the well-known Fuzzy C-Means (FCM) algorithm and its variants [16][64][5], which are essentially generalizations of the classic k-means algorithm. Only a few exceptions (e.g. see [125]) try to develop operators that could act directly on fuzzy partitions of data. Contrarily, most authors have chosen to adapt the existing evolutionary clustering techniques to the fuzzy domain, not only for convenience, but mainly because, to date, there is no strong evidence that the more complex fully fuzzy formulation of the problem can be counterbalanced by efficiency and/or efficacy gains. This is an interesting open question still to be tackled.

Roughly speaking, the evolutionary algorithms for fuzzy clustering that are somehow based upon adaptations of existing approaches developed for hard evolutionary clustering (see Section II) can be broadly divided into two main categories. The first (and most representative) one is composed of algorithms that encode and evolve prototypes for the FCM algorithm or for one of its variants [37][4][91][115][80][15][56][57][58][99][111][89]. In this case, the prototypes are encoded and manipulated using essentially the same techniques already discussed in Section II. Essentially, the only differences between these algorithms and their hard counterparts discussed in Section II are: (i) they

have to compute the fuzzy partition corresponding to every genotype; and (ii) they use as fitness functions clustering validity criteria that are capable of assessing fuzzy partitions. In what concerns fuzzy validity criteria, we refer the interested reader to [64][20][112] and references therein. The fuzzy partition corresponding to every genotype, by its turn, is computed as a function of the prototypes using the standard FCM-like formulae.

The second category of evolutionary fuzzy clustering algorithms is composed of algorithms that use some variant of FCM as a local search operator to speed up their convergence by refining rough partitions explored by the evolutionary search, while providing the necessary computations to get the fuzzy partition [1][19][67]. In [19] it is shown that an evolutionary algorithm is able to outperform, in terms of computational efficiency, traditional approaches to determine satisfactory estimates of the unknown number of fuzzy clusters, under both the theoretical (asymptotic time complexity analyses) and experimental (statistical analyses) perspectives.

IV. ENSEMBLES AND MULTI-OBJECTIVE CLUSTERING

As previously mentioned, there is no single definition of clustering and each clustering algorithm, or even different runs of the same algorithm, may produce different partitions for the same data set. The partitions produced are also influenced by the validity criterion adopted. Two approaches have been proposed in the literature to reduce these limitations: multi-objective clustering and ensembles. In this section we will discuss the use of these two approaches combined with evolutionary algorithms.

A. Multi-Objective Evolutionary Clustering

Unlike supervised learning tasks, such as classification, clustering is an unsupervised learning task, so that there is no "ground truth" to tell us what the "correct" or "optimal" solution is. This suggests that the quality of a clustering solution should be evaluated by a diverse set of validity criteria, rather than a single criterion, in order to mitigate the strong bias imposed by any particular validity criterion.

In practice, multiple criteria, considering different aspects of the quality of a clustering solution, often represent conflicting goals for an optimization method. Consider, for instance, the well-known and previously mentioned criterion of minimizing the sum of intra-cluster distances. In a general scenario where the number of clusters is variable, optimizing only this criterion is not enough, because it can be trivially minimized by assigning each object to a distinct singleton cluster. In this scenario, clearly, we also need to penalize a candidate solution for having a large number of clusters, i.e., we also want to favor solutions with a small number of large clusters. Unfortunately these two objectives – minimizing intra-cluster distances and minimizing the number of clusters – are conflicting with each other.

This raises the question of how an evolutionary algorithm should cope with such conflicting objectives – assuming we have decided that the algorithm will use two or more conflicting validity criteria, which is often desirable in practice. The conventional and simplest approach would be to convert the corresponding multi-objective clustering problem into a single-objective one, by defining the fitness function as a weighted formula where different weights are assigned to different objectives. However, this approach has several drawbacks, such as (in a nutshell): mixing non-commensurable objectives (e.g. distance and number of clusters) into the same formula; requiring an ad-hoc assignment of weight values to different objectives, which often requires many runs of the algorithm to try to "optimize" the weight values, etc. These drawbacks are extensively discussed in the literature – see e.g. [22] and [33].

A more principled solution consists of developing a truly multi-objective evolutionary algorithm for clustering, i.e. an algorithm with the main characteristic of using a multi-objective function following the principle of Pareto dominance. According to the definition of this type of dominance relation between two candidate solutions, a candidate clustering solution c_1 dominates another candidate clustering solution c_2 if and only if: (a) c_1 is strictly better than c_2 in at least one of all the objectives considered in the fitness function; and (b) c_1 is not worse than c_2 in any of the objectives considered in the fitness function.

When using this type of Pareto dominance-based, multi-objective fitness function, the goal of the evolutionary algorithm is to find the "Pareto front", i.e. the set of all non-dominated solutions. Note that, since the goal is to return to the user a set of non-dominated solutions, rather than a single solution as it is typically the case in single-objective optimization, a multi-objective evolutionary algorithm necessarily needs special mechanisms, such as elitism procedures that preserve non-dominated solutions, selection methods adapted to cope with the Pareto dominance concept, genetic operators that promote diversity in the population (to favor the discovery of non-dominated solutions as spread as possible across the Pareto front), etc. Such special mechanisms are well documented in the literature – again, see [22] and [33] – and they are normally generic mechanisms that can be applied regardless of whether the evolutionary algorithm is solving a clustering problem or another kind of problem. Therefore, in this section, we focus only on the crucial aspect of multi-objective evolutionary algorithms that is very specific to the clustering task, namely, the definition of the multi-objective fitness function. Hence, let us now briefly review the multi-objective fitness function of some evolutionary algorithms for clustering reported in the literature.

In an evolutionary approach for multi-objective clustering, Handl and Knowles [59] use a fitness function based on both compactness and connectedness of clusters. Cluster

compactness is expressed by means of the *overall deviation* of a partitioning – Equation (2). Connectedness is measured by the degree to which neighboring objects have been placed in the same cluster:

$$\text{Conn}(\mathbf{C}) = \sum_{i=1}^{N} \sum_{j=1}^{L} z_{i,\mathbf{n}_{ij}} \quad (16)$$

where $\mathbf{n}_{ij}$ is jth nearest neighbor of object $\mathbf{x}_i$, L is the number of neighbors that contribute to the measure, and $z_{(.)}$ is given by:

$$z_{i,\mathbf{n}_{ij}} = \begin{cases} \frac{1}{j} \text{ if } \nexists \mathbf{C}_k : \mathbf{x}_i \in \mathbf{C}_k \wedge \mathbf{n}_{ij} \in \mathbf{C}_k \\ 0 \text{ otherwise.} \end{cases} \quad (17)$$

The authors [59] remark that while the objective value associated with the *overall deviation* – Equation (2) – necessarily improves with an increasing number of clusters, the opposite is the case for the connectivity – Equation (16). The interaction of these two objective functions allows keeping the number of clusters stable, thus avoiding convergence to trivial solutions whilst allowing exploration of interesting regions of the search space.

Another evolutionary algorithm for multi-objective clustering is described in [82]. The two objectives used in this work are the total intra-cluster variation (computed over all clusters) and the number of clusters. Both objectives should be minimized, but they are conflicting with each other, as previously explained. Hence, by using the concept of Pareto dominance, the algorithm manages to discover a diverse set of non-dominated clustering solutions, where, for each different number of clusters, the algorithm can find the smallest possible total intra-cluster variance. This allows us to present a set of solutions with different trade-offs between the two objectives to the user, who can then make a more informed choice about the solution to be used in practice.

Bandyopadhyay et al. [8] proposed a multi-objective evolutionary algorithm that performs fuzzy clustering. There are two objectives being simultaneously optimized. One of them is J_m defined in (5). The other is the well-known Xie-Beni index [131], which is essentially a ratio of a global measure of intra-cluster variation divided by a local measure of cluster separation – namely, the distance between the two closest clusters. Note that the numerator of this ratio is similar to the first objective, but the denominator is measuring an aspect of clustering quality not captured by the first objective. Hence, the value of this second objective will be optimized (minimized) when all clusters have an intra-cluster variation as small as possible and when the two closest clusters are as far away from each other as possible.

In a variation of this work, Mukhopadhyay et al. [106] use two objectives where the first one is also a measure of total intra-cluster variation, but they simplify the second measure to be a direct global measure of the clusters' separation (essentially the total summation of distances between all pairs of clusters).

Ripon et al. [117] also propose a multi-objective evolutionary clustering algorithm with two objectives. The first one is essentially, as usual, a kind of measure of average *intra*-cluster variation computed over all clusters. The measure's average value across clusters was used – rather than the measure's total summation across clusters – in order to produce a normalized value of the measure taking into account the number of clusters, which varies for different individuals in the evolutionary algorithm's population. The second objective is a measure of *inter*-cluster distance, which measures the average distance separating a pair of clusters – computed over all pairs of clusters.

Another relevant work is the multi-objective evolutionary algorithm described in [78]. In this work, the algorithm is not used for solving a clustering problem, and so it is not strictly within the scope of this paper. However, the algorithm is used to select attributes for a clustering algorithm (*k*-means), and this is considered relevant enough to be mentioned here, since most of the objectives in the fitness function used in this work could equally well be used in an evolutionary algorithm for clustering. This is because the evolutionary algorithm is used as a wrapper around a clustering algorithm, so that the fitness of an individual (i.e., a candidate set of selected attributes) is computed by running a clustering algorithm with the selected attributes and measuring the corresponding clustering validity criteria. More precisely, this work used a fitness function with four different objectives (clustering validity criteria), namely: (a) cluster cohesiveness – related to intra-cluster distance; (b) separation between clusters – related to inter-cluster distance; (c) number of clusters; and (d) number of selected attributes. Note how these objectives cover different aspects of the quality of a clustering solution and are seamlessly integrated in a Pareto-based multi-objective fitness function.

To summarize, out of the six works mentioned above, five use two objectives and one uses four objectives. Hence, researchers have focused mainly on just two objectives, probably for the sake of simplicity. In addition, in all the six works previously mentioned, one of the objectives used was a kind of total *intra*-cluster distance measure (to be minimized), and five of those works used an objective related to a kind of *inter*-cluster distance measure (to be maximized). In principle, a larger number of objectives could be considered, to try to discover better clusters, and it would be interesting to investigate the use of a larger diversity of types of objective functions for measuring clustering quality.

B. Ensemble-Based Evolutionary Clustering

The combination of techniques in a group or ensemble is easily found in many classification and regression applications. In these applications, the outputs provided by different techniques are combined by one of several strategies in order to provide a consensus output value. The main goal is the improvement of the overall performance in terms of accuracy or precision by trying to use the best features of each individual technique [86]. For such, these approaches use

either the label of the class (classification) or the desired value (regression).

A formal definition of cluster ensemble is given by Topchy et al. [126]. Given a set of P partitions $\prod = \{\pi^1, \pi^2, \ldots, \pi^P\}$ of a data set resulting from several applications of one or more clustering algorithms, the goal is to look for a final partition (consensus partition), π^F, of better quality than the initial partitions (partitions basis). The best quality depends on the clustering validity criterion adopted.

More informally, Handl and Knowles [59][60] say that clustering ensembles occur by the combination of a set of partitions previously produced by several runs of a single algorithm or by a set of algorithms. The use of ensemble is not so straightforward for clustering techniques. To begin with, there is no label associated with each object. Therefore, more sophisticated strategies are needed in order to combine partitions found by different algorithms or different runs of the same algorithm in a consensus partition.

According with Fred and Jain [49], the partition obtained by the combination of the initial partitions should be consistent or agree in some way with them, be robust to small variations in these partitions, and be consistent with external information regarding the structure of the data (if that information is available).

The main steps for the combination of multiple partitions are the induction of the partitions to be combined and the definition of a consensus function to combine these partitions [126]. Next, the main approaches followed in the literature for each step are briefly presented.

The initial partitions can be produced by different clustering algorithms, several runs of the same clustering algorithm (with different initial seeds), several runs of a weak clustering algorithm (clustering algorithms simpler than the conventional algorithms) and by several sub-samplings of the same original data set by the same clustering algorithm.

Regarding the consensus function, the most usual functions are based on co-association, graph, mutual information, and voting. A function based on co-association tries to keep together objects found together in most of the individual partitions [49]. The graph-based functions look for a consensus partition using partitioning techniques employed for graphs [124]. The functions based on mutual information maximize the mutual information between the labels of the initial partitions and the labels of the consensus partition. The voting function, after labeling the clusters, defines how many times each object belonged to each cluster. Each object can be assigned to its most frequent cluster. The definition of the correspondence of labels for different partitions is not simple.

In spite of the difficulty associated with this issue, there are several works investigating the ensemble of partitions [49][86][124][51][76][133][42]. However, only the last two works use a genetic algorithm for producing a clustering ensemble. Another of these works [76] uses genetic algorithms as one of the individual clustering algorithms of an ensemble.

Yoon et al. [133] use multiple crossover repetitions to combine partitions created by different clustering algorithms. Each pair selected for a crossover operation should present a high overlap in the cluster objects. The initial population comprises all clusters created by the clustering algorithms used in the ensemble. The authors argue that their method, named heterogeneous clustering ensemble (HCE), differ from other ensemble approaches by taking characteristics from the individual algorithms and the data set into account during the ensemble procedure. This method was compared with individual clustering algorithms using a gene-expression data set.

Handl and Knowles [59][60] agree with other authors in that ensembles tend to be more robust and produce higher quality solutions than a single partition produced by an individual clustering algorithm. However, they point out that ensembles do not explore the full potential of multi-objective clustering, since clusters that cannot be detected by one of the ensemble components probably will not be present in the final population.

Of course, it is also possible to combine multi-objective clustering with clustering ensemble, as is the case of the work of Faceli et al. [42][41], which proposes a Multi-Objective Clustering Ensemble method, named MOCLE. This method combines an ensemble of data partitions with multi-objective clustering. The initial population used by MOCLE is a set $\prod=\{\pi^1,\pi^2,\ldots,\pi^P\}$ of P partitions, where $\pi^i=\{\mathbf{C}^i_1,\mathbf{C}^i_2,\ldots,\mathbf{C}^i_{k(i)}\}$ is a partition of the data set $\mathbf{X}=\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ into $k(i)$ clusters, such that $\mathbf{C}^i_1 \cup \mathbf{C}^i_2 \cup \ldots \cup \mathbf{C}^i_{k(i)} = \mathbf{X}$. The initial partitions are of varying quality and are induced by different clustering algorithms with different values for their free parameters. These partitions usually represent a large variety of clusters types. MOCLE evolves towards a concise, stable and robust set of alternative structures, which should represent different views of the data set.

V. APPLICATIONS

The application fields for evolutionary clustering algorithms are essentially the same as those for non-evolutionary algorithms, though, in practice, the use of evolutionary approaches is seemingly more appropriate when no domain knowledge (e.g. about the approximate number of clusters) is available.

Next, we will briefly comment on some applications found in the literature of evolutionary clustering. These applications will be divided into five groups: image processing, bioinformatics, computational finance, RBF neural network design and others.

In image processing applications, evolutionary clustering algorithms are mainly used to identify regions with particular interest in an image. Bandyopadhyay and her co-authors, for example, applied evolutionary clustering [10][11] and fuzzy clustering [100][111] algorithms to distinguish landscape regions like rivers, habitations and vegetation areas in satellite images. In [8] the authors applied multi-objective clustering

genetic algorithms to pixel classification in remote sensing imagery. In a similar application by Liu and Chapman [89], evolutionary clustering is employed for automated road extraction from satellite imagery. In a third image processing application, Scheunders [121] dealt with quantization of color images. Hall et al. [58] applied a genetically guided fuzzy clustering strategy to brain tissue MRI (Magnetic Resonance Image) quantization. Das et al. [27] applied a differential evolution algorithm to automatic segmentation of images.

Several authors have reported the application of evolutionary clustering algorithms to bioinformatics, particularly to gene-expression analysis. Such applications are important due to the growing amounts of gene-expression data quickly becoming available and the need for more sophisticated analysis tools for medical diagnosis. There are two main groups of gene-expression analysis applications: identification of groups of genes that have closely related expression levels (in order, for instance, to better understand and diagnose some diseases using tissues as input attributes) – see [115], [97], [103], [95], [70], [68], [66], and [9] – and the discovery of new sub-groups of pathologies, using the gene-expression levels as input attributes – by looking for different structures in a gene-expression data set [42][108][35].

For finance applications, evolutionary clustering algorithms have been used to group either customer or company profiles, as in the works from Lacerda et al. [87], for credit risk assessment, and Krovi [84], for cluster analysis of bankrupt and non-bankrupt companies.

Another application is the evolutionary design of other machine learning techniques. Different approaches for the hybrid training of RBF networks using evolutionary clustering to select the number and location of basis functions were investigated in [88][29].

There are many other applications where evolutionary clustering has been successfully used. They include, for example, the work from Sarafis et al. [120], who applied an evolutionary algorithm to the problem of clustering earthquakes, using data from an earthquake catalog; the work from Casillas et al. [21], applying genetic algorithm to document clustering; the use of genetic clustering for intrusion detection in computer networks, by Liu et al. [92]; and the use of a clustering genetic algorithm to extract rules from multilayer perceptrons [71].

VI. CONCLUSIONS

A. Summary

This paper presents an up-to-date survey on evolutionary algorithms for clustering. It tries to reflect the profile of this area by focusing more on those subjects that have been given more importance in the literature. Particularly, the paper has focused mainly on hard partitional algorithms, though overlapping (soft/fuzzy) approaches have also been covered.

An original contribution of the present paper is that it discusses key issues on the design of evolutionary algorithms for data partitioning problems, such as usually adopted representations, evolutionary operators, and fitness functions, just to mention a few. In particular, mutation and crossover operators commonly described in the literature are conceptually analyzed, giving especial emphasis to those genetic operators specifically designed for clustering problems (i.e., cluster-oriented and context-sensitive operators). In addition, advantages and disadvantages of the most common representation schemes are discussed, and asymptotic comparative analyses in terms of running time and memory space requirements are reported. Finally, several references are provided that describe applications of evolutionary algorithms for clustering in different domains, such as image processing, computer security, and bioinformatics.

For the sake of clarity, algorithms designed for fixed and variable number of clusters have been reviewed separately. In brief, algorithms that assume a fixed number of clusters (k) – e.g., Bandyopadhyay and Maulik [10]; Estivill-Castro and Murray [39]; Fränti et al. [48]; Kivijärvi et al. [79]; Krishna and Murty [83]; Krovi [84]; Bezdek et al. [14]; Kuncheva and Bezdek [85]; Lu et al. [95][94]; Lucasius et al. [96]; Maulik and Bandyopadhyay [100]; Merz and Zell [103]; Murthy and Chowdhury [107]; Scheunders [121]; Sheng and Liu [122] – are particularly suitable for applications in which there is information regarding k, notably when domain knowledge is available that suggests a reasonable value for the number of clusters. In practice, this situation may take place when there is good reason to believe that a small interval of values for k may contain the "best" number of clusters, k^*. In other situations, clustering algorithms that explore partitions with different numbers of clusters are often preferred (e.g., Cole [23]; Cowgill et al. [26]; Bandyopadhyay and Maulik [12][11]; Hruschka and Ebecken [65]; Casillas et al. [21]; Hruschka et al. [69][70][68]; Ma et al. [97]; Alves et al. [2]; Tseng and Yang [128]; Pan and Cheng [113]; and Handl and Knowles [59]). Such evolutionary algorithms search for both k^* and the corresponding optimal partition in regions of the space where they are more likely to be found.

Table II provides a summary of evolutionary algorithms designed for optimizing fixed and variable numbers of clusters. Additionally, this table also allows a unified view of the algorithms for the reader particularly interested in the nature of the data structure used for manipulating the clusters during the evolutionary search. Such data structures have been categorized into three main types, namely: *centroid-based*, *medoid-based*, and *label-based*. Algorithms whose main data structures do not strictly adhere to those just mentioned (e.g. *tree-based* and *graph-based*) are listed in the last row of the table.

The next section closes the paper by suggesting some topics for future research that, in the authors' opinion, should deserve special attention from the scientific community interested in evolutionary algorithms for clustering.

TABLE II. SUMMARY OF EVOLUTIONARY ALGORITHMS FOR HARD PARTITIONAL CLUSTERING.

	Fixed *k*	**Variable *k***
Label-Based	Krovi [84] Murthy/Chowdhury [107] Krishna/Murty [83] Lu et al. [95][94]	Cole [23] Cowgill et al. [26] Hruschka et al. [65][69] Hruschka et al. [70][68] Alves et al. [2] Ma et al. [97]
Centroid-Based	Scheunders [121] Fränti et al. [48] Merz/Zell [103] Kivijärvi et al. [79] Bandyopadhyay/Maulik [10][100]	Bandyopadhyay/Maulik [12][11]
Medoid-Based	Kuncheva/Bezdek [85] Lucasius et al. [96] Estivill-Castro/Murray [39] Sheng /Liu [122]	
Others	Bezdek et al. [14]	Casillas et al. [21] Tseng/Yang [128] Handl/Knowles [59] Pan and Cheng [113]

B. Future Trends

In most of the references on evolutionary algorithms for clustering, only the quality of the partitions is of concern, whereas little attention has been given to computational efficiency, which is a critical issue when one thinks of serious large-scale data clustering problems. We shall note that traditional randomized approaches[9] can possibly find solutions as good as those found by evolutionary algorithms (mainly if these are based on local search engines, such as *k*-means or FCM). From this standpoint, an important question to be answered is: What are the scenarios in which evolutionary algorithms are more computationally efficient than traditional randomized approaches? More generally, one may want to know the relative computational efficiency of evolutionary algorithms when compared with other (probabilistic) algorithms designed for a given clustering task. Since efficiency issues are almost untouched in the literature that addresses evolutionary algorithms for clustering, one might have the impression that there is an implicit claim suggesting that these algorithms are in general efficient. It is likely that such an implicit claim is mostly based on the fact that evolutionary algorithms are widely believed to be effective on (any) NP-hard global optimization problems, being able to provide near-optimal solutions in reasonable time. However, it is important to bear in mind that, in practice, the success of an evolutionary algorithm to "solve" a given problem is highly dependent upon how it has been designed (in terms of encoding scheme, operators, set of parameters, etc.). According to the conventional wisdom of good science, ideally such design choices should be carefully analyzed (theoretically and/or empirically). However, due to the probabilistic nature of the search process performed by evolutionary algorithms, such analyses are usually hard to be accomplished. In the authors' opinion, that is why this issue is still almost untouched in the related literature [19]. Particularly, most of the literature on evolutionary clustering does not provide detailed theoretical analyses in terms of time complexity. We believe that this issue is an important research area for future work. In addition, research efforts aimed at investigating the computational efficiency of evolutionary and non-evolutionary approaches in a systematic, empirical fashion, making use of rigorous statistical analyses, are in order.

There is also much work to be done on investigating the theoretical underpinnings of evolutionary algorithms for clustering. The book by Falkenauer [43] can be considered as a pioneering work in this research direction. In brief, the author seriously questioned the applicability of the schema theorem [63][54] for the particular context of grouping problems (broadly defined) being solved by standard genetic algorithms. Although such a book carefully elaborates on this subject, as well as it provides deep insights on how genetic algorithms for clustering problems should be designed, formal justifications are not given. Instead, the author emphasizes the need of meticulously choosing both the encoding scheme and the operators in such a way that they make sense with respect to the structure of grouping problems. Doing so, the author argues that the premises of the schema theorem can be satisfied, and valid approaches can be derived to tackle grouping problems by using genetic algorithms. Since the remarkable book of Falkenauer was published, in 1998, several evolutionary algorithms specifically designed for particular data partitioning problems have been proposed, and good results have been reported in many applications. However, formal analyses concerning the theoretical soundness of such algorithms are still largely untouched in the literature and are worth of investigation.

Another research direction that deserves more investigation is multi-objective clustering. Recall that there is no "ground truth" in clustering, and so it is important to consider multiple objectives (different clustering validity criteria) when evaluating the fitness of an individual representing a candidate clustering solution. Although multi-objective evolutionary algorithms for clustering have already been proposed, this research topic is currently under-explored in the literature.

Finally, a topic that is almost untouched in the literature is the combination of evolutionary approaches with traditional hierarchical clustering algorithms, especially those derived from the Lance-Williams scheme. This is probably due to the fact that it is not straightforward to define a fitness function capable of guiding the evolution of dendrograms. To the best of our knowledge, only Lozano and Larrañaga [93] address this topic, which is a possible venue for future research.

[9] Systematically executing a partitional clustering algorithm multiple times, possibly for different numbers of clusters, and then selecting the particular partition that provides the best result according to a specific relative validity criterion [72][64].

ACKNOWLEDGEMENTS

We would like to express our appreciation to the anonymous referees of the original manuscript for the constructive comments they made.

REFERENCES

[1] V. S. Alves, R. J. G. B. Campello, E. R. Hruschka, "A Fuzzy Variant of an Evolutionary Algorithm for Clustering", *In Proc. IEEE Int. Conference on Fuzzy Systems*, pp. 375-380, 2007.

[2] V. S. Alves, R. J. G. B. Campello, E. R. Hruschka, "Towards a Fast Evolutionary Algorithm for Clustering", *In Proc. IEEE Congress on Evolutionary Computation*, pp. 6240-6247, 2006.

[3] L. J. Arabie, G. Hubert, P. DeSoete, *Clustering and Classification*, World Scientific, 1999.

[4] G. P. Babu, M. N. Murty, "Clustering with Evolution Strategies", *Pattern Recognition*, vol. 27, pp. 321-329, 1994.

[5] R. Babuška, *Fuzzy Modeling for Control*, Kluwer, 1998.

[6] T. Bäck, D. B. Fogel, Z. Michalewicz, *Evolutionary Computation 1: Basic Algorithms and Operators*, Institute of Physiscs Publishing (IOP), Bristol and Philadelphia, 2000.

[7] P. Baldi, S. Brunak, *Bioinformatics - The Machine Learning Approach*, 2nd Ed., MIT Press, 2001.

[8] S. Bandyopadhyay, U. Maulik, A. Mukhopadhyay, Multiobjective genetic clustering for pixel classification in remote sensing imagery. IEEE Transactions on Geoscience and Remote Sensing, v. 45, n. 5, pp. 1506-1511, 2007.

[9] S. Bandyopadhyay, A. Mukhopadhyay, U. Maulik, An improved algorithm for clustering gene expression data, Bioinformatics, v. 23, n. 21, pp. 2859-2865, 2007.

[10] S. Bandyopadhyay, U. Maulik, "An Evolutionary Technique based on *k*-Means Algorithm for Optimal Clustering in R^N", *Information Sciences*, Vol. 146, pp. 221-237, 2002.

[11] S. Bandyopadhyay, U. Maulik, "Genetic Clustering for Automatic Evolution of Clusters and Application to Image Classification", *Pattern Recognition*, Vol. 35, pp. 1197-1208, 2002.

[12] S. Bandyopadhyay, U. Maulik, "Nonparametric Genetic Clustering: Comparison of Validity Indices", *IEEE Trans. on Systems, Man, and Cybernetics – Pt. C*, Vol. 31, pp. 120-125, 2001.

[13] P. M. BertoneGerstein, "Integrative Data Mining: The New Direction in Bioinformatics – Machine Learning for Analyzing Genome-Wide Expression Profiles", *IEEE Engineering in Medicine and Biology*, Vol. 20, pp. 33-40, 2001.

[14] J. C. Bezdek, S. Boggavaparu, L. O. Hall, A. Bensaid, "Genetic Algorithm Guided Clustering", *In Proc. IEEE Congress on Evolutionary Computation*, pp. 34-40, 1994.

[15] J. C. Bezdek, R. J. Hathaway, "Optimization of Fuzzy Clustering Criteria using Genetic Algorithms", *In Proc. IEEE World Congress on Computational Intelligence*, pp. 589-594, 1994.

[16] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithm*, Plenum Press, 1981.

[17] J. P. Bigus, *Data Mining with Neural Networks*, McGraw-Hill, 1996.

[18] R. B. Calinski, J. Harabasz, "A Dendrite Method for Cluster Analysis", *Communications in Statistics*, Vol. 3, pp. 1-27, 1974.

[19] R. J. G. B Campello, V. S. Alves, E. R. Hruschka, "On the Efficiency of Evolutionary Fuzzy Clustering", *Journal of Heuristics*, DOI: 10.1007/s10732-007-9059-6.

[20] R. J. G. B. Campello, E. R. Hruschka, "A Fuzzy Extension of the Silhouette Width Criterion for Cluster Analysis" Fuzzy Sets and Systems, vol. 157, n. 21, pp. 2858-2875, 2006.

[21] A. Casillas, M. Y. González de Lena, R. Martínez, "Document Clustering into an Unknown Number of Clusters Using a Genetic Algorithm", *In. Proc. Int. Conference on Text Speech and Dialogue*, LNCS 2807, pp. 43-49, 2003.

[22] C. A. Coello Coello, D. A. Van Veldhuizen, G. B. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems.* Kluwer, 2002.

[23] R. M. Cole, *Clustering with Genetic Algorithms*, MSc Thesis, University of Western Australia, Australia, 1998.

[24] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to Algorithms*, MIT Press, 2nd Ed., 2001.

[25] D. W. Corne, N. R. Jerram, J. D. Knowles, M. J. Oates, "PESA-II: Region-Based Selection in Evolutionary Multiobjective Optimization", *In Proc. Genetic and Evolutionary Computation Conference*, pp. 283–290, 2001.

[26] M. C. Cowgill, R. J. Harvey, L. T. Watson, A Genetic Algorithm Approach to Cluster Analysis, *Computational Mathematics and its Applications*, Vol. 37, pp. 99-108, 1999.

[27] S. Das, A. Abraham, A. Konar, "Automatic Clustering Using an Improved Differential Evolution Algorithm", IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans, vol. 38, n.1, January, 2008.

[28] D. L. Davies, D. W. Bouldin, "A Cluster Separation Measure", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol.1, pp. 224-227, 1979.

[29] L. N. de Castro, E. R. Hruschka, R. J. G. B. Campello, "An Evolutionary Clustering Technique with Local Search to Design RBF Neural Network Classifiers", In Proc. of the IEEE Int. Conference on Neural Networks, pp. 2083-2088, 2004.

[30] K. A. de Jong, *Evolutionary Computation: A Unified Approach*, MIT Press, 2006.

[31] C. S. de Oliveira, A. S. G. Meiguins, B.S. Meiguins, P.I. Godinho, A.A. Freitas, An evolutionary density and grid-based clustering algorithm. *In Proc. XXIII Brazilian Symposium on Databases (SBBD-2007)*, pp. 175-189, 2007.

[32] J.V. de Oliveira, W. Pedrycz, Advances in Fuzzy Clustering and its Applications, Wiley, 2007.

[33] K. Deb, *Multi-objective Optimization Using Evolutionary Algorithms*. Wiley, 2001.

[34] A. P. Dempster, N. Laird, D. B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm", *Journal of the Royal Statistical Society*, Vol. B39, pp. 1-38, 1977.

[35] M. Dolled-Filhart, L. Ryden, M. Cregger, K. Jirstrom, M. Harigopal, R. L. Camp, D. L. Rimm, Classification of breast cancer using genetic algorithms and tissue microarrays. *Clin Cancer Res* **12:** 6459-6468, 2006.

[36] J. C. Dunn, "A Fuzzy Relative of the ISODATA Process and its use in Detecting Compact Well-Separated Clusters", *Journal of Cybernetics*, Vol. 3, pp. 32-57, 1973.

[37] M. A. Egan, M. Krishnamoorthy, K. Rajan, "Comparative Study of a Genetic Fuzzy C-Means Algorithm and a Validity Guided Fuzzy C-Means Algorithm for Locating Clusters in Noisy Data", *In Proc. IEEE World Congress on Computational Intelligence*, pp. 440-445, 1998.

[38] M. Ester, H.-P. Kriegel, J. Xu, W. Sander, A density-based algorithm for discovering clusters in large spatial databases with noise. *In Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining (KDD-06)*, pp. 226-231. AAAI Press, 1996.

[39] V. Estivill-Castro, A. T. Murray, "Spatial Clustering for Data Mining with Genetic Algorithms", *In Proc. Int. ICSC Symposium on Engineering of Intelligent Systems*, pp. 317-323, 1997.

[40] B. S. Everitt, S. Landau, M. Leese, *Cluster Analysis*, Arnold Publishers, 2001.

[41] K. Faceli, A. C. P. L. F. de Carvalho, M. C. P. Souto, "Cluster Ensemble and Multi-objective Clustering Methods, In: "*Pattern Recognition Technologies and Applications: Recent Advances,* Brijesh Verma; Michael Blumenstein. (Editors.). Hershey, Idea Group (to appear).

[42] K. Faceli, A. C. P. L. F. de Carvalho, M. C. P. Souto, "Multi-objective Clustering Ensemble", *International Journal of Hybrid Intelligent Systems*, Vol. 4(3), pp. 145-156, 2007.

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

[44] P. Fazendeiro, J. Valente de Oliveira, "A Semantic Driven Evolutive Fuzzy Clustering Algorithm", *In Proc. IEEE Int. Conference on Fuzzy Systems*, pp. 1-6, 2007.

[45] D. B. Fogel, *Evolutionary Computation: Principles and Practice for Signal Processing*, SPIE (Society of Photo-Optical Instrumentation Engineers) Press, 2000.

[46] D. B. Fogel, P. K. Simpson, "Evolving Fuzzy Clusters", *In Proc. IEEE Int. Conference on Neural Networks*, pp. 1829-1834, 1993.

[47] C. Fralley, A. E. Raftery, "How Many Clusters? Which Clustering Method? Answer via Model-Based Cluster Analysis", *The Computer Journal*, Vol. 41, pp. 578-588, 1998.

[48] P. Fränti, J. Kivijärvi, T. Kaukoranta, O. Nevalainen, "Genetic Algorithms for Large-Scale Clustering Problems", *The Computer Journal*, Vol. 40, pp. 547-554, 1997.

[49] A. L. N. Fred, A. K. Jain, "Combining multiple clusterings using evidence accumulation", *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. 27 (6), pp. 835-850, 2005.

[50] A. A. Freitas, *A Review of Evolutionary Algorithms for Data Mining*, In: *Soft Computing for Knowledge Discovery and Data Mining,* pp. 61-93, O. Maimon; L. Rokach (Editors), Springer, 2007.

[51] J. Ghosh, A. Strehl, S. Merugu, A Consensus Framework for Integrating Distributed Clusterings Under Limited Knowledge Sharing, Proc. of NSF Workshop on Next Generation Data Mining, pp. 99-108, Baltimore, MD, November, 2002.

[52] A. Ghozeil, D. B. Fogel, "Discovering Patterns in Spatial Data using Evolutionary Programming", *In Proc. 1st Annual Conference on Genetic Programming*, pp. 521-527, 1996.

[53] F. Glover, Tabu Search: Part II, ORSA J. Comput., v.2, n.1, pp.4-32, 1990.

[54] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, 1989.

[55] M. Halkidi, Y. Batistakis, M. Vazirgiannis, "On Clustering Validation Techniques", *Journal of Intelligent Information Systems*, Vol. 17, pp. 107-145, 2001.

[56] L. O. Hall, J. C. Bezdek, S. Boggavarpu, A. Bensaid, "Genetic Fuzzy Clustering", *In Proc. Annual Conference of the North American Fuzzy Information Processing Society (NAFIPS)*, pp. 411-415, 1994.

[57] L. O. Hall, B. Özyurt, "Scaling Genetically Guided Fuzzy Clustering", *In Proc. Int. Symposium on Uncertainty Modeling and Analysis & Annual Conference of the North American Fuzzy Information Processing Society (ISUMA-NAFIPS)*, pp. 328-332, 1995.

[58] L. O. Hall, I. B. Özyurt, J. C. Bezdek, "Clustering with a Genetically Optimized Approach", *IEEE Trans. on Evolutionary Computation*, Vol. 3, pp. 103-112, 1999.

[59] J. Handl, J. Knowles, "An Evolutionary Approach to Multiobjective Clustering", *IEEE Trans. on Evolutionary Computation*, Vol. 11, pp. 56-76, 2007.

[60] J. Handl, J. Knowles, D. B. Kell, "Computational Cluster Validation in Post-Genomic Data Analysis", *Bioinformatics*, Vol. 21, pp. 3201–3212, 2005.

[61] T. Hastie, R. Tibshirani, J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer, 2001.

[62] S. Haykin, *Neural Networks – A Comprehensive Foundation*, Prentice Hall, 2nd Ed., 1999.

[63] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, 1975.

[64] F. Höppner, F. Klawonn, R. Kruse, T. Runkler, *Fuzzy Cluster Analysis: Methods for Classification, Data Analysis and Image Recognition*, Wiley, 1999.

[65] E. R. Hruschka, N. F. F. Ebecken, "A Genetic Algorithm for Cluster Analysis", *Intelligent Data Analysis*, Vol. 7, pp. 15-25, 2003.

[66] E. R. Hruschka, R. J. G. B. Campello, L. N. de Castro, "Clustering Gene-Expression Data: A Hybrid Approach

that Iterates between k-Means and Evolutionary Search", In: *Hybrid Evolutionary Algorithms*, C. Grosan, A. Abraham, H. Ishibuchi (Editors), Springer, pp. 313-335, 2007.

[67] E. R. Hruschka, R. J. G. B Campello, L. N. de Castro, "Evolutionary Search for Optimal Fuzzy C-Means Clustering", *In Proc. Int. Conference on Fuzzy Systems*, pp. 685-690, 2004.

[68] E. R. Hruschka, R. J. G. B. Campello, L. N. de Castro, "Evolving Clusters in Gene-Expression Data", *Information Sciences*, Vol. 176, pp. 1898-1927, 2006.

[69] E. R. Hruschka, R. J. G. B. Campello, L. N. de Castro, "Improving the Efficiency of a Clustering Genetic Algorithm", *In Proc. 9th Ibero-American Conference on Artificial Intelligence*, LNCS 3315, pp. 861-870, 2004.

[70] E. R. Hruschka, L. N. de Castro, R. J. G. B. Campello, "Evolutionary Algorithms for Clustering Gene-Expression Data", *In Proc. 4th IEEE Int. Conference on Data Mining*, pp. 403-406, 2004.

[71] E. R. Hruschka, N. F. F. Ebecken, "Extracting Rules from Multilayer Perceptrons in Classification Problems: A Clustering-Based Approach", *Neurocomputing*, Vol. 70, pp. 384-397, 2006.

[72] A. K. Jain, R. C. Dubes, *Algorithms for Clustering Data*, Prentice Hall, 1988.

[73] A. K. Jain, M. N. Murty, P. J. Flynn, "Data Clustering: A Review", *ACM Computing Surveys*, Vol. 31, pp. 264-323, 1999.

[74] D. Jiang, C. Tang, A. Zhang, "Cluster Analysis for Gene Expression Data: A Survey", *IEEE Trans. on Knowledge and Data Engineering*, Vol. 16, pp. 1370-1386, 2004.

[75] L. Kaufman, P. J. Rousseeuw, *Finding Groups in Data – An Introduction to Cluster Analysis*, Wiley Series in Probability and Mathematical Statistics, 1990.

[76] P. Kellam, X. Liu, N. J. Martin, C. Orengo, S. Swift, A. Tucker, "Comparing, contrasting and combining clusters in viral gene expression data", *In Proc. 6th Workshop on Intelligent Data Analysis in Medicine and Pharmacology*, pp. 56–62, 2001.

[77] H. Kim, G. H. Golub, H. Park, "Missing Value Estimation for DNA Microarray Gene Expression Data: Local Least Squares Imputation", *Bioinformatics*, Vol. 21, pp. 187-198, 2005.

[78] Y. Kim, W. N. Street, F. Menczer, Feature selection in unsupervised learning via evolutionary searching. In Proc. 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2000), pp. 365-369. ACM Press, 2000.

[79] J. Kivijärvi, P. Fränti, O. Nevalainen, "Self-Adaptive Genetic Algorithm for Clustering", *Journal of Heuristics*, Vol. 9, pp. 113-129, 2003.

[80] F. Klawonn, "Fuzzy Clustering with Evolutionary Algorithms", *In Proc. of 7th Int. Fuzzy Systems Association (IFSA) World Congress*, pp. 312-323, 1997.

[81] D. E. Knuth, The Art of Computing Programming, vol. III: Sorting and Searching, Addison-Wesley, Reading, MA, 1973.

[82] E. E. Korkmaz, J. Du, R. Alhajj, K. Barker, Combining advantages of new chromosome representation scheme and multi-objective genetic algorithms for better clustering. *Intelligent Data Analysis, Vol. 10, No. 2,* pp, 163-182, 2006.

[83] K. Krishna, N. Murty, "Genetic K-means Algorithm", *IEEE Trans. on Systems, Man and Cybernetics – Pt. B*, Vol. 29, pp. 433-439, 1999.

[84] R. Krovi, "Genetic Algorithms for Clustering: A Preliminary Investigation", *In Proc. of the 25th Hawaii Int. Conference on System Sciences*, Vol. 4, pp. 540-544, 1992.

[85] L. I. Kuncheva, J. C. Bezdek, "Selection of Cluster Prototypes from Data by a Genetic Algorithm", *In Proc. 5th European Congress on Intelligent Techniques and Soft Computing*, pp. 1683-1688, 1997.

[86] L. I. Kuncheva, S. T. Hadjitodorov, L. P. Todorova, "Experimental comparison of cluster ensemble methods", *In Proc. FUSION*, pp. 105-115, 2006.

[87] E. G. Lacerda, A. C. P. F. de Carvalho, A. P. Braga, T. B. Ludermir, "Evolutionary Radial Basis Functions for Credit Assessment", *Applied Intelligence*, pp. 167-181, Vol. 22 (3), Springer, May 2005.

[88] E. G. Lacerda, A. C. P. F. de Carvalho, T. B. Ludermir, "Evolutionary Optimization of RBF networks, Radial basis function neural networks: design and applications", pp. 282-310, *In Radial Basis Function Networks 1: Recent Developments in Theory and Application*, R. J. Howlett, L Jain (Editors), Physica-Verlag, 2001.

[89] H. Liu, J. Li, M. A. Chapman, "Automated Road Extraction from Satellite Imagery using Hybrid Genetic Algorithms and Cluster Analysis", *Journal of Environmental Informatics*, Vol. 1, no. 2, pp. 40-47, 2003.

[90] H. Liu, L. Yu, "Toward Integrating Feature Selection Algorithms for Classification and Clustering", *IEEE Trans. on Knowledge and Data Engineering*, Vol. 17, pp. 1-12, 2005.

[91] J. Liu, W. Xie, "A Genetics-Based Approach to Fuzzy Clustering", *In Proc. Int. Conference on Fuzzy Systems*, pp. 2233-2240, 1995.

[92] Y. Liu, K. Chen, X. Liao, W. Zhang, "A Genetic Clustering Method for Intrusion Detection", *Pattern Recognition*, Vol. 37, pp. 927-942, 2004.

[93] J. A. Lozano, P. Larrañaga, Applying genetic algorithms to search for the best hierarchical clustering of a dataset, Pattern Recognition Letters, n. 20, pp. 911-918, 1999.

[94] Y. Lu, S. Lu, F. Fotouhi, Y. Deng, S. J. Brown, "FGKA: A Fast Genetic K-means Clustering Algorithm", *In Proc. ACM Symposium on Applied Computing*, pp. 622-623, 2004.

[95] Y. Lu, S. Lu, F. Fotouhi, Y. Deng, S. J. Brown, "Incremental Genetic k-Means Algorithm and its Application in Gene Expression Data Analysis", *BMC Bioinformatics*, Vol. 28, 172, 2004.

[96] C. B. Lucasius, A. D. Dane, G. Kateman, "On *k*-Medoid Clustering of Large Data Sets with the Aid of a Genetic

Algorithm: Background, Feasibility and Comparison", *Analytica Chimica Acta*, Vol. 282, pp. 647-669, 1993.

[97] P. C. H. Ma, K. C. C. Chan, X. Yao, D. K. Y. Chiu, "An Evolutionary Clustering Algorithm for Gene Expression Microarray Data Analysis", *IEEE Trans. on Evolutionary Computation*, Vol. 10, pp. 296-314, 2006.

[98] T. Martinez, K. Schulten, Topology Representing Networks, Neural Networks, v.7, n. 3, pp. 507-522, Elsevier, 1994.

[99] U. Maulik, S. Bandyopadhyay, "Fuzzy Partitioning Using Real Coded Variable Length Genetic Algorithm for Pixel Cassification", *IEEE Trans. on Geosciences and Remote Sensing*, Vol. 41, no. 5, pp. 1075–1081, 2003.

[100] U. Maulik, S. Bandyopadhyay, "Genetic Algorithm-based Clustering Technique", *Pattern Recognition*, Vol. 33, pp. 1455-1465, 2000.

[101] J. B. McQueen, "Some Methods of Classification and Analysis of Multivariate Observations", In: *Proc. 5th Berkeley Symposium on Mathematical Statistics and Probability*, pp. 281-297, 1967.

[102] G. Mecca, S. Raunich, A. Pappalardo, "A New Algorithm for Clustering Search Results", *Data and Knowledge Engineering*, Vol. 62, pp. 504-522, 2007.

[103] P. Merz, A. Zell, "Clustering Gene Expression Profiles with Memetic Algorithms", *In Proc. Parallel Problem Solving from Nature*, LNCS 2439, pp. 811-820, 2002.

[104] G. W. Milligan, M. C. Cooper, "An Examination of Procedures for Determining the Number of Clusters in a Data Set", *Psychometrika*, Vol. 50, pp. 159-179, 1985.

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

[106] A. Mukhopadhyay, U. Maulik, S. Bandyopadhyay, Multiobjective genetic fuzzy clustering of categorical attributes. *Proc. 10th Int. Conf. on Information Technology,* pp. 74-79. IEEE Computer Society, 2007.

[107] C. A. Murthy, N. Chowdhury, "In Search of Optimal Clusters using Genetic Algorithms", *Pattern Recognition Letters*, Vol. 17, pp. 825-832, 1996.

[108] M. C. Naldi, A. C. P. L. F. de Carvalho, "Clustering Using Genetic Algorithm Combining Validation Criteria", *In Proc. 15th European Symposium on Artificial Neural Networks,* pp. 139-147, Bruges, Belgium, 2007.

[109] Naldi, M. C., de Carvalho, A. C. P. L. F., Campello, R. J. G. B., Hruschka, E. R., *Genetic Clustering for Data Mining*. In: *Soft Computing for Knowledge Discovery and Data Mining*, O. Maimon; L. Rokach (Editors), Springer, pp. 113-132, 2007.

[110] M. Ouyang, W. J. Welsh, P. Georgopoulos, "Gaussian Mixture Clustering and Imputation of Microarray Data", *Bioinformatics*, Vol. 20, pp. 917-923, 2004.

[111] M. K. Pakhira, S. Bandyopadhyay, U. Maulik, "A Study of some Fuzzy Cluster Validity Indices, Genetic Clustering and Application to Pixel Classification", *Fuzzy Sets and Systems*, Vol. 155, pp. 191-214, 2005.

[112] N. R. Pal, J. C. Bezdek, "On Cluster Validity for the Fuzzy c-Means Model", *IEEE Trans. on Fuzzy Systems*, Vol. 3, pp. 370-379, 1995.

[113] S. Pan, K. Cheng, Evolution-Based Tabu Search Approach to Automatic Clustering, IEEE Transactions on Systems, Man, and Cybernetics, Part C – Applications and Reviews, v. 37, n. 5, pp. 827-838, 2007.

[114] P. A. Pantel, *Clustering by Commitee*, PhD Thesis, Department of Computer Sciences of the University of Alberta, Canada, 2003.

[115] H.-S. Park, S.-H. Yoo, S.-B. Cho, "Evolutionary Fuzzy Clustering Algorithm with Knowledge-Based Evaluation and Applications for Gene Expression Profiling", *Journal of Computational and Theoretical Nanoscience*, Vol. 2, pp. 1-10, 2005.

[116] V. J. Rayward-Smith, "Metaheuristics for Clustering in KDD", *In Proc. IEEE Congress on Evolutionary Computation*, pp. 2380-2387, 2005.

[117] K.S.N. Ripon, C.-H. Tsang, S. Kwong, M.-K. Ip, Multi-objective evolutionary clustering using variable-length real jumping genes genetic algorithm. *Proc. of the 18th Int. Conf. on Pattern Recognition (ICPR'06)*. IEEE Computer Society, 2006.

[118] E. Ruspini, "Numerical Methods for Fuzzy Clustering", *Information Sciences*, Vol. 2, pp. 319-350, 1970.

[119] I. Sarafis, *Data Mining Clustering of High Dimensional DataBases with Evolutionary Algorithms*, PhD Thesis, Heriot-Watt University, UK, 2005.

[120] L. Sarafis, P. W. Trinder, A. M. S. Zalzala, NOCEA: a rule-based evolutionary algorithm for efficient and effective clustering of massive high-dimensional databases. *Applied Soft Computing, Vol. 7, No. 3,* pp. 668-710, June 2007.

[121] P. Scheunders, "A Genetic *c*-Means Clustering Algorithm Applied to Color Image Quantization", *Pattern Recognition*, Vol. 30, pp. 859-866, 1997.

[122] W. Sheng, X. Liu, "A Hybrid Algorithm for K-Medoid Clustering of Large Data Sets", *In Proc. IEEE Congress on Evolutionary Computation*, pp. 77-82, 2004.

[123] R. Srikanth, R. George, N. Warsi, D. Prabhu, F. E. Petry, B. P. Buckles, "A Variable-Length Genetic Algorithm for Clustering and Classification", *Pattern Recognition Letters*, Vol. 16, pp. 789-800, 1995.

[124] A. Strehl, J. Ghosh, "Cluster ensembles - a knowledge reuse framework for combining partitions", *Journal of Machine Learning Research*, Vol. 3, pp. 583-617, 2002.

[125] H. Sun, S. Wang, Q. Jiang, "FCM-Based Model Selection Algorithms for Determining the Number of Clusters", *Pattern Recognition Letters*, Vol. 37, pp. 2027-2037, 2004.

[126] A. Topchy, A. Jain, W. Punch, "A mixture model for clustering ensembles", *In Proceedings of the SIAM International Conference on Data Mining*, pp. 331–338, 2004..

[127] O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein, R. B. Altman, "Missing Value Estimation Methods for DNA Microarray", *Bioinformatics*, Vol. 17, pp. 520-525, 2001.

[128] L. Y. Tseng, S. B. Yang, "A Genetic Approach to the Automatic Clustering Problem", *Pattern Recognition*, Vol. 34, pp. 415-424, 2001.

[129] F. Valafar, "Pattern Recognition Techniques in Microarray Data Analysis: A Survey", *Annals of New York Academy of Sciences*, Vol. 980, pp. 41-64, 2002.

[130] T. Van Le, "Evolutionary Fuzzy Clustering", *In Proc. IEEE Congress on Evolutionary Computation*, pp. 753-758, 1995.

[131] X. L. Xie, G. Beni, "A validity measure for fuzzy clustering," IEEE Transactions on Pattern Analysis and Machine Intelligence, v. 13, n. 8, pp. 841–847, 1991.

[132] R. Xu, D. Wunsch II, "Survey of Clustering Algorithms", *IEEE Trans. on Neural Networks*, Vol. 16, pp. 645-678, 2005.

[133] H.-S. Yoon, S.-Y. Ahn, S.-H. Lee, S.-B. Cho, J. H. Kim, "Heterogeneous Clustering Ensemble Method for Combining Different Cluster Results", Proc. *BioDM 2006*, pp. 82-92, Lecture Notes in Computer Science, Vol. 3916, 2006.

[134] B. Yuan, G. J. Klir, J. F. Swan-Stone, "Evolutionary Fuzzy C-Means Clustering Algorithm", *In Proc. Int. Conference on Fuzzy Systems*, pp. 2221-2226, 1995.

Eduardo Raul Hruschka received his B.Sc. degree in Civil Engineering from Federal University of Paraná, Brazil, in 1995, and his M.Sc. and Ph.D. degrees in Computational Systems from Federal University of Rio de Janeiro, Brazil, in 1998 and 2001, respectively. He is currently assistant professor of the Department of Computer Sciences of the University of São Paulo (USP) at São Carlos, Brazil. His primary research interests are in data mining, with particular emphasis on clustering algorithms, evolutionary computation, feature selection, missing values imputation, and artificial neural networks. He has authored or coauthored more than 50 research publications in peer-reviewed reputed journals, book chapters, and conference proceedings. Dr. Hruschka has been a reviewer for several journals such as IEEE Transactions on Systems, Man and Cybernetics, IEEE Transactions on Knowledge and Data Engineering, IEEE Transactions on Evolutionary Computation, Information Sciences, Journal of Heuristics, and Bioinformatics. He has also been a member of the Program Committee of several international conferences, including the IEEE International Conference on Data Mining.

Ricardo J. G. B. Campello was born in Recife - PE, Brazil. He received the BSc degree in Electronics Engineering from State University of São Paulo (Unesp), Ilha Solteira - SP, in 1994, and the MSc and Ph.D. degrees in Electrical Engineering from the School of Electrical and Computer Engineering of the State University of Campinas (Unicamp), Campinas - SP, in 1997 and 2002, respectively. In 2002 he was a visiting scholar at the Laboratoire D'Informatique, Signaux et Systèmes de Sophia Antipolis, Université de Nice - Sophia Antipolis (UNSA), France. Since 2007 he is with the Department of Computer Sciences of the University of São Paulo (USP) at São Carlos. His current research interests fall primarily into the areas of Soft Computing, Machine Learning, Data Mining and Dynamic Systems Identification/Control.

Alex A. Freitas obtained his BSc in Computer Science from FATEC-SP, Brazil, in 1989; his MSc in Computer Science from UFSCar, Brazil, in 1993; and his PhD in Computer Science from the University of Essex, UK, in 1997. He worked as a visiting Lecturer at CEFETP-PR, Brazil, in 1998; and as an Associate Professor at PUC-PR, Brazil, from 1999 to June 2002. In July 2002 he moved to the University of Kent, UK, where he is now a Reader in Computational Intelligence and the Director of Research of the Computing Laboratory. He is a member of the editorial board of three international journals, namely: Intelligent Data Analysis, The International Journal of Data Warehousing and Mining, and the International Journal of Computational Intelligence and Applications. He has authored two research-oriented books (both in the area of data mining), and has published more than 10 invited book chapters and more than 100 refereed papers in journals and conference proceedings. His current research interests are data mining and knowledge discovery, biologically-inspired computational intelligence algorithms and bioinformatics. He is a member of IEEE, AAAI (Association for Advancement of Artificial Intelligence), BCS-SGAI (British Computer Society's Specialist Group on Artificial Intelligence), and ACM SIGKDD (Special Interest Group on Knowledge Discovery and Data Mining).

André C. Ponce de Leon F. de Carvalho received his B.Sc. and M.Sc. degrees in Computer Science from the Universidade Federal de Pernambuco, Brazil. He received his Ph.D. degree in Electronic Engineering from the University of Kent, UK. Prof. André de Carvalho is Full Professor at the Department of Computer Science, Universidade de São Paulo, Brazil. He has published around 60 Journal and 200 Conference refereed papers. He has been involved in the organization of several conferences and journal special issues. His main interests are Machine Learning, Data Mining, Bioinformatics, Evolutionary Computation, Bioinspired Computing and Hybrid Intelligent Systems.