A Survey of Multiobjective Evolutionary Clustering

ANIRBAN MUKHOPADHYAY, Department of Computer Science and Engineering, University of Kalyani, West Bengal, India
UJJWAL MAULIK, Department of Computer Science and Engineering, Jadavpur University, West Bengal, India
SANGHAMITRA BANDYOPADHYAY, Machine Intelligence Unit, Indian Statistical Institute, West Bengal, India

Data clustering is a popular unsupervised data mining tool that is used for partitioning a given dataset into homogeneous groups based on some similarity/dissimilarity metric. Traditional clustering algorithms often make prior assumptions about the cluster structure and adopt a corresponding suitable objective function that is optimized either through classical techniques or metaheuristic approaches. These algorithms are known to perform poorly when the cluster assumptions do not hold in the data. Multiobjective clustering, in which multiple objective functions are simultaneously optimized, has emerged as an attractive and robust alternative in such situations. In particular, application of multiobjective evolutionary algorithms for clustering has become popular in the past decade because of their population-based nature. Here, we provide a comprehensive and critical survey of the multitude of multiobjective evolutionary clustering techniques existing in the literature. The techniques are classified according to the encoding strategies adopted, objective functions, evolutionary operators, strategy for maintaining nondominated solutions, and the method of selection of the final solution. The pros and cons of the different approaches are mentioned. Finally, we have discussed some real-life applications of multiobjective clustering in the domains of image segmentation, bioinformatics, web mining, and so forth.

Categories and Subject Descriptors: G.1.6 [**Mathematics of Computing**]: Optimization—*Global optimization*; H.2.8 [**Information Systems**]: Database Management—*Database applications; Data mining*; I.5.3 [**Computing Methodologies**]: Pattern Recognition—*Clustering; Algorithms*

General Terms: Algorithms

Additional Key Words and Phrases: Clustering, multiobjective optimization, evolutionary algorithms, Pareto optimality

ACM Reference Format:
Anirban Mukhopadhyay, Ujjwal Maulik, and Sanghamitra Bandyopadhyay. 2015. A survey of multiobjective evolutionary clustering. ACM Comput. Surv. 47, 4, Article 61 (May 2015), 46 pages.
DOI: http://dx.doi.org/10.1145/2742642

A. Mukhopadhyay acknowledges the support received from the DST-PURSE grant at the University of Kalyani. U. Maulik and S. Bandyopadhyay acknowledge the grant under the DST-INRIA-CNRS targeted program project titled "A big data perspective for energy management in smart grids and dwellings."
Authors' addresses: A. Mukhopadhyay, Department of Computer Science and Engineering, University of Kalyani, Kalyani-741235, West Bengal, India; email: anirban@klyuniv.ac.in; U. Maulik, Department of Computer Science and Engineering, Jadavpur University, Kolkata-700032, West Bengal, India; email: umaulik@cse.jdvu.ac.in; S. Bandyopadhyay, Machine Intelligence Unit, Indian Statistical Institute, Kolkata-700108, West Bengal, India; email: sanghami@isical.ac.in.

1. INTRODUCTION

Clustering [Jain et al. 1999; Jain and Dubes 1988] is a popular unsupervised exploratory pattern classification technique that partitions an input space into K regions $\{C_1, C_2, \ldots, C_K\}$ based on some similarity/distance measure. The number of clusters K may or may not be known *a priori*. Given a set of objects $X = \{x_1, x_2, \ldots, x_n\}$ of size n in a multidimensional space, the goal of clustering algorithms is usually to produce a $K \times n$ partition matrix $U(X)$. The partition matrix is denoted by $U = [u_{kj}]$, $k = 1, 2, \ldots, K$ and $j = 1, 2, \ldots, n$. Here, u_{kj} corresponds to the membership of object x_j to cluster C_k. In crisp or hard clustering, $u_{kj} = 1$ if $x_j \in C_k$, and $u_{kj} = 0$ otherwise. In contrast, for fuzzy clustering [Bezdek 1981; Pal and Bezdek 1995] of the data, $0 \leq u_{kj} \leq 1$, and the following conditions hold on U (representing nondegenerate clustering): $0 < \sum_{j=1}^{n} u_{kj} < n$, $\sum_{k=1}^{K} u_{kj} = 1$, and $\sum_{k=1}^{K} \sum_{j=1}^{n} u_{kj} = n$ [Maulik et al. 2011].

Clustering algorithms try to evolve a suitable grouping of the points of the input dataset so that some measures are optimized. Thus, the problem of clustering can be posed as an optimization problem [Maulik et al. 2011]. The objective functions to be optimized may represent various properties of the clusters, such as compactness, separation, and connectivity. Since goodness of any clustering solution is usually measured with respect to some cluster validity index [Bezdek and Pal 1998; Maulik and Bandyopadhyay 2002; Wanga and Zhanga 2007], the most common way to pose clustering as an optimization problem is to optimize a validity index. The search space is composed of all possible groupings of data points and the respective validity index scores. Popular partitional clustering methods like K-means [Jain and Dubes 1988] and fuzzy C-means [Bezdek 1981] perform greedy search over the search space for optimizing the cluster compactness. Even though these techniques have low time complexity, they often get trapped into local optima depending on the cluster centers selected in the beginning [Selim and Ismail 1984; Groll and Jakel 2005]. Moreover, they optimize only one cluster validity measure (cluster compactness), and thus may not be able to capture diverse characteristics of the datasets.

To address the problem of local optima, some global optimization techniques like Genetic Algorithms (GAs) [Goldberg 1989; Holland 1975] have been widely employed to obtain the global optimum value of the chosen validity measure. GAs are randomized search and optimization methods that are guided by the rules of evolution theory. They contain a large amount of inbuilt parallelism. GAs execute multimodal search in complex landscapes and usually yield near-optimal solutions for the objective or fitness function of an optimization problem [Bandyopadhyay and Pal 2007]. Conventional GA-based clustering techniques [Maulik and Bandyopadhyay 2000] optimize a single-cluster validity index as the objective function. However, due to different data properties, a single validity measure does not work equally well for all datasets. Thus it is more beneficial to optimize multiple such validity indices simultaneously to capture different characteristics of the datasets.

Simultaneous optimization of multiple objectives helps achieve improved robustness toward diverse data properties [Maulik et al. 2011]. Hence, it is useful to utilize multiobjective GAs (MOGAs) for clustering. In multiobjective optimization (MOO) [Deb 2001; Coello Coello et al. 2007; Maulik et al. 2011], search is performed over a number of conflicting objective functions. There are three major approaches to MOO [Freitas 2004]: (i) Transformation of the multiobjective problem into a single-objective problem using a weighted formula; (ii) ordering the objectives according to importance (lexicographic approach); and (iii) the Pareto-based approach, in which multiple objective functions are optimized simultaneously to yield a final solution set containing a number of nondominated solutions. None of these nondominated solutions can further be improved with respect to any one objective without degrading the solution with respect to another objective. The last approach is regarded as the most principled approach in the multiobjective data-mining community [Freitas 2004]. Multiobjective clustering techniques [Bandyopadhyay et al. 2007a; Handl and Knowles 2007] optimize more than one cluster validity index simultaneously, leading to high-quality results. The relative importance of different clustering criteria is usually unknown. Therefore, it is better to optimize them simultaneously instead of aggregating them in a single criterion to be optimized. The yielded set of near-Pareto-optimal solutions consists of different nondominated solutions, which the user can compare subjectively to find the most promising ones for the problem at hand [Maulik et al. 2011].

In the past decade, multiobjective evolutionary algorithms have been heavily used for the clustering problem. However, there has been no dedicated effort to review all of these methods. The most prominent effort in this direction can be found in Hruschka et al. [2009], in which a generic survey of evolutionary algorithms for data clustering has been made with respect to different encoding strategies, capability of capturing number of clusters, different evolutionary operations, and so on. The authors have also discussed a few multiobjective clustering algorithms in the article. However, due to the enormous development of multiobjective evolutionary algorithms for clustering and a wide variety of approaches, we felt the necessity for a dedicated survey of these techniques. Moreover, since Hruschka et al. [2009], a number of new multiobjective evolutionary algorithms for data clustering have come out. Recently, in Mukhopadhyay et al. [2014a] and Mukhopadhyay et al. [2014b], we have surveyed multiobjective evolutionary algorithms for different data-mining tasks. In the first part of the article [Mukhopadhyay et al. 2014a], multiobjective approaches for feature selection and classification problems have been reviewed. In the second part [Mukhopadhyay et al. 2014b], multiobjective algorithms for clustering, association rule mining and other data-mining tasks have been surveyed. Therefore, only a small portion in Mukhopadhyay et al. [2014b] was devoted to multiobjective clustering. It may be noted that the literature of multiobjective evolutionary algorithms for clustering is very rich, and there are multiple issues involved. The most that could be done in Mukhopadhyay et al. [2014b] was to provide a few pointers and references to some relevant works. Multiobjective clustering involved a lot of different issues that could not be adequately addressed in the earlier article. We thus felt the need for a dedicated survey on multiobjective evolutionary algorithms for clustering.

In this article, a comprehensive coverage of the subject has been provided so that anyone starting research in this area is provided all the relevant information for a complete understanding. The purpose of this article is to introduce the readers to almost all facets of multiobjective evolutionary clustering, such as encoding strategies, choice of objective functions, effect of objective functions on encoding strategies, evolutionary operators (e.g., selection, crossover and mutation), variation of operators depending on solution representation, maintaining nondominated solutions, and different approaches for obtaining the final solution from the nondominated set. As most of the multiobjective clustering algorithm employed some MOGAs as the underlying optimization tool, we have mainly concentrated on MOGAs for clustering. However, clustering methods based on other multiobjective evolutionary algorithms, such as differential evolution, genetic programming, and gene expression programming, are also discussed. We have critically presented the merits and demerits of the techniques with respect to these issues. Moreover, we have discussed the different applications of multiobjective evolutionary clustering algorithms in different fields of science and engineering. We feel that the present article is ideal for anyone intending to start research in multiobjective evolutionary clustering or for someone who wants to become familiar with the state-of-the-art in this field.

2. MULTIOBJECTIVE OPTIMIZATION AND CLUSTERING

In this section, first we describe the different concepts of multiobjective optimization and provide an overview of available multiobjective evolutionary algorithms. Thereafter, the general framework of a multiobjective evolutionary clustering algorithm is described.

2.1. Concepts of Multiobjective Optimization

Many real-world situations demand simultaneous optimization of multiple objectives for solving certain problems. Thus, conventional GAs, which optimize a single objective function, cannot tackle these multiobjective problems. In multiobjective optimization, it is usually difficult to compare one solution with another due to the presence of more than one objective value. Therefore, it is not easy to define optimality for these problems, since in most of the cases one cannot find a single solution that provides the best values for all the objective functions. Usually, these problems yield a set of solutions. In absence of any knowledge about relative importance of the objective functions, all these solutions are considered to be equally acceptable and equivalent. The best solution is often subjective and depends on the requirements of the problem and the decision maker [Maulik et al. 2011; Deb 2001; Coello Coello et al. 2007; Mukhopadhyay et al. 2014a].

Multiobjective optimization can formally be stated as follows [Deb 2001; Coello Coello 1999; Coello Coello et al. 2007; Coello Coello 2006]: Find the vector $\overline{x}^* = [x_1^*, x_2^*, \ldots, x_t^*]^T$ of decision variables that will satisfy the α inequality constraints

$$g_i(\overline{x}) \geq 0, \quad i = 1, 2, \ldots, \alpha, \tag{1}$$

and the β equality constraints

$$h_i(\overline{x}) = 0, \quad i = 1, 2, \ldots, \beta, \tag{2}$$

and optimize the vector function

$$\overline{f}(\overline{x}) = [\overline{f_1}(\overline{x}), \overline{f_2}(\overline{x}), \ldots, \overline{f_{\mathcal{K}}}(\overline{x})]^T. \tag{3}$$

The constraints given in Equations (1) and (2) define the feasible region $\mathcal{F}$ of acceptable solutions. The solutions residing outside this region are unacceptable since they fail to satisfy one or more constraints. The vector $\overline{x}^*$ denotes an optimal solution in $\mathcal{F}$. As stated earlier, in the context of multiobjective optimization, the important problem is the definition of optimality. This is because it is very uncommon that one will find a situation in which a single vector $\overline{x}^*$ provides the optimum values for all the objective functions.

The notion of Pareto optimality is helpful in the context of multiobjective optimization. For minimization problems, a formal definition of Pareto optimality may be given as follows [Maulik et al. 2011; Deb 2001; Coello Coello et al. 2007; Mukhopadhyay et al. 2014a]: A decision vector $\overline{x}^*$ is called Pareto-optimal if and only if there is no $\overline{x}$ that dominates $\overline{x}^*$, that is, there is no $\overline{x}$ such that

$$\forall i \in 1, 2, \ldots, \mathcal{K}, \; f_i(\overline{x}) \leq f_i(\overline{x}^*)$$

and

$$\exists i \in 1, 2, \ldots, \mathcal{K}, \; f_i(\overline{x}) < f_i(\overline{x}^*).$$

In other words, $\overline{x}^*$ is called Pareto-optimal if there exists no feasible vector $\overline{x}$ that causes a reduction in an objective value without a simultaneous increase in at least another. In general, a Pareto-optimum represents a set of solutions called *nondominated* solutions.

It is not easy to extend the conventional search and optimization methods such as gradient descent search and other nonconventional ones such as simulated annealing to the multiobjective case, since their basic principles do not permit them to track more than one solution at a time. On the contrary, population-based techniques such as Evolutionary Algorithms are appropriate and suitable for handling such situations. There are several approaches for multiobjective optimization using evolutionary algorithms [Maulik et al. 2011; Mukhopadhyay et al. 2014a]. In aggregating approaches, the multiobjective problem is modeled as a single-objective one by combining the objective functions using some weights and then optimizing the combined objective by traditional evolutionary algorithms. In population-based non-Pareto approaches such as Vector Evaluated Genetic Algorithm (VEGA) [Deb 2001], a special selection operator is used and a number of subpopulations are generated by applying proportional selection based on each objective function in turn. Among the Pareto-based approaches, MOGA [Fonseca and Fleming 1993], Niched Pareto GA (NPGA) [Horn and Nafpliotis 1993], and Nondominated Sorting GA (NSGA) [Srinivas and Deb 1994] are nonelitist approaches. Although the concept of Pareto optimality is considered in these problems, elitism is not incorporated; thus they do not always ensure better solutions in the child populations. In the beginning of the last decade, several elitist models of Pareto-based multiobjective evolutionary algorithms were proposed such as Strength Pareto Evolutionary Algorithm (SPEA) [Zitzler and Thiele 1998] and SPEA2 [Zitzler et al. 2001], Pareto Archived Evolutionary Strategy (PAES) [Knowles and Corne 1999], Pareto Envelope-Based Selection Algorithm (PESA) [Corne et al. 2000] and PESA-II [Corne et al. 2001], and Nondominated Sorting Genetic Algorithm (NSGA-II) [Deb et al. 2002]. Most of the recent applications of multiobjective evolutionary algorithms for clustering problems have exploited the underlying search strategy of one of these Pareto-based elitist approaches. Another recent approach regarding the design of multiobjective evolutionary algorithms is to employ a selection mechanism depending on a performance measure. For example, the Indicator-Based Evolutionary Algorithm (IBEA) [Zitzler and Künzli 2004] is planned to be adapted to the user's choices by formalizing such choices in terms of continuous generalizations of the dominance relation. Since then, a few other indicator-based techniques like the S Metric Selection Evolutionary Multiobjective Optimization Algorithm (SMS-EMOA) [Beume et al. 2007] (which is based on the hypervolume [Zitzler 1999]) have also been developed. The most important advantage of indicator-based algorithms such as SMS-EMOA is that they appear to scale better when there are many objectives (four or more). However, the methods that depend on the hypervolume are very computationally expensive. Since no clustering algorithms are found that depend on an indicator-based multiobjective evolutionary algorithm, these approaches are not discussed further here, and they are cited only for the sake of completeness. In addition to the genetic algorithms, other metaheuristic algorithms for multiobjective optimization include differential evolution-based multiobjective optimization (DEMO) [Robic and Filipic 2005], multiobjective optimization using differential evolution (MODE) [Xue et al. 2005], Pareto differential evolution (PDE) [Abbass and Sarker 2002], nondominated sorting differential evolution (NSDE) [Iorio and Li 2004], and Archived Multiobjective Simulated Annealing (AMOSA) [Bandyopadhyay et al. 2008].

2.2. General Framework of Multiobjective Evolutionary Clustering

Before discussing the general framework for multiobjective evolutionary algorithms for clustering, we first discuss the basic framework for generic single-objective evolutionary algorithms for clustering. Any evolutionary algorithm for clustering has to handle some issues. These include the following:

—Solution representation or chromosome encoding
—Choice of objective function
—Designing the evolutionary operators such as selection, crossover, and mutation

Note that all three components are also part of any multiobjective evolutionary strategy. However, in MOO, some additional issues are to be taken resolved. For example, for the multiobjective case, the suitable choice of the objective functions (more than one) is not a trivial task, as numerous possible combinations may exist. The selection strategy should also be chosen carefully; most of the time, this depends on the underlying MOO tool. Another important objective is to design a technique for keeping track of nondominated solutions. Finally, as MOO yields a set of nondominated clustering solutions, a suitable strategy must be devised for obtaining the final clustering solution from the final nondominated solution set. Therefore, as a whole, any multiobjective evolutionary clustering algorithm has to do of the following:

(1) Choose a possible encoding of chromosome to represent a clustering solution and generate the initial population of chromosomes.
(2) Choose a suitable set of objective functions from the cluster validity indices that are to be optimized simultaneously.
(3) Design suitable evolutionary operators such as selection, crossover, and mutation to manipulate the chromosomes that encode clustering solutions.
(4) Use an effective population maintenance strategy for keeping track of the current nondominated front.
(5) Develop a technique to obtain a single clustering solution from the set of solutions in the final nondominated near Pareto-optimal set.

The existing evolutionary multiobjective clustering algorithms mainly differ in these five points. Algorithm 1 shows the pseudocode of a generic multiobjective evolutionary algorithm for data clustering. Some of these steps, such as encoding strategy, crossover operation, and mutation operation, as mentioned earlier, are needed for evolutionary clustering algorithms in general. The other steps, such as choosing the set of objective functions, selection operation, storing current nondominated front, and obtaining final solution from the nondominated front, are directly related to multiobjective clustering. However, for the sake of completeness, we have considered all five steps for surveying the multiobjective evolutionary clustering algorithms. In the next five sections, we have made a study of different approaches for dealing with these five important issues for solving multiobjective evolutionary clustering problem.

ALGORITHM 1: Pseudocode for a Generic Multiobjective Evolutionary Algorithm for Clustering

```
1:  Input: Dataset, Algorithm parameters.
2:  Output: Clustering of dataset
3:  $i = 0$;
4:  $P_i \leftarrow$ Initialize_Population(Dataset,Algorithm parameters);
5:  $Fit_i \leftarrow$ Evaluate_Population($P_i$);
6:  while Termination Criterion is not met do
7:      $i = i + 1$;
8:      $Mat_i \leftarrow$ Select($P_{i-1}, Fit_{i-1}$);
9:      $Cross_i \leftarrow$ Crossover($Mat_i$);
10:     $Mut_i \leftarrow$ Mutation($Cross_i$);
11:     $Fit_i \leftarrow$ Evaluate_Population($Mut_i$);
12:     $P_i \leftarrow$ Update_Population($P_{i-1}, Fit_{i-1}, Mut_i, Fit_i$);
13: end while
14: $C \leftarrow$ Select_From_Pareto_Front($P_i$);
15: return $C$;
```

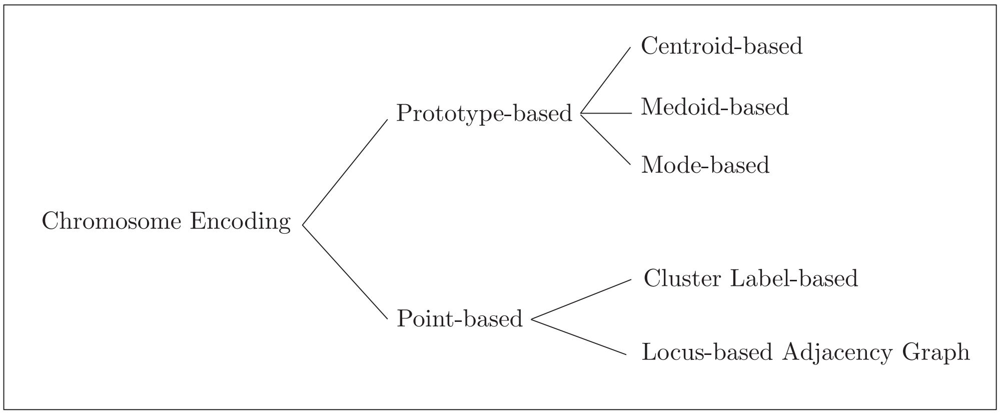

Fig. 1. Classification of chromosome encoding schemes for different multiobjective evolutionary clustering algorithms.

3. SOLUTION REPRESENTATION TECHNIQUES

To solve a clustering problem using multiobjective evolutionary algorithms, one has to first encode a number of possible clustering solutions into a string or chromosome. The approaches for encoding a clustering solution as a string can broadly be grouped into two major categories: prototype-based approach and point-based approach [Mukhopadhyay et al. 2014b]. In prototype-based approaches, a chromosome encodes cluster representatives or prototypes, such as cluster centroids, medoids, and modes. On the contrary, in a point-based approach, a complete clustering solution, that is, class assignments of the points, is encoded in the chromosomes. Figure 1 shows the broad classification of different encoding techniques used in various multiobjective clustering algorithms. The details of these approaches and their variants are described in the next sections.

3.1. Prototype-Based Encoding

The first genetic algorithm-based approach for selecting cluster prototypes was proposed in Kuncheva and Bezdek [1997], who employed a binary encoding of chromosomes of length equal to the number of points, where the points having value 1 are selected in the prototype set. A variation of this encoding was made in Maulik and Bandyopadhyay [2000], who, for the first time, used real-valued chromosome to encode the cluster prototypes directly. In prototype-based encoding, cluster prototypes, such as centroids, medoids, or modes, are encoded in the chromosome. Here, we describe in detail each type of encoding scheme used in multiobjective clustering.

3.1.1. Centroid-Based Encoding. In this approach, the chromosomes are comprised of real numbers that represent the coordinates of the cluster centers. Let us assume that a chromosome encodes the K cluster centers in d-dimensional space. Its length l will then be $d \times K$. For example, in two-dimensional space, the chromosome

$$\langle 2.4\ 5.9\ 0.36\ 2.7\ 5.3\ 10.2 \rangle$$

encodes three cluster centers, (2.4, 5.9), (0.36, 2.7), and (5.3, 10.2). Cluster centers are considered to be atomic. In the case of multiobjective clustering, this encoding scheme was first utilized by Mukhopadhyay et al. [2006], who have used this encoding policy in a series of works after that [Bandyopadhyay et al. 2007a, 2007b; Maulik et al. 2009; Mukhopadhyay and Maulik 2009]. While decoding a chromosome, they have also incorporated a local improvement stage by updating the chromosome by taking the mean of the points assigned to the clusters, as done in K-means [Jain and Dubes 1988] or fuzzy C-means [Bezdek 1981]. In addition to Maulik et al. [2009], this encoding technique has been used by other researchers [Kundu et al. 2009; Kirkland et al. 2011; Zhu et al. 2012; Chen and Wang 2005; Caballero et al. 2006; Ripon et al. 2006a, 2006b; Won et al. 2008].

The main advantage of using this encoding scheme is that the chromosomes generated by it are usually not very long, since in general the number of features d and the number of clusters K are much smaller than the number of points n. As the chromosome lengths are smaller, it takes less time to apply different genetic operators, such as crossover and mutation, on these chromosomes, which in turn results in quicker convergence of the algorithm [Mukhopadhyay et al. 2009a; Maulik et al. 2011]. Furthermore, encoding the centroids helps identify overlapping clusters well, and is particularly suitable for incorporating fuzziness in the clustering method. However, one important problem with this type of encoding is that, although it encodes the round-shaped clusters well, it cannot capture the clusters with nonconvex or inconsistent shapes. Also, if the chromosomes encode different number of clusters [Mukhopadhyay and Maulik 2011], the chromosomes will be of variable lengths, which may create complexity in designing the evolutionary operators. Furthermore, in this type encoding, if the dataset has a large number of attributes (large value of d), then it will result in longer chromosomes. Thus, for higher-dimensional datasets, this encoding strategy should be avoided.

3.1.2. Medoid-Based Encoding. When the datasets are categorical in nature, or the coordinates of the actual data points are not known, it is not possible to find the centers of the clusters. In that case, it is more convenient to use an actual point that is most centrally located in a cluster to represent that cluster. These points are called cluster medoids. A cluster medoid is that point of the cluster from which the sum of the distances to the other points of the cluster is minimum. There are few approaches of multiobjective evolutionary clustering in which the cluster medoids, or the indices of the points representing the cluster medoids, are encoded in the chromosomes. The length of each chromosome is the same as the number of clusters K. Each position of the chromosome can have an integer value from the set $\{1, 2, \ldots, n\}$, n being the number of data points. Thus a chromosome is basically a vector of point indices. If a point index is present in a chromosome, it signifies that the corresponding point is a cluster medoid. It may be noted that a valid chromosome cannot contain a point index more than once [Mukhopadhyay and Maulik 2007]. A modified medoid-based encoding has been adopted in Ripon and Siddique [2009]. Here, instead of integer encoding, a binary encoding is used. The chromosome length is taken as equal to the number of points. In a binary string, the medoid positions are indicated by bit 1; the other positions are indicated as bit 0. The advantage of medoid-based encoding is that, unlike centroids, medoids do not suffer from the presence of outliers. Moreover, medoids can be computed without the information of the coordinates of the points. The distance matrix among the points alone is sufficient for medoid calculation. However, given a cluster, finding the medoid is computationally expensive if the distance matrix is not precalculated, which is very likely when the value of n is reasonably large.

3.1.3. Mode-Based Encoding. In some of the multiobjective clustering techniques for categorical data [Mukhopadhyay et al. 2009a, 2014b], in place of cluster medoids, cluster modes have been utilized to represent the clusters. Cluster modes are useful for the datasets with categorical features for which one cannot compute the mean (centroid) of a cluster. The mode of a cluster containing categorical points is defined as a vector of the attributes for which each component of the vector corresponds to the most frequent value occurring under the respective attribute over all the points of the cluster. In mode-based chromosome representation, each chromosome consists of feature values representing K cluster modes. If each object is described in terms of d categorical attributes $\{A_1, A_2, \ldots, A_d\}$, the length of a chromosome will be $K \times d$. Here, the first d positions (or genes) correspond to the d attribute values of the first cluster mode, the second d positions correspond to that of the second cluster mode, and so on. For example, if $d = 2$ and $K = 3$, then the chromosome

$$\langle c_{11}\ c_{12}\ c_{21}\ c_{22}\ c_{31}\ c_{32} \rangle$$

represents three cluster modes (c_{11}, c_{12}), (c_{21}, c_{22}), and (c_{31}, c_{32}). Here, c_{ij} is the value of the jth attribute in the ith cluster mode, and $c_{ij} \in DOM(A_j)$, $1 \leq i \leq K$, $1 \leq j \leq d$. Cluster modes are also susceptible to noise or outliers. However, computation of cluster modes is much faster than that of cluster medoids.

In the cluster prototype-based encoding of chromosomes, the length of the chromosome depends on the number of cluster prototypes it is encoding. If the number of clusters K is fixed, then the length of all the chromosomes is the same [Bandyopadhyay et al. 2007a]. If the number of clusters is not known, then one can use variable-length chromosomes for which each chromosome can encode a different number of clusters [Mukhopadhyay et al. 2009; Mukhopadhyay and Maulik 2011; Chen and Wang 2005; Ripon et al. 2006b; Won et al. 2008; Kirkland et al. 2011]. For this case, special crossover and mutation operations are designed to handle variable chromosome lengths.

3.2. Point-Based Encoding

Another popular approach for encoding is point-based encoding, in which the complete clustering of the data points are encoded instead of only the representatives/prototypes of the clusters. There are mainly two approaches under a point-based encoding scheme. These are described here.

3.2.1. Cluster Label-Based Encoding. This solution representation approach is the most common form of point-based encoding. In this representation, the length of a chromosome is equal to the number of points in the dataset, and each position denotes the cluster label of the respective point. Thus, if position i of the chromosome is assigned a value k, then the ith data point is put in cluster k. As is evident, a chromosome can be composed of only integer values drawn from the set $\{1, 2, \ldots, K\}$, where K is the maximum number of clusters. This encoding approach has been adopted in different multiobjective cluster works [Handl and Knowles 2004; Özyer et al. 2004; Liu et al. 2005; Demir et al. 2010; Praditwong et al. 2011; Özyer et al. 2011].

The main advantage of this type of encoding is that the decoding step is straightforward and does not need any extra effort, since the data points are already given the cluster labels in the encoding. Also, this encoding approach can capture clusters with arbitrary shapes if evolved with suitable objective functions. This encoding strategy is suitable for any data types, such as continuous and categorical. Also, the length of the chromosomes is fixed and is independent of the number of clusters encoded in the chromosome. However, there are several disadvantages of this encoding policy. First, the chromosome length is the same as the number of points. Therefore, for larger datasets, its convergence becomes slower. This is because the chromosomes, and thus the search space, in such cases become bigger [Mukhopadhyay et al. 2009a]. Another problem with this type of encoding is that it can produce a large number of redundant solutions. For example, the chromosomes 1 1 1 2 2 3 3 3 and 2 2 2 3 3 1 1 1 refer to the same clustering solution with three clusters. Moreover, while doing crossover/mutation, as the cluster i of one solution does not always correspond to the cluster i of another solution, the clustering structure may get damaged. Finally, this encoding policy does not support fuzzy clustering directly, since each data point is assigned to exactly one cluster. Thus, it may not be suitable for detecting overlapping clusters.

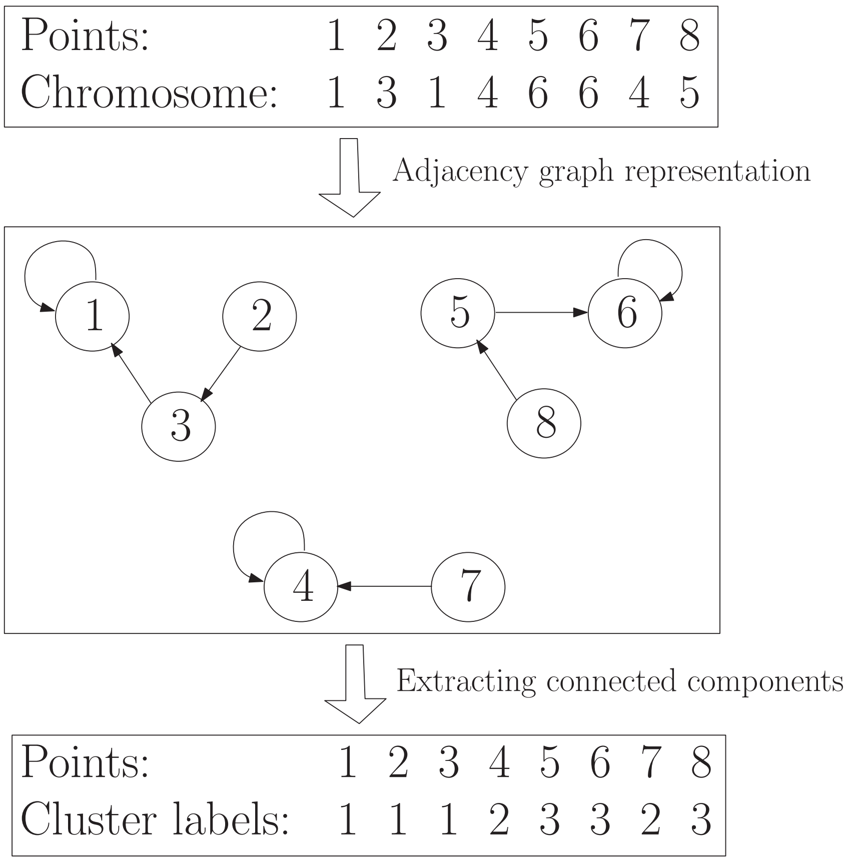
Points: 1 2 3 4 5 6 7 8
Chromosome: 1 3 1 4 6 6 4 5
Adjacency graph representation
1
2
3
4
5
6
7
8
Extracting connected components
Points: 1 2 3 4 5 6 7 8
Cluster labels: 1 1 1 2 3 3 2 3

Fig. 2. Locus-based adjacency graph representation of the chromosomes.

3.2.2. Locus-Based Adjacency Graph Encoding. Handl and Knowles [2005a, 2005c, 2007], proposed a variant of the cluster label-based encoding strategy. In this representation, a chromosome contains n positions (n is the number of data points), each having an integer value from the set $\{1, \ldots, n\}$. If position i contains an integer value j, it is interpreted as a link between the data points i and j. In the corresponding clustering solution, these two points will be assigned to the same cluster. In this way, each chromosome corresponds to a graph that contains a vertex for each data point, and the links between two data points represent the edges. Therefore, decoding of a chromosome requires identification of all the connected components of the graph; this can be performed in linear time [Handl and Knowles 2007]. The data points contained in the same connected component will then belong to the same cluster. The number of connected components will denote the number of cluster. Thus, in this strategy a chromosome encodes both the clustering and the number of clusters. Figure 2 illustrates this encoding scheme. In addition to Handl and Knowles, many other researchers have employed this representation technique [Demir et al. 2007; Matake et al. 2007; Qian et al. 2008; Shirakawa and Nagao 2009; Folino and Pizzuti 2010; Demir et al. 2010; Kim et al. 2010].

The main advantage of this type of encoding is that here the number of clusters is automatically encoded in the chromosome itself. Also, the chance of generating redundant chromosomes, as in label-based encoding, is much smaller. Moreover, this type of encoding can capture arbitrarily shaped clusters. However, this encoding policy suffers from some disadvantages similar to that discussed for label-based encoding. For example, here also the chromosome length is equal to the number of data points and hence large for large datasets. However, it can be managed by using guided evolutionary operators, as in Handl and Knowles [2007], which results in faster convergence. Another disadvantage is that this encoding does not support incorporation of fuzziness in the clustering process directly.

3.3. Effect of Solution Representation on Objective Functions

As mentioned before, solution representation or encoding techniques do not have a direct relationship with; rather, it is an issue in evolutionary algorithms for clustering in general. However, the choice of solution representation techniques has a major effect on the computation of the objective functions (validity indices). In general, there are two major classes of cluster validity indices that are used as the objective functions to be optimized in MOO problems. These are prototype-based and cluster label-based validity indices. Sections 4.1 and 4.2 describe some of the popularly used validity measures in these two categories, respectively. The validity indices that are based on cluster prototypes are most suitable if they are used for prototype-based chromosome representation [Bandyopadhyay et al. 2007a]. As these validity indices are based on cluster prototypes, therefore, if the chromosomes encode the cluster prototypes, no extra decoding step is necessary. The prototypes represented in the chromosomes can directly be used to compute the validity index values [Bandyopadhyay et al. 2007a]. Similarly, the point-based encoding strategies are suitable for the computation of the label-based validity indices, since it does not need the computation of cluster prototypes [Handl and Knowles 2007]. On the contrary, if prototype-based encoding is used for computation of a label-based validity index, or point-based representation is employed for computing a prototype-based validity measure, some additional decoding step will be necessary. In many of the multiobjective evolutionary algorithms for clustering, multiple validity indices from different categories are optimized. Details of these combinations can be found in the next section. It may be noted that if the validity indices to be optimized are taken from different categories, then the additional decoding step will be necessary for computation of one of the objectives, since the algorithm can use either prototype-based or point-based chromosome representation only.

3.4. Population Initialization

Initialization of the population is the first task for an evolutionary algorithm as depicted in line 4 of Algorithm 1. In most of the multiobjective evolutionary clustering algorithms, the chromosomes in the initial population have been generated randomly. In prototype-based chromosome representation, the prototypes in the initial chromosomes are usually some randomly selected data points. On the other hand, for point-based encoding, the cluster label vectors have been initialized with random strings so that each point gets a random cluster label. In a few algorithms, some specialized initialization has been utilized. For example, in the VIENNA algorithm [Handl and Knowles 2004], the chromosomes have been initialized using Voronoi diagrams. In Handl and Knowles [2007], Qian et al. [2008], and Shirakawa and Nagao [2009], the chromosomes have been initialized using minimum spanning trees and K-means clustering algorithms. A minimum spanning tree is built using all the data points and then a number of uninteresting links are removed to create each chromosome. Also, some chromosomes are created based on some runs of K-means clustering algorithm. The primary objective of this type of initialization is to build a better initial population than random, because here the chromosomes already encode some potentially good clustering solutions, which will be refined in subsequent generations. Thus it is expected that as the algorithm starts from a better position, the chance of quick convergence increases. However, one basic disadvantage of this approach is that the initialization itself takes a longer amount of time than the actual algorithm, compared to those for which random initialization has been used.

4. CHOICE OF OBJECTIVE FUNCTIONS

One of the important aspects of multiobjective evolutionary clustering is the choice of suitable objective functions that are to be optimized simultaneously (lines 5 and 11 of Algorithm 1). Generally, cluster validity indices [Maulik and Bandyopadhyay 2002] are used as the objective functions. Several validity indices in different combinations have been used in various multiobjective evolutionary clustering algorithms. In this section, we discuss them in detail. First, we describe the validity indices, then discuss how they are used in different algorithms. We have classified the validity indices in two categories: indices based on cluster prototypes and indices based on cluster labels. Note that here we describe only those indices that have been used as objective functions in multiobjective clustering algorithms. Finally, the importance of the right choice of objective functions is discussed. The following are definitions for symbols in the cluster validity indices:

n - number of data points
d - number of attributes/features
K - number of clusters
m - fuzzy exponent
x_i - ith data point
z_k - kth cluster prototype
u_{ki} - membership degree of the ith data point to the kth cluster
$D(.,.)$ - distance function
C - set of all clusters
C_k - kth cluster

These symbols are not defined any further in different equations. Only the notations not mentioned here are defined.

4.1. Validity Indices Based on Cluster Prototypes

The validity indices based on cluster prototypes (centroids, medoids, and modes), both crisp and fuzzy, have been heavily used as objective functions in multiobjective evolutionary clustering algorithms. We describe these indices here. Although we have used the term cluster centers, cluster medoids or cluster modes could be used as well.

4.1.1. J_m Index. The J_m [Bezdek 1981] index is minimized by fuzzy C-means clustering. It is defined as follows:

$$J_m = \sum_{k=1}^{K} \sum_{i=1}^{n} u_{ki}^m D^2(z_k, x_i), \tag{4}$$

where $D(z_k, x_i)$ is the distance of the ith data point x_i from the kth cluster center z_k. J_m represents the global fuzzy cluster variance. The smaller value of J_m corresponds to more compact clusters. However, J_m value depends on the number of clusters K. That is, J_m value gradually decreases with the increase in K value. Note that J_m takes its minimum value 0 for $K = n$ [Maulik et al. 2011].

The normalized version of the J_m index, denoted by $\mathcal{J}_m$ is defined as [Mukhopadhyay et al. 2009a]:

$$\mathcal{J}_m = \sum_{k=1}^{K} \frac{\sum_{i=1}^{n} u_{ki}^m D^2(z_k, x_i)}{\sum_{i=1}^{n} u_{ki}}. \tag{5}$$

This is also to be minimized for getting more compact clusters.

4.1.2. Overall Cluster Deviation. The overall cluster deviation ($Dev(C)$) [Handl and Knowles 2007] is the crisp version of the J_m index. This is computed as the overall summed distances between data points and their corresponding cluster center:

$$Dev(C) = \sum_{C_k \in C} \sum_{x_i \in C_k} D(z_k, x_i). \tag{6}$$

Overall deviation must be minimized in order to obtain compact clusters. This objective is practically the same as the objective of K-means clustering.

4.1.3. Davies-Bouldin Index. The Davies-Bouldin (DB) index [Davies and Bouldin 1979] is a function of the ratio of the sum of within-cluster scatter to between-cluster separation. The scatter within the ith cluster, S_i, is computed as

$$S_i = \frac{1}{|C_i|} \sum_{x \in C_i} D^2(z_i, x). \tag{7}$$

Here $|C_i|$ denotes the number of data points belonging to cluster C_i. The distance between two clusters C_i and C_j, d_{ij} is defined as the distance between the centers:

$$d_{ij} = D^2(z_i, z_j). \tag{8}$$

The DB index is then defined as

$$DB = \frac{1}{K} \sum_{i=1}^{K} R_i, \tag{9}$$

where

$$R_i = \max_{j, j \neq i} \left\{ \frac{S_i + S_j}{d_{ij}} \right\}. \tag{10}$$

The value of the DB index must be minimized in order to achieve proper clustering.

4.1.4. Xie-Beni Index. The Xie-Beni (XB) index [Xie and Beni 1991] is defined as a function of the ratio of the total fuzzy cluster variance σ to the minimum separation sep of the clusters. Here σ and sep can be written as

$$\sigma = \sum_{k=1}^{K} \sum_{i=1}^{n} u_{ki}^2 D^2(z_k, x_i) \tag{11}$$

and

$$sep = \min_{k \neq l} \{D^2(z_k, z_l)\}. \tag{12}$$

The XB index is then written as

$$XB = \frac{\sigma}{n \times sep} = \frac{\sum_{k=1}^{K} \sum_{i=1}^{n} u_{ki}^2 D^2(z_k, x_i)}{n \times (\min_{k \neq l} \{D^2(z_k, z_l)\})}. \tag{13}$$

It may be noted that for compact and well-separated clustering, the value of σ should be small while *sep* should be large, thereby giving smaller values of the *XB* index. Therefore the objective is to minimize the *XB* index for achieving proper clustering [Maulik et al. 2011].

4.1.5. Intracluster Entropy. Intracluster entropy (H) [Ripon et al. 2006a] is used to measure the average purity of clusters without using the class labels of data objects. This is defined as

$$H = \sum_{i=1}^{K} \left[(1 - H(C_i).g(z_i))\right]^{\frac{1}{k}}, \tag{14}$$

where

$$H(C_i) = -[g(z_i)log_2 g(z_i) + (1 - g(z_i))log_2(1 - g(z_i))]. \tag{15}$$

The average similarity $g(z_i)$ between a cluster center z_i and the data points belonging to cluster i is defined as

$$g(z_i) = \frac{1}{n}\sum_{j=1}^{n}\left[0.5 + \frac{CO(z_i, x_j)}{2}\right], \tag{16}$$

where CO is the cosine distance defined as

$$CO(x_i, x_j) = \frac{\sum_{k=1}^{d} x_{ik}.x_{jk}}{\sqrt{\sum_{k=1}^{d} x_{ik}^2}.\sqrt{\sum_{k=1}^{d} x_{jk}^2}}. \tag{17}$$

The index H has to be maximized to obtain homogeneous clusters.

4.1.6. Cluster Separation. Cluster separation *Sep* or intercluster distance [Ripon et al. 2006a] is computed as the average distance among the cluster centers, that is,

$$Sep(C) = \frac{2}{K(K-1)}\sum_{i=1}^{K}\sum_{j=1, j\neq i}^{K} D^2(z_i, z_j). \tag{18}$$

$Sep(C)$ must be maximized to obtain well-separated clusters.

The fuzzy version of cluster separation is also available. The fuzzy separation $\mathcal{S}$ is computed as follows [Mukhopadhyay et al. 2009a; Mukhopadhyay and Maulik 2011]: the center z_i of the i^{th} cluster is assumed to be the center of a fuzzy set $\{z_j | 1 \leq j \leq K, j \neq i\}$. Thus the membership degree of each z_j to z_i, $j \neq i$ is computed as:

$$\mu_{ij} = \frac{1}{\sum_{l=1, l\neq j}^{K}\left(\frac{D(z_j, z_i)}{D(z_j, z_l)}\right)^{\frac{2}{m-1}}}, \quad i \neq j. \tag{19}$$

Subsequently, the fuzzy separation is defined as:

$$\mathcal{S} = \sum_{i=1}^{K}\sum_{j=1, j\neq i}^{K} \mu_{ij}^m D^2(z_i, z_j). \tag{20}$$

The fuzzy separation $\mathcal{S}$ is also to be maximized in order to obtain well-separated clusters.

4.1.7. Average Between Group Sum of Squares (ABGSS). In Kirkland et al. [2011], a variant of the cluster separation measure, called Average Between Group Sum of Squares (ABGSS) is proposed. This computed the average distance of the cluster centers from the centroid of the dataset as follows:

$$ABGSS = \frac{\sum_{i=1}^{K} n_i . D^2(z_i, \overline{z})}{K}. \tag{21}$$

Here, $\overline{z}$ denotes the center of the dataset, that is, the mean of all the points, and n_i denotes the number of points in cluster i. The objective is to maximize ABGSS to obtain well-separated clusters.

4.1.8. $\mathcal{I}$ Index. The $\mathcal{I}$ index [Maulik and Bandyopadhyay 2002; Pakhira et al. 2004] is defined as follows:

$$\mathcal{I} = \left(\frac{1}{K} \times \frac{E_1}{E_K} \times D_K\right)^{\gamma}, \tag{22}$$

where

$$E_K = \sum_{k=1}^{K} \sum_{j=1}^{n} u_{kj} D(z_k, x_j) \tag{23}$$

and

$$D_K = \max_{i,j=1}^{K} \{D(z_i, z_j)\}. \tag{24}$$

$\mathcal{I}$ index is composed of three factors: $\frac{1}{k}$, $\frac{E_1}{E_K}$, and D_K. The first factor tries to decrease $\mathcal{I}$ as K increases. The second factor is the ratio of E_1 to E_K. Where E_1 is a constant for a given dataset, E_K gets reduced with an increase in K. Therefore the $\mathcal{I}$ index value increases with the increase in E_K. This, in turn, encourages formation of more compact clusters. Finally, the third factor, D_K, which represents the highest separation between any pair of clusters, will increase with the value of K. However, it should be noted that D_K cannot be more than the maximum separation between two points in the dataset. Hence, these three factors critically balance each other through contention. The contrast between the different cluster configurations is controlled by the power γ. A larger value of index $\mathcal{I}$ implies better clustering [Maulik et al. 2011].

4.2. Validity Indices Based on Cluster Labels

Although most of the cluster validity indices used for multiobjective clustering are based on the distances computed among the cluster prototypes, there are some indices that do not use cluster prototypes for computation; rather, they are based on cluster label vectors. Here, we discuss such indices.

4.2.1. Dunn Index. If $\delta(C_i, C_j)$ represents the distance between clusters C_i and C_j, and $\Delta(C_i)$ represents the diameter of cluster C_i, then the Dunn family of indices takes the following form [Dunn 1974]:

$$DN = \min_{1 \le i \le K} \left\{ \min_{1 \le j \le K, j \ne i} \left\{ \frac{\delta(C_i, C_j)}{\max_{1 \le k \le K}\{\Delta(C_k)\}} \right\} \right\}. \tag{25}$$

Originally, Dunn used the following forms of δ and Δ:

$$\delta(C_i, C_j) = \min_{x_i \in C_i, x_j \in C_j} \{D(x_i, x_j)\} \tag{26}$$

and

$$\Delta(C_i) = \max_{x_i, x_k \in C_i} \{D(x_i, x_k)\}. \tag{27}$$

A larger value of Dunn index corresponds to compact and well-separated clusters. The goal is therefore to maximize the Dunn index [Maulik et al. 2011].

4.2.2. Cluster Connectedness. Cluster connectedness ($Conn(C)$) represents the connectedness of the clusters and is defined as [Handl and Knowles 2007]

$$Conn(C) = \sum_{i=1}^{n} \left(\sum_{j=1}^{L} \kappa_{i,nn_{ij}} \right), \tag{28}$$

where $\kappa_{i,nn_{ij}} = \frac{1}{j}$ if $\nexists C_k : i \in C_k \wedge nn_{ij} \in C_k$, and $\kappa_{i,nn_{ij}} = 0$ otherwise. Here nn_{ij} is the jth nearest neighbor of data point i. L, a user-defined parameter, decides the number of neighbors contributing to the connectedness measure. In order to facilitate quick computation of the connectedness index, one should precompute the complete list of nearest neighbors for each data point. The objective is to minimize cluster connectedness for obtaining highly connected clusters [Maulik et al. 2011].

4.2.3. Edge Index. The Edge index [Shirakawa and Nagao 2009] evaluates the overall summed distances on boundaries between the clusters. This index is a measure of the difference in the boundary between the clusters. This is defined as

$$Edge(C) = -\sum_{i=1}^{n} \sum_{j \in \mathbb{F}_i} \xi_{i,j}, \tag{29}$$

where $\xi_{i,i} = D(x_i, x_j)$ if $\nexists C_k : i \in C_k \wedge j \in C_k$, and $\xi_{i,j} = 0$ otherwise. Here, $\mathbb{F}_i$ is the set of N nearest neighbors of ith point. N is a user-defined integer number. The $Edge(C)$ index must be minimized to obtain well-separated clustering.

4.2.4. Silhouette Index. Let a_i be the mean distance of a point x_i from the other points of the cluster to which x_i belongs, and let b_i be the minimum of the mean distances of x_i from the points of the other clusters. The silhouette width s_i of the point can then be defined as [Rousseeuw 1987]

$$s_i = \frac{b_i - a_i}{max\{a_i, b_i\}}. \tag{30}$$

Silhouette index $\mathcal{S}$ [Rousseeuw 1987] is the mean silhouette width of all the data points, that is,

$$\mathcal{S} = \frac{1}{n} \sum_{i=1}^{n} s_i. \tag{31}$$

The silhouette index score lies between -1 and 1; a higher value corresponds to a better clustering result [Maulik et al. 2011].

4.2.5. Min-Max Cut. The Min-Max cut index [Demir et al. 2010] aims to minimize the distances among the points within a cluster and maximize the distances of the points from different clusters. This is defined as

$$MinMaxCut(C) = \sum_{i=1}^{K} \frac{\sum_{j \in C_i} \sum_{k \notin C_i} D(x_j, x_k)}{\sum_{j \in C_i} \sum_{k \in C_i} D(x_j, x_k)}. \tag{32}$$

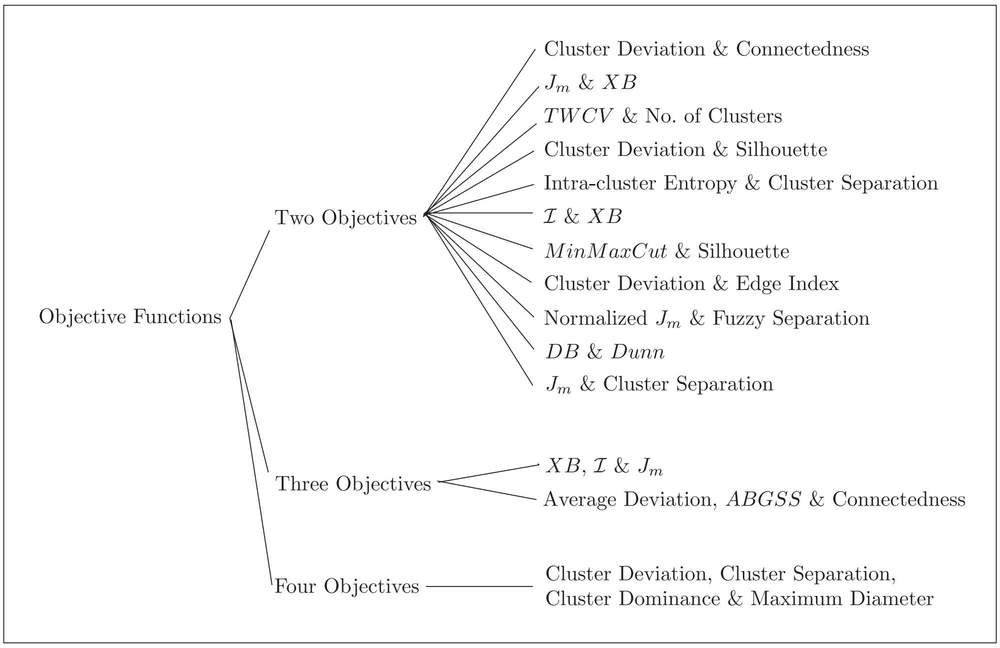
Objective Functions
Two Objectives
Cluster Deviation & Connectedness
J_m & XB
$TWCV$ & No. of Clusters
Cluster Deviation & Silhouette
Intra-cluster Entropy & Cluster Separation
$\mathcal{I}$ & XB
$MinMaxCut$ & Silhouette
Cluster Deviation & Edge Index
Normalized J_m & Fuzzy Separation
DB & $Dunn$
J_m & Cluster Separation
Three Objectives
XB, $\mathcal{I}$ & J_m
Average Deviation, $ABGSS$ & Connectedness
Four Objectives
Cluster Deviation, Cluster Separation, Cluster Dominance & Maximum Diameter

Fig. 3. Overview of different combinations of cluster validity indices used in different multiobjective evolutionary clustering algorithms.

Maximization of the numerator ensures that the points within a cluster are distant from the other points and minimization of the denominator ensures that the points within a cluster are close with each other. Thus, overall, the index must be maximized to obtain good-quality clusters.

4.2.6. Total Within-Cluster Variance. The Total Within-Cluster Variance (*TWCV*) [Liu et al. 2005] is defined as

$$TWCV = \sum_{i=1}^{n}\sum_{j=1}^{d} x_{ij}^2 - \sum_{k=1}^{K}\frac{1}{n_k}\sum_{j=1}^{d}\left(\sum_{x_i \in C_k} x_{ij}\right)^2. \tag{33}$$

Here, x_{ij} denotes the jth feature value of the ith data point, and n_k denotes the number of points in cluster C_k. The goal is to minimize *TWCV* to obtain compact clusters.

4.3. Use of Validity Indices as Objective Functions

Most of the multiobjective evolutionary clustering algorithms have used different combinations of two cluster validity indices to be optimized simultaneously. Figure 3 shows an overview of different combinations of validity indices used, to be optimized simultaneously in different multiobjective clustering algorithms.

In Handl and Knowles [2004, 2005a, 2007], the MOCK clustering algorithm optimizes two validity indices, overall cluster deviation ($Dev(C)$) and cluster connectedness ($Conn(C)$). Both the indices are minimized simultaneously during the optimization process. It is interesting to note that $Dev(C)$ is a prototype-based objective, whereas $Conn(C)$ is a cluster-label–based objective. The goal of the MOCK algorithm is to capture those clusters that are compact and the points within a cluster that are well-connected. As MOCK uses a locus-based adjacency encoding of chromosomes, it is necessary to compute the cluster centers before computing $Dev(C)$. Some other multiobjective clustering works also have used these two objectives [Chen and Wang 2005; Qian et al. 2008; Demir et al. 2010].

Bandyopadhyay et al. [2007a], Mukhopadhyay and Maulik [2009], and Maulik et al. [2009] used two validity indices, J_m and *XB*. J_m represents the global cluster variance, that is, it computes the within-cluster variance added up over all the clusters. A lower J_m score corresponds to a superior clustering solution. On the contrary, the *XB* index is composed of both global (numerator) and local (denominator) metrics. The numerator of the *XB* index is similar to J_m; the denominator has an additional factor corresponding to the separation between the two closest clusters. Therefore, the *XB* index can be minimized by minimizing the numerator (fuzzy variance) and by maximizing the cluster separation. Note that these two factors may fail to attain their best values for the same clustering, especially when the datasets have overlapping and complex clusters. These two validity indices have also been used in some other multiobjective clustering approaches by the authors [Mukhopadhyay et al. 2006; Bandyopadhyay et al. 2010].

There are various other combinations of objective functions/validity indices used in multiobjective evolutionary clustering. In Özyer et al. [2004], Liu et al. [2005], and Du et al. [2005], the two validity indices used for optimization are *TWCV* and the number of clusters *K*. Both the objectives, which are contradictory in nature, must be minimized simultaneously. In Mukhopadhyay and Maulik [2007], a multiobjective clustering algorithm for categorical data is proposed that employs overall deviation $Dev(C)$ (with respect to medoids rather than centroids) and silhouette index as the two objective functions. In Ripon et al. [2006a] and Ripon and Siddique [2009], the intra-cluster entropy *H* and cluster separation $Sep(C)$ have been optimized simultaneously. In Demir et al. [2007], Demir et al. [2010], and Kim et al. [2010], the Min-Max Cut and the silhouette index have been optimized as the two competing objectives. The two validity measures, overall deviation $Dev(C)$ and the Edge index $Edge(C)$, have been minimized simultaneously in Shirakawa and Nagao [2009] to obtain compact and well-separated clusters. The validity indices normalized J_m index ($\mathcal{J}$) and fuzzy cluster separation $\mathcal{S}$ have been minimized simultaneously in Mukhopadhyay et al. [2009a] and Mukhopadhyay and Maulik [2011]. In Mukhopadhyay et al. [2009a], in place of cluster centroids, cluster modes were used for computing the objective values, since this algorithm has been applied on categorical data. In Bandyopadhyay et al. [2010], along with several other combinations, *DB* and *Dunn* indices have also been chosen as the two objectives to be simultaneously optimized. In Mukhopadhyay et al. [2011], the two validity measures J_m and cluster separation have been optimized simultaneously.

All the multiobjective evolutionary clustering algorithms, as previouslly discussed, have optimized two cluster validity indices as the two objectives. However, a few multi-objective clustering techniques have also employed more than two objective functions. For example, in Mukhopadhyay et al. [2009b], three cluster validity indices, J_m, *XB* and $\mathcal{I}$, have been optimized simultaneously. In Kirkland et al. [2011] also, three objective functions, average cluster variance, ABGSS, and the cluster connectedness, have been employed. In Özyer et al. [2011], four validity indices have been utilized as the objective functions. They are overall cluster deviation, cluster separation, cluster dominance, and the diameter of the largest cluster, which are optimized simultaneously.

4.4. Discussion on the Selection of the Validity Indices

It is evident from the discussion in the previous section that a number of different combinations of cluster validity indices have been chosen as the objectives to be optimized simultaneously in various approaches for multiobjective evolutionary clustering algorithms. These validity indices can be classified based on their ability to capture particular properties of the clusters, such as compactness, separation, combination of compactness and separation, and cluster density. For example, the validity indices J_m, cluster deviation $Dev(C)$, intracluster entropy (H), and *TWCV* represent the compactness of the clusters, and their optimization leads to finding compact clusters. The validity indices cluster separation *Sep*, *ABGSS*, and $Edge(C)$ denote separation among the points of different clusters, thus their optimization results in well-separated clusters. There are some validity indices that combine both cluster compactness and separation. The validity indices *DB*, *XB*, $\mathcal{I}$, *Dunn*, *Silhouette*, and *MinMaxCut* belong to this category. These indices are optimized to obtain both compact and well-separated clusters. Another type of cluster validity measure is based on cluster density. Out of all the validity measures discussed here, only cluster connectedness $Conn(C)$ belongs to this class.

In general, it is a good idea that the set of objective functions in a multiobjective clustering method have representatives from multiple categories of validity indices. This is because, in that case, the algorithm will be able to optimize multiple characteristics of the evolved clusters. Most of the algorithms have followed this approach, as depicted in Figure 3. As can be seen from the figure, in two-objective algorithms, one of the combinations is the validity indices from compactness and separation categories (e.g., intracluster entropy and cluster separation, cluster deviation and $Edge(C)$, J_m and cluster separation, normalized J_m and fuzzy separation). In this type of algorithm, the evolved clusters are expected to be compact as well as separated from each other. In some two-objective algorithms, at least one or both of the objective functions belong to combined compactness-separation index (e.g., J_m and *XB*, cluster deviation and *Silhouette*, $\mathcal{I}$ and *XB*, *MinMaxCut* and *Silhouette*, *DB* and *Dunn*). Also, in these algorithms, optimization of both cluster compactness and separation has been taken care of. One usual problem in using compactness/separation-based objective functions is that they perform well for round-shaped and overlapping clusters; however, they do not perform well if the clusters have arbitrary shapes [Maulik et al. 2011]. On the other hand, in Handl and Knowles [2007] and many other works, the authors have used $Dev(C)$ and $Conn(C)$ as the two objective functions to be optimized simultaneously. $Dev(C)$ takes care of the compactness of the clusters, whereas $Conn(C)$ is expected to produce well-connected clusters. It may be noted that the algorithms that rely on optimization of these two objective functions can capture arbitrary shapes of clusters due to the presence of $Conn(C)$ criteria. However, there is not enough evidence showing good performance of these algorithms in the presence of overlapping clusters. The $Conn(C)$ objective may try to combine two overlapping clusters into one. Moreover, the cluster separation is not included in the objective functions. As mentioned before, there is very little effort to optimize more than two objective functions in multiobjective evolutionary clustering approaches. As is apparent from Figure 3, only two approaches combine three objective functions, and only one approach combines four objective functions. It seems that most researchers are not interested in optimizing more than two validity measures. This might be because of the fact that the performance of the popular multiobjective evolutionary algorithms such as NSGA-II, PAES, and PESA-II degrades with the increase of the number of objective functions [Saxena et al. 2013]. Second, if we use more objective functions, the objective function set might be redundant as more than one objective function will be chosen from each category (compactness/separation/density) of the validity indices. In fact, the choice of a suitable set of objective functions is not a trivial ; the clustering output may heavily depend on this choice [Handl and Knowles 2012]. Taking this into consideration, an interactive multiobjective clustering algorithm is proposed in Mukhopadhyay et al. [2013]. In this method, the algorithm is allowed to interact with a human expert (decision maker) during the optimization process to interactively learn the suitable set of objective functions for the input dataset, in addition to the clustering solution. However, we feel that some effort is needed to compare the performance of several combinations of validity measures in a systematic way to draw some specific conclusion on the choice of a good set of objective functions.

5. DESIGNING EVOLUTIONARY OPERATORS

In this section, we discuss the different evolutionary operators, such as selection, crossover, and mutation, that have been employed in various multiobjective evolutionary clustering algorithms.

5.1. Selection Operator

The use of the selection operator is for generating a mating pool of chromosomes (line 8 of Algorithm 1). In this operator, the multiobjective algorithms are fundamentally different from the traditional single-objective algorithms. Most of the multiobjective evolutionary clustering algorithms have used some standard multiobjective evolutionary tool as the underlying optimization framework. Therefore, the selection operators used in the multiobjective clustering algorithms are basically parts of the underlying MOO method they are using.

PESA-II [Corne et al. 2001] is a MOGA that has been used by Handl and Knowles [2005c, 2006, 2007] as the underlying optimization tool in their multiobjective clustering algorithms. In PESA-II, a region-based selection strategy is used. In this selection strategy, the objective space is divided into several hyperboxes using grids, and each solution is placed in a certain grid location based on its objective values. The unit of selection is a hyperbox rather than an individual. The procedure of selection is to choose (using any of the known selection methods) a hyperbox with a lesser number of solutions in it and then randomly select an individual within such a hyperbox. Other works have used a similar selection strategy [Qian et al. 2008; Shirakawa and Nagao 2009; Matake et al. 2007].

The series of multiobjective clustering algorithms proposed in Bandyopadhyay et al. [2007a], Maulik et al. [2009], Mukhopadhyay et al. [2009a], and others are mainly based on another popular MOO tool, Nondominated Sorting Genetic Algorithm-II (NSGA-II) [Deb et al. 2002]. In NSGA-II, a crowding distance-based binary tournament selection strategy is used. In this selection strategy, the parent and child populations are combined, then nondominated sorting [Deb et al. 2002] is applied to classify the solutions into different ranks based on their nondomination status. Thereafter, top P solutions (P being the population size) are selected using their ranks and crowding distance [Deb et al. 2002]. Crowding distance is a measure of the density of the region where a particular solution resides in the objective space. If two solutions are from two different ranks, then the better-rank solution is selected. Otherwise, if they are coming from the same rank, then the crowding distance measure is used to choose the solution coming from the less-crowded region in order to maintain diversity of the population. Crowding distance of a solution is computed by drawing a hyperbox around it, including its preceding and following solutions in the objective space, then computing the volume of the box. Larger crowding distance indicates that a solution is situated in a less-crowded region [Deb et al. 2002]. Many other researchers have used this selection strategy for their multiobjective clustering algorithms [Özyer et al. 2004; Chen and Wang 2005; Ripon et al. 2006a, 2006b; Ripon and Siddique 2009; Won et al. 2008; Liu et al. 2005; Kim et al. 2010; Folino and Pizzuti 2010; Özyer et al. 2011; Kirkland et al. 2011].

In Demir et al. [2007, 2010], instead of PESA-II or NSGA-II, another multiobjective evolutionary optimization algorithm, called Strength Pareto Evolutionary Algorithm-2 (SPEA2) has been used as the underlying optimization strategy for their graph-based sequence clustering (GraSC) algorithm. SPEA2 assigns fitness values to the solutions according to their dominance and density criteria. Similar to PESA-II, two populations are maintained. These are called working population and archive population, respectively. In every generation, the nondominated solutions in the working population and in the archive are copied into the next-generation archive. The fitness of each individual is calculated with respect to the strength of all the nondominated solutions that dominate it. After computing the fitness, a proportional selection algorithm is applied. The selection occurs from both the population and the archive.

Liu et al. [2005] used NPGA [Horn and Nafpliotis 1993] as the underlying multiobjective optimization tool. Note that NPGA is a nonelitist approach. In this approach, a Pareto dominance-based binary tournament selection with a sample of population is employed to find the better of two candidate solutions. Roughly 10 individual solutions are considered for finding dominance, and the nondominated solution is selected. If both individual solutions are found to be either dominated or nondominated, then the winner of the tournament is determined by fitness sharing.

The primary objective of a selection strategy is to ensure the wide spread of solutions in the nondominated set. A good selection scheme thus helps the clustering algorithm to capture different possible compromises among the objective functions yielding tradeoff clustering solutions in the nondominated front. In the Pareto-based approaches, the selection method mainly aims to compare two solutions based on their isolation from other solutions, and the solution situating in the more isolated region is deemed to be better. NPGA is an older approach compared to the other algorithms mentioned earlier, and it does not take into account the isolation of the solutions during selection. Also, it is already proven to be outperformed by the other methods [Zitzler et al. 2001; Corne et al. 2001; Deb et al. 2002]. In SPEA2 and NSGA-II, the unit of selection is an individual, that is, during selection, two individuals are compared based on their isolation status. On the other hand, in PESA-II, the unit of selection is hyperbox, and two hyperboxes in the objective space are compared in each turn. The hyperbox containing the smaller number of solutions is selected, then one random solution is selected from that hyperbox. In Corne et al. [2001], the region-based selection has been shown to perform better than the individual-based selection of PAES, SPEA, and PESA, and has also been shown to work faster than the crowded comparison method of NSGA-II. However, there are no direct comparisons among the different selection strategies under clustering context, although this could be an interesting study.

5.2. Crossover Operation

Crossover operation is used for exchanging genetic information among the chromosomes in the mating pool (line 9 of Algorithm 1). Various crossover operations have been used in different multiobjective clustering algorithms. We broadly classify these approaches according to the chromosome types on which they are applied. The broad classification is shown in Figure 4.

5.2.1. Crossover for Prototype-Based Encoding. For prototype-based encoding, most of the algorithms have used single-point crossover. When the chromosomes encode the cluster centroids [Bandyopadhyay et al. 2007a] or cluster modes [Mukhopadhyay et al. 2009a], single-point crossover operation has been adopted in such a way that the crossover point always falls between two cluster centroids/modes, that is, the crossover point is not allowed to fall within the different components of a cluster centroid/mode. This is done in order to avoid abrupt disruptions of the cluster centroids/modes. When cluster medoids are encoded, that is, indices of the points acting as cluster medoids are encoded in the chromosome, no such restriction is needed for the crossover points [Mukhopadhyay and Maulik 2007]. However, there is a chance of yielding invalid chromosomes after crossover since the same point-index may occur in a child chromosome more than once.

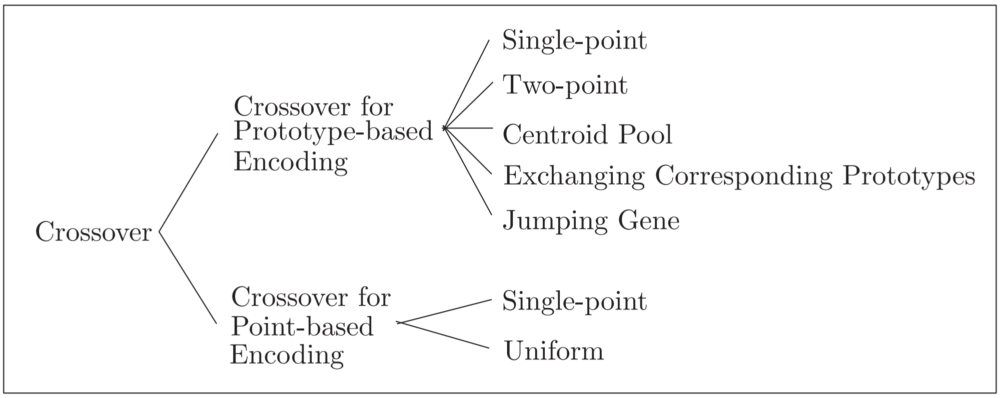
Crossover
Crossover for Prototype-based Encoding
Single-point
Two-point
Centroid Pool
Exchanging Corresponding Prototypes
Jumping Gene
Crossover for Point-based Encoding
Single-point
Uniform

Fig. 4. Classification of crossover schemes for different multiobjective evolutionary clustering algorithms.

Mukhopadhyay and Maulik [2007] handle this situation using a penalty functions approach, in which an invalid chromosome is given bad fitness values for all the objectives so that the invalid chromosomes automatically go out of the competition in subsequent generations.

In many of the centroid-based encoding techniques, the number of cluster centers encoded in the chromosome has been varied, that is, the different chromosomes may have different lengths [Mukhopadhyay and Maulik 2011; Won et al. 2008; Ripon et al. 2006a, 2006b]. To handle variable-length chromosomes, modified crossover operations are used. In Chen and Wang [2005], a modified two-point crossover has been used to handle variable-length chromosomes. Here, the cluster centers encoded in the chromosomes are arranged in ascending order of the values of the first coordinate. The range of the chromosomes, between which the exchange occurs between the parent chromosomes, are determined by the first cluster center values. If the two chromosomes encode a different number of clusters, then after crossover, the lengths of both chromosomes may change.

In another single-point crossover approach [Mukhopadhyay and Maulik 2011], while doing crossover between two parent chromosomes, two crossover points—one for each parent—are generated randomly. Thereafter, the child chromosomes are produced by exchanging the parts of the chromosomes beyond the crossover points. The cluster centers are again considered to be indivisible. This guarantees that the exchange of information happens in such a way that both offspring contain at least two cluster centers. For this, the operator is defined as follows [Mukhopadhyay et al. 2009]: Let two parents P_1 and P_2 consist of M_1 and M_2 number of cluster centers, respectively. Now, λ_1, the crossover point in P_1, is obtained as $\lambda_1 = rand()\%M_1$. Let the crossover point in P_2 be λ_2 varies between $\underline{\lambda_2}$ (lower bound) and $\overline{\lambda_2}$ (upper bound). $\underline{\lambda_2}$ and $\overline{\lambda_2}$ are given by [Maulik et al. 2008]:

$$\underline{\lambda_2} = \min[2; \max[0; 2 - (M_1 - \lambda_1)]], \quad \text{and} \tag{34}$$

$$\overline{\lambda_2} = [M_2 - max[0; (2 - \lambda_1)]]. \tag{35}$$

If $\overline{\lambda_2} \geq \underline{\lambda_2}$ then λ_2 is an integer between $\underline{\lambda_2}$ and $\overline{\lambda_2}$ generated randomly. Otherwise, $\lambda_2 = 0$. Note from these equations that if crossover points λ_1 and λ_2 are obtained according to these rules, then the number of clusters in both offspring solutions will not be less than 2.

In Won et al. [2008], a different crossover approach is used for handling variable-length chromosomes as follows: Let us assume that two parents P1 and P2 encode K_1 and K_2 cluster centroids, respectively. The crossover operator builds a pool of all the $K_1 + K_2$ centroids and divides them into two groups to obtain the recombined offspring chromosomes. The size of one group is chosen as a uniform random integer in $[max(2, K_1 + K_2 - K^*), min(K^*, K_1 + K_2 - 2)]$, where K^* is the maximum number of clusters allowed. Hence, the offspring chromosomes represent centroids more than one but no greater than K^*.

Another approach for crossover with variable-length chromosomes is employed in Kirkland et al. [2011]. First, it finds the largest cluster (containing maximum number of points) of the smallest chromosome. Then for all the points in that cluster, the clusters to which these points belong in the larger chromosome are obtained. Thereafter, the center of the largest cluster of the smaller chromosome is exchanged with the corresponding set of cluster centers in the larger chromosome. The resulting crossover is therefore an exchange of one cluster in one solution with the corresponding smaller clusters in the other solution.

In Ripon et al. [2006a, 2006b], a jumping gene operator is used for crossover purposes. In this operation, two chromosomes are chosen randomly from the mating pool as parents. Thereafter, for each chromosome, the first feature value of an arbitrary cluster center is picked up. A random number of consecutive features are then chosen from any of the parent chromosomes (donor). Subsequently, the chosen features are cut from the donor and then pasted into the other parent starting from the chosen one. Note that, after this operation, the donor chromosome decreases in length, whereas the other chromosome becomes a greater length. A similar crossover operation has been adopted in Ripon and Siddique [2009], in which fixed-length binary chromosomes are used to encode the cluster medoids.

For prototype-based encoding, the chromosome lengths are usually small because it does not depend on the number of data points. Hence the crossover operations are computationally less expensive. However, for the datasets with a large number of attributes, the length of the chromosomes increases substantiallyand the crossover operation also becomes time consuming. Another issue is that the chance of the disruption of cluster centers has to be avoided by carefully choosing the crossover points. If the chromosomes encode a variable number of clusters, then some additional level of complexity comes for the crossover operations. Although different approaches have been taken in this context, as discussed earlier and depicted in Figure 4, no work has been done to compare these methods empirically. Another important disadvantage of prototype-based encoding is that it may happen that one cluster prototype is encoded in both parent chromosomes. Thus, after crossover, one of the children may receive the same prototype twice. Either one has to discard this faulty child or has to take some corrective measure to repair the chromosome, which in turn increases the time requirement.

5.2.2. Crossover for Point-based Encoding. As discussed before, there are two main categories of point-based encoding. One is cluster label-based encoding, the another is locus-based adjacency graph encoding. For cluster label-based encoding, both single-point and uniform crossover operators have been found to be used by different multiobjective clustering algorithms. In Liu et al. [2005], the single-point crossover operation has been used. They have applied some postprocessing of the offspring chomosomes by relabeling each position of an offspring to contain values in $\{1, \ldots, K\}$ if the offspring has K different labels after crossover. Similar single-point crossover has been used in Özyer et al. [2004] and Praditwong et al. [2011]. On the other hand, in Demir et al. [2010], instead of single-point crossover, uniform crossover [Goldberg and Segrest

1987], with minor modification, has been applied on the parent chromosomes. In this approach, initially, all of the data points are marked as *uncovered*. At each step, they randomly select one of the uncovered points (x) and one of the parent chromosomes. Thereafter, all the uncovered points in the cluster containing x in the selected parent are put in one cluster. The newly covered data points are marked as *covered*. This goes on until all points are marked as *covered*. In this process, only one offspring is generated from each crossover operation and the number of clusters in the offspring may be the same as or different from its parents. However, it still possesses some clustering information contained in the parent chromosomes. Note that before applying this crossover operation, the parent chromosomes are relabeled so that the first data point and the other points in the same cluster of it are put in cluster 1; next, the unlabeled point and other points within the same cluster of it are put into cluster 2, and so on.

For locus-based adjacency graph representation, the most popular algorithm MOCK [Handl and Knowles 2005c, 2006, 2007] use a simple uniform crossover approach [Goldberg and Segrest 1987]. In this approach, a random binary mask vector of the same length of the chromosomes is produced. Subsequently the genes of the two parent chromosomes corresponding to the 1 positions of the mask are exchanged for yielding two offspring chromosomes. They used this approach as they found that uniform crossover is unbiased with respect to the ordering of genes and can generate any combination of alleles from the two parent chromosomes. Several other algorithms [Qian et al. 2008; Folino and Pizzuti 2010; Shirakawa and Nagao 2009; Kim et al. 2010], in which locus-based adjacency graph encoding technique has been adopted, have used a similar uniform crossover technique.

The main advantage of crossover in point-based encoding is that the crossover operation does not have to deal with variable-length chromosomes, since all the chromosomes have a fixed length equal to the number of points. However, a major problem is that the crossover operation usually needs to handle large chromosomes when the datasets are large. Thus for larger datasets, the crossover operation is time consuming. Another problem, particularly for cluster label-based encoding, is the problem of consistency between the parents. It may happen that the same cluster is represented by different labels in two parents. Thus after crossover, the children will contain disrupted and inconsistent cluster labels. However, this problem does not occur for adjacency representation. In either case, uniform crossover seems to be a better option than single-point crossover, since in uniform crossover, better mixing of the parents is achieved, yielding higher diversity in the offspring solutions. As in the case of prototype-based encoding, no comparative study is available among different crossover strategies.

5.3. Mutation Operation

Mutation refers to small changes in the chromosomes and is used for maintaining the diversity of the population (line 10 of Algorithm 1). Different multiobjective clustering algorithms have used different mutation techniques. We have again broadly classified these techniques into two categories depending on the encoding policies of the algorithms. Figure 5 shows the broad classification of different mutation schemes used in different multiobjective clustering algorithms.

5.3.1. Mutation for Prototype-Based Encoding. The majority of the algorithms proposed by Maulik et al. [2009] that use centroid-based encoding employ a real-point mutation operator as follows [Bandyopadhyay et al. 2007a; Mukhopadhyay and Maulik 2009; Maulik et al. 2009; Mukhopadhyay et al. 2009]: A cluster center z_k, encoded in a chromosome, is mutated as follows. A random value ϵ between 0 and 1 is obtained with uniform distribution. Suppose the value of the cluster center z_k in the lth feature is z_{kl}.

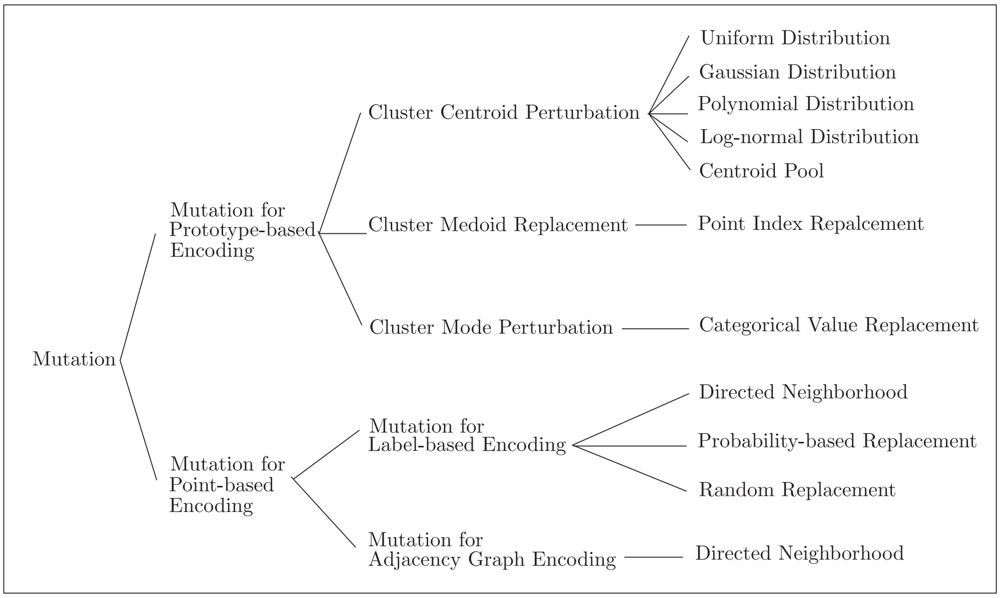
Mutation
Mutation for Prototype-based Encoding
Mutation for Point-based Encoding
Cluster Centroid Perturbation
Cluster Medoid Replacement
Cluster Mode Perturbation
Mutation for Label-based Encoding
Mutation for Adjacency Graph Encoding
Uniform Distribution
Gaussian Distribution
Polynomial Distribution
Log-normal Distribution
Centroid Pool
Point Index Repalcement
Categorical Value Replacement
Directed Neighborhood
Probability-based Replacement
Random Replacement
Directed Neighborhood

Fig. 5. Classification of mutation schemes for different multiobjective evolutionary clustering algorithms.

Then after mutation z_{kl} takes the value z'_{kl} such that

$$z'_{kl} = (1 \pm 2.\epsilon).z_{kl}, \quad \text{when} \quad z_{kl} \neq 0,$$

and

$$z'_{kl} = (\pm 2.\epsilon), \quad \text{when} \quad z_{kl} = 0.$$

The $+$ and $-$ signs occur with equal probability. The basic idea of this mutation operator is to shift a randomly selected centroid slightly from its current position.

In Chen and Wang [2005], standard Gaussian mutation with zero mean and standard deviation $\sigma_1, \sigma_2, \ldots, \sigma_m$ is chosen as the mutation operator. The parameters $\sigma_1, \sigma_2, \ldots, \sigma_m$ are the self-adaptive mutation parameters that are adjusted as in Schwefel [1993].

In Ripon et al. [2006a] and Ripon et al. [2006b], with similar encoding, a slight different mutation strategy has been adopted. Let x_i be the value of the ith gene chosen to be mutated with a probability pm. After mutation, the new value y_i is produced using the following polynomial probability distribution:

$$P(\delta) = \frac{\eta + 1}{2}(1 - |\delta|)^{\delta},$$

where δ and η are perturbation factor and distribution index for mutation, respectively. If the lower and upper bounds of feature x_i are x_i^L and x_i^U, respectively, and r_i is an arbitrary value between 0 and 1, then the new mutated value is computed as follows:

$$y_i = x_i + \left(x_i^U - x_i^L\right)\overline{\delta_i},$$

where $\overline{\delta_i} = (2r_i)^{1/(\eta+1)} - 1$ if $r_i < 0.5$, and $\overline{\delta_i} = 1 - [2(1 - r_i)]^{1/(\eta+1)}$ otherwise.

Another approach, called self-adaptive log-normal mutation operator, has been used in Won et al. [2008]. For an offspring representing centroids, two self-adaptation constants and are set to $1/\sqrt{2\sqrt{Kd}}$ and $1/\sqrt{2Kd}$ (K = number of clusters, d = number of attributes), respectively. The mutation step size $\sigma_{k,j}$ corresponding to $c_{k,j}$ for $k \in \{1, \ldots, K\}$

and $j \in \{1, \ldots, d\}$ is updated as:

$$\sigma_{k,j} \leftarrow \sigma_{k,j} \exp(\rho + \rho_{k,j}),$$

where $\rho = \tau_0 N(0, 1)$ is a random constant commonly used for all the mutation step sizes in the offspring, $\rho_{k,j} = \tau_0 \mathcal{N}(0, 1)$ is a constant randomly chosen for each mutation step size, and $\mathcal{N}(0, 1)$ is a normally distributed random number whose mean and variance are zero and one, respectively. The centroid element is then updated as:

$$c_{k,j} \leftarrow c_{k,j} + \sigma_{k,j}\mathcal{N}(0, 1).$$

The mutated $c_{k,j}$ is limited within the valid range $[L_j, U_j]$.

In Kirkland et al. [2011], three mutation operators for decreasing the length of the chromosome, increasing the length of the chromosome, and updating the chromosome without changing its length are employed with percentage probabilities 50%, 25%, and 25%, respectively. In the first operator, to decrease the chromosome length, one cluster prototype is removed. To choose the cluster prototype to be removed, for all cluster prototypes the nearest cluster prototype is found. The victim prototype is chosen as that one, whose distance from the nearest prototype is minimum. Hence the points belonging to the deleted cluster prototype are likely to be placed in its nearest cluster after removal. In the second mutation operator, the length of the chromosome is increased as follows. They keep a pool of potential cluster prototypes. From this pool, the cluster prototype that is farthest from any of the prototypes encoded in the chromosome is added to the chromosome. This ensures that the added cluster prototype is not close to any pre-existing cluster prototypes so that it is expected to produce a new and interesting cluster. The final mutation operator does not change the length of the chromosome. It updates the cluster prototypes by replacing them with the mean of the points in the corresponding clusters.

For medoid-based encoding, as the indices of the points are encoded, the mutation operation is used as follows [Mukhopadhyay and Maulik 2007]: A random position is selected from the chromosome and the corresponding point index is replaced by a randomly chosen point index from the dataset, so that the chromosome does not contain repeated point indices after mutation.

Where cluster mode-based encoding is adopted [Mukhopadhyay et al. 2009a] for clustering categorical data, mutation has been done as follows. If a chromosome is selected for mutation, the position to be mutated is picked up randomly. Subsequently, the categorical value of that position is substituted by another value selected randomly from the domain of the corresponding categorical feature.

The mutation operators for prototype-based encoding try to change the prototypes randomly. For real-valued chromosomes (where cluster centers are encoded), real-valued mutation operators are used for changing the coordinate values of the cluster centers. For most of the cases, these are real-valued operations, and work fast. For variable-length chromosomes, some additional measure for changing the length of the chromosome must be taken. For centroid-based encoding, although different mutation operators have been used, as discussed earlier, one systematic comparison is never done to determine the most suitable mutation operator. For medoid-based encoding, replacement of a cluster medoid with a different point may induce a significant change in the encoded clustering structure. Moreover, the same medoid could be duplicated in the chromosome after mutation, in which case the chromosome should be discarded or repaired. For mode-based encoding also, replacement of an attribute value with another may yield a reasonable change in the clustering. Hence for medoid- and mode-based encoding, mutation in one or two positions may affect the chromosome heavily. In general, for prototype-based encoding, the mutation operators need to change only in a few positions due to shorter length of the chromosomes.

5.3.2. Mutation for Point-Based Encoding. The multiobjective clustering algorithms that have used a cluster label-based encoding scheme have used several types of mutation procedures. In Handl and Knowles [2004], a directed mutation operator in which multiple points close to each other change their cluster labels simultaneously is used. In this mutation operator, each point of a chromosome is mutated with probability $1/n$, where n is the number of points. When a point is mutated, its cluster label is changed randomly and along with that point, a predefined number of its nearest neighbors are also given the new label as given to this point. As per Handl and Knowles [2004], this mutation operator increases the speed of the convergence of the algorithm.

A different mutation operator is used in Özyer et al. [2004] and Liu et al. [2005], who employ similar encoding policy. During this mutation, each gene value a_i is replaced by a_i' based on a probability distribution for $i = 1, \ldots, n$ simultaneously. a_i' is a cluster number randomly selected from $\{1, \ldots, K\}$ with the probability distribution $p_1, p_2 \ldots, p_K$ defined using the following formula [Özyer et al. 2004; Liu et al. 2005]:

$$p_j = \frac{1.5 * \mathbb{D}_{\max}(x_i) - D(x_i, c_k)}{\sum_{k=1}^{K} 1.5 * \mathbb{D}_{\max}(x_i) - D(x_i, c_k)}. \tag{36}$$

Here $j \in \{1 \ldots, K\}$, $\mathbb{D}_{\max}(x_i) = \max_k\{D(x_i, c_k)\}$, and p_j represents the probability interval of a mutating gene to be assigned to cluster j (e.g., Roulette Wheel). In Özyer et al. [2011], the same mutation operator is used but the probability distribution is defined in a different way as

$$p_j = \frac{e^{-D(x_i, c_j)}}{\sum_{r=1}^{K} e^{-D(x_i, c_r)}}. \tag{37}$$

After mutation, the K-means centroid-updating is applied to reorganize the cluster label of each point.

In Demir et al. [2010] and Praditwong et al. [2011], simple standard mutation is used. Unlike Handl and Knowles [2004], Demir et al. [2010], and Praditwong et al. [2011] have employed random cluster label assignment-based mutation without the neighborhood bias, that is, in this case, the change in cluster label of the mutated point does not affect the other points, including its nearest neighbors.

In the algorithms that use locus-based adjacency graph encoding [Handl and Knowles 2005a, 2005c, 2006, 2007], the directed neighborhood mutation operator is used with slight variation. In this mutation, each point i is linked to its L nearest neighbors $\{nn_{i1}, nn_{i2}, \ldots, nn_{iL}\}$. Hence the search space is reduced to L^n. The probability of mutation is found adaptively. Note that "longer links" in the encoded in the chromosome are expected to be less preferable. For example, a link $i \to j$ with $j = nn_{il}$ should be more desirable than a link $i \to j'$, where $j' = nn_{ik}$ and $l < k$. It is used as a bias to the mutation probability of each link $i \to j$. The mutation probability pm is then given by [Handl and Knowles 2007]:

$$pm = \frac{1}{n} + \left(\frac{l}{n}\right)^2, \tag{38}$$

where $j = nn_{il}$ and n is the number of data points. This helps discard unfavorable links faster. These mutation operators have been adopted by several other works as well [Qian et al. 2008; Folino and Pizzuti 2010; Shirakawa and Nagao 2009].

For point-based encoding, mutation operation usually takes longer due to larger length of the chromosome. Also, changes in one/two positions do not alter the clustering structure in the chromosome much. Hence, in general, mutation operations with low mutation probability are not able to generate much diversity in the population, resulting in low convergence rate. To cope with this problem, Handl and Knowles applied neighborhood-based mutation [Handl and Knowles 2007], in which change in one position induces changes in neighboring points as well. It helps todiversify the population as well as increase the convergence rate.

6. MAINTAINING NONDOMINATED SOLUTIONS

Unlike single-objective evolutionary algorithms, in multiobjective techniques, an important step is to keep the set of current nondominated solutions. It is depicted in line 12 in Algorithm 1. There are mainly two different approaches for this. First is to keep the nondominated chromosome within the population itself. The second approach is to store it in a separate archive outside the population. The storing strategy used by multiobjective evolutionary clustering algorithms mainly depends on the underlying multiobjective evolutionary algorithm. These are discussed here.

6.1. Storing Nondominated Solutions within Population

The NSGA-II algorithm [Deb et al. 2002] is used by many of the multiobjective evolutionary clustering algorithms [Bandyopadhyay et al. 2007a; Özyer et al. 2004; Chen and Wang 2005; Ripon and Siddique 2009; Won et al. 2008; Liu et al. 2005; Kim et al. 2010; Kirkland et al. 2011]. In this algorithm, the current nondominated solutions are kept within the population. In this algorithm, the whole population is classified using nondominated sorting to give the solutions different ranks. Elitism is performed by combining the parent and child populations to create a population of twice the original population size. From this combined population, half of the solutions are propagated to the next generation based on the nondominated sorting and crowding distance criteria. Thus no separate archive is kept for storing the current nondominated front.

A few multiobjective clustering methods, such as those in Liu et al. [2005], employed NPGA [Horn and Nafpliotis 1993] as the underlying optimization tool. In NPGA also, the current nondominated front is maintained within the population. Note that NPGA is a nonelitist approach and uses special tournament selection and fitness sharing concepts. Thus it is not as effective as NSGA-II, which is elitist in nature [Deb et al. 2002].

The major advantage of keeping the nondominated solutions within the population is that in this case the multiobjective evolutionary algorithm is very similar to the single-objective one, and no extra overhead is required to maintain a separate external archive and update that in every generation. However, one possible drawback is that the size of the nondomination front cannot go beyond the population size; if so, it is truncated using a crowding distance metric. For a clustering problem, the number of possible trade-off clustering solutions can be very large because of the complexity of the dataset and different number of clusters. In such cases, NSGA-II-based clustering algorithms may fail to generate all possible trade-offs if the population size is not sufficiently large.

6.2. Storing Nondominated Solutions in Archive

The other two multiobjective evolutionary algorithms using clustering, that is, PESA-II [Corne et al. 2001] and SPEA2 [Zitzler et al. 2001], have used an external archive for storing the current nondominated solutions. The PESA-II algorithm has been used in clustering methods proposed in Handl and Knowles [2007], Qian et al. [2008], Shirakawa and Nagao [2009], and Matake et al. [2007]. In PESA-II, the algorithm maintains an internal (primary) population and an external population (archive). In each generation, every nondominated solution of the internal population enters the archive if it is not dominated by any of the archive solutions. After the incorporation of the nondominated solutions in the archive, all the dominated solutions are removed

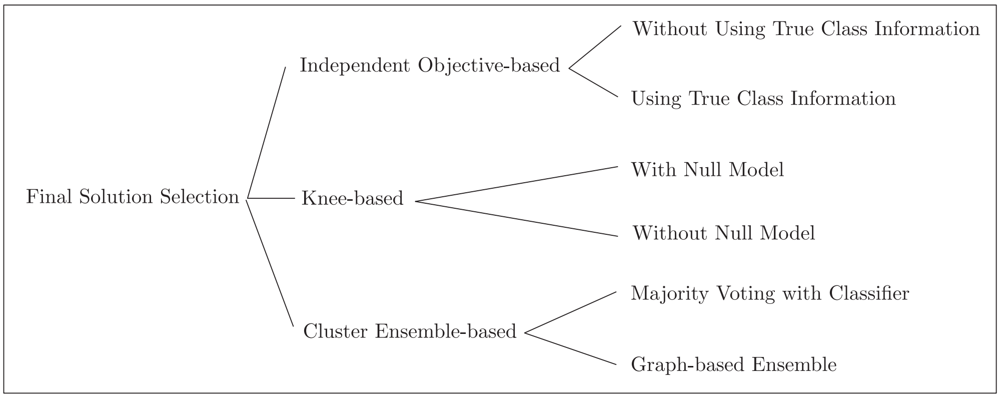
Final Solution Selection
Independent Objective-based
Without Using True Class Information
Using True Class Information
Knee-based
With Null Model
Without Null Model
Cluster Ensemble-based
Majority Voting with Classifier
Graph-based Ensemble

Fig. 6. Classification of different approaches for obtaining the final solution from the nondominated solutions in different multiobjective evolutionary clustering algorithms.

from it. If the archive size still exceeds a predefined limit, then the extra solutions are eliminated based on crowding factor.

In SPEA2, an external archive is also maintained along with the internal population. The external archive contains the solutions from the first nondominated fronts; however, it may also contain solutions from other fronts if the size of the first front is less than the predefined archive size. On the other hand, if the size of the archive exceeds the predefined threshold, then truncation is performed by clustering while preserving the boundary solutions. Unlike PESA-II, in SPEA2, the archive size is kept constant over the generations and it may contain dominated solutions along with the nondominated ones.

The advantage of keeping an archive for storing the nondominated clustering solutions is that the archive does not involve the evolutionary operators such as crossover and mutation. Hence we can keep a higher number of nondominated solutions (as the archive size may be larger than population size) without increasing time complexity, since crossover and mutation are still applied to the solutions in the main population only. However, the drawback of keeping an archive is that there is extra overhead for updating the archive in each generation and applying the truncation operation from time to time. Some experimental work is needed for comparing the different population maintenance strategies for the multiobjective evolutionary clustering techniques to study the effectiveness of the two approaches of keeping track of the nondominated front.

7. OBTAINING FINAL SOLUTION

As per the nature of the multiobjective optimization algorithms, any multiobjective clustering method generates a set of nondominated trade-off solutions. None of these solutions can be improved further in any objective value without degrading some other objective value. Therefore there is a need of a method to obtain the final clustering solutions from the nondominated set. A number of such techniques have been proposed by different multiobjective clustering algorithms. Here, we classify all the methods in three broad categories: selecting a solution based on an independent objective, selecting a solution by finding the knee of the nondominated front, and obtaining a final solution based on ensembly of the nondominated solutions [Mukhopadhyay et al. 2014b]. The classification of different approaches is depicted in Figure 6.

7.1. Independent Objective-Based Approach

Many of the multiobjective clustering techniques have used this simple approach. In this approach, the solution that provides the best value for a cluster validity index that is not used as an objective function of the algorithm is selected as the final clustering solution. Bandyopadhyay et al. [2007a] and Mukhopadhyay et al. [2006] optimized J_m and *XB* indices simultaneously, and the final solution is chosen from the nondominated set using index $\mathcal{I}$. In a similar approach, in Mukhopadhyay and Maulik [2011], fuzzy cluster compactness and separation have been used as the two objectives, while $\mathcal{I}$ index is considered as the selection criterion. In Mukhopadhyay et al. [2009], *XB* and $\mathcal{I}$ indices were taken as the objective functions and silhouette index is used for picking up the final solution. In Özyer et al. [2004], the two objective functions were *TWCV* and the number of clusters. The authors have employed *DB* index and *SD* index [Halkidi et al. 2000, 2001] for choosing the final solution from the nondominated front. In Liu et al. [2005] also, *TWCV* and the number of clusters were used as the optimizing criteria, and other different validity measures, such as Dunn index, *DB* index, and silhouette index, have been exploited for final solution selection. The results were compared among themselves. In the GraSC algorithm, Demir et al. [2010] have chosen silhouette index and min-max cut index to optimize, and used *DB* index to pick up the ultimate solution. Shirakawa and Nagao [2009] have proposed another simple scheme to choose a solution from the final nondominated front. In this scheme, they have first normalized the two objective values of the nondominated solutions within range [0, 1]. Subsequently, they have added the two objective values for each solution and reported the solution having the best value of the sum. In Kirkland et al. [2011], three objective functions—average cluster variance, *ABGSS*, and the cluster connectedness—have been optimized, and Rand index ($\mathcal{R}$) [Maulik et al. 2011] has been considered to obtain the final solution from the Pareto front. Note that computation of $\mathcal{R}$ needs the knowledge of true clustering of the dataset. Therefore this approach is not useful when the true clustering information is not known. In Faceli et al. [2008] also, adjusted Rand index (*ARI*) [Maulik et al. 2011] is used for reducing the set of nondominated solutions to a smaller set so that the user can compare the solutions in the reduced set to select one manually. However, they did not compute *ARI* against true clustering; rather, they computed *ARI* value between each pair of solutions in the nondominated front. Thereafter they chose the solution that is most different from the other solutions. The solutions that are nearer to the chosen solution (selected based on a threshold value on *ARI* score) are deleted. This process continues until no more solutions can be deleted. This way, they select a much reduced set from the complete set of nondominated solutions.

Although the use of an independent validity index for choosing the final solution is simple to implement and has a low time requirement, the final result may be biased depending on the validity index chosen for picking up the final solution from the Pareto front. Moreover, this approach can also be criticized in that this independent validity measure could have been optimized directly.

7.2. Knee-Based Approach

Given a nondominated front, an interesting solution is one for which the change of one objective value induces maximum change in the others. The solution is termed a "knee" of the nondominated front. Many multiobjective clustering approaches have employed some knee-finding technique for choosing the final clustering solution from the nondominated set. Handl and Knowles have used this knee-based approach in their MOCK algorithm [Handl and Knowles 2005c, 2006, 2007]. This approach is motivated by the GAP statistic [Tibshirani et al. 2001]. This is performed by comparing the generated nondominated front with some control fronts. These control fronts are produced by applying MOCK on some random control data of same size and same attribute ranges. The solution corresponding to the largest distance between the actual nondominated front and the control fronts is chosen to be the final solution. However, there exists no suitable motivation for selecting the knee solution, as it is not well explained under clustering context why this knee solution should be the most interesting one [Mukhopadhyay et al. 2014b]. Another important limitation of the approach is its time complexity, because here the algorithm must be executed multiple times with random datasets to produce the control fronts. Due to this, Handl and Knowles [2005b] proposed using the same model, but within the space of principal components to reduce the time complexity for high-dimensional data.

The problem of time complexity in choosing the knee solution is also addressed in Matake et al. [2007], in which the authors proposed a scheme to first reduce the size of the nondominated front by removing redundant and unpromising solutions. Subsequently, they proposed two schemes for determining the final solution from the reduced front. For reducing the size of the nondominated front, they assumed the front to be convex, and thereafter removed partly nonconvex solutions by calculating the adjacent angles. In the first scheme of solution selection from the reduced nondominated front, they used only the adjacent angle as the determiner of the knee solution. Although this scheme is definitely much faster than that of MOCK, it is not as good as that of MOCK. It is prone to errors when a clear knee is not observed in the nondominated front, although the reduced front includes the proper solution. To overcome this drawback, they provided a second scheme that redefines the overall deviation index by incorporating a filter to nullify the reduction during separation of wide noncircular clusters. They re-evaluate the modified overall deviation for all nondominated fronts, then determine the knee based on the modified nondominated front. This scheme has been shown to provide a better result than the first one, but both schemes perform not as well as MOCK. However, they are able to reduce the time complexity by a large factor.

The knee-based approaches have been used for two-dimensional, nondominated fronts only. This may be because of the complexity of computing the control fronts [Handl and Knowles 2007], which is further increased if the number of objective functions is more than two. Moreover, for the approaches that do not employ control fronts for selecting the knee solution and rely on computing adjacent angles [Matake et al. 2007], extending the concept of angles to higher dimensions is not so straightforward. Therefore, computing knee solution from higher-dimensional Pareto fronts is an important issue to be addressed in future work.

7.3. Cluster Ensemble-Based Approach

All the methods discussed above use the properties of the Pareto-optimal front or some external metric to select a solution from the non-dominated front. These methods almost all the times choose one solution from the non-dominated set as the final solution. However, it may be noted that evidently all the non-dominated solutions possess some information about the clustering structure of the dataset in hand. Therefore a consensus among the non-dominated solutions may result in a good clustering solution [Maulik et al. 2009]. There are a few recent approaches that address this problem.

Motivated by this observation, Maulik et al. proposed a novel technique for obtaining the final solution that considers all the nondominated solutions using the input dataset as a guiding factor [Maulik et al. 2009; Mukhopadhyay and Maulik 2009; Mukhopadhyay et al. 2009a]. The approach is to integrate the multiobjective clustering technique with a supervised classifier to obtain the final solution from the nondominated set. The basic motivating idea is that if some data points are most of the time clustered together by a majority of the nondominated solutions, then these points may confidently be assumed to be clustered appropriately. Therefore these points can be utilized to train a supervised classifier, which can then be employed for clustering the rest of the points having lower confidence regarding their cluster assignment. Thus this method gives equal importance to all nondominated solutions and a majority voting technique is applied for identifying the points of the training set. In the majority voting technique, first the cluster label vectors are generated for each nondominated solution based on nearest-centroid criteria. Thereafter, the nondominated label vectors are made consistent with each other [Mukhopadhyay and Maulik 2009; Maulik et al. 2009]. Subsequently, for each point, if a majority of the solutions gives that the same class label, then the point is assigned to that cluster. If for some point no class label gets the majority, then the point is not assigned to any class immediately. Finally, the points that are given a class label in this process are treated as a training set and a classifier is trained using these points. In different works, the authors used different classifiers such as SVM [Mukhopadhyay and Maulik 2009; Maulik et al. 2009; Mukhopadhyay et al. 2011], ANN [Mukhopadhyay et al. 2008], and *k*-nn [Mukhopadhyay et al. 2009a] for this purpose. The remaining points, which are not assigned to any cluster during the majority voting stage, are now classified by the trained classifier. This way, the class labels for all points are generated.

In another work [Mukhopadhyay et al. 2009b], some existing well-known cluster ensemble techniques, such as Cluster-based Similarity Partitioning Algorithm (CSPA), HyperGraph Partitioning Algorithm (HGPA), and Meta-CLustering Algorithm (MCLA) [Strehl and Ghosh 2002] have been used to combine the nondominated solutions to obtain the final clustering, and their performance has been compared. All these cluster ensemble approaches are based on graph partitioning [Strehl and Ghosh 2002]. In CSPA, data points are considered as nodes of the graph and the edge weight between two nodes is computed based on the frequency by which the corresponding points are grouped together in the candidate clustering solutions. The resulting graph is then partitioned to obtain the final clustering. In HGPA, a hypergraph representation for the candidate clustering solutions is computed and subsequently a hypergraph partitioning algorithm is used to obtain the final clustering. In MCLA, a meta-graph is constructed based on the candidate clustering solutions, which contains several hyperedges. Then the related hyperedges are grouped and collapsed to obtain the final clustering. Mukhopadhyay et al. [2009b] first generated candidate nondominated clustering solutions through NSGA-II–based multiobjective clustering, and then one of the three ensemble techniques is applied to these nondominated solutions to get the final clustering. This approach has been applied for clustering MRI brain images, and the results indicate that the HGPA algorithm performs best in most cases. In a similar approach [Qian et al. 2008], MCLA has been used for the ensemble purpose.

The ensemble-based methods are promising and have well-defined motivation. But these methods are usually time-consuming. Moreover, the final solution depends on the choice of the ensemble algorithm. Furthermore, for some cases, ensemble-based methods need to map one nondominated solution to another [Mukhopadhyay et al. 2009a] to make the cluster labeling consistent with each other. Thus the final clustering solution also depends on the employed mapping scheme.

Finally, we have summarized the process of some well-known multiobjective evolutionary clustering algorithms in two tables. A total of 19 different algorithms, having diverse range of techniques with respect to different evolutionary algorithm and clustering issues, are considered here. In Table I, we have provided an overview of these algorithms with respect to their encoding strategy and population initialization method, objective functions that have been optimized, the MOO algorithm that is used as the underlying optimization tool, and the type of data on which they are applied. In Table II, the same algorithms are considered, and we report the technique for crossover, mutation, and obtaining a final solution from the nondominated set used in different

Table I. Overview of Different Popular Multiobjective Evolutionary Clustering Algorithms: Encoding Technique and Population Initialization Method, Objective Functions Used, Underlying MOO Algorithm Used, and Data Types on Which They Operate

Algorithm	Encoding/ Initialization	Objectives	MOO Algorithm	Data type
VIENNA [Handl and Knowles 2004]	Label-based/ Voronoi cell	$Dev(C)$ and $Conn(C)$	PESA-II	Continuous
MOCK-am [Handl and Knowles 2005c]	Adjacency graph/ MST and K-medoids	$Dev(C)$ and $Conn(C)$	PESA-II	Categorical/ Distance matrix
MOKGA [Liu et al. 2005]	Label-based/ Random	$TWCV$ and # Clusters K	NPGA	Continuous
MOEA (Dynamic) [Chen and Wang 2005]	Centroid-based/ Random	$Dev(C)$ and $Conn(C)$	NSGA-II	Continuous
MOCK [Handl and Knowles 2007]	Adjacency graph/ MST and K-means	$Dev(C)$ and $Conn(C)$	PESA-II	Continuous
VRJGGA [Ripon et al. 2006a]	Centroid-based/ Random	Entropy H and Separation $Sep(C)$	NSGA-II	Continuous
MOGA [Bandyopadhyay et al. 2007a]	Centroid-based/ Random	J_m and XB	NSGA-II	Continuous
MOGA (medoid) [Mukhopadhyay and Maulik 2007]	Medoid-based/ Random	$Dev(C)$ and Silhouette	NSGA-II	Categorical
GraSC [Demir et al. 2007]	Label-based/ Random	$MinMaxCut$ and Silhouette	SPEA2	Categorical/ Distance matrix
MECEA [Qian et al. 2008]	Adjacency graph/ MST and K-means	$Dev(C)$ and $Conn(C)$	PESA-II	Continuous
MOES (Hybrid) [Won et al. 2008]	Centroid-based (variable)/ Random	$TWCV$ & # Clusters	NSGA-II	Continuous
MOGA-SVM [Maulik et al. 2009]	Centroid-based/ Random	J_m and XB	NSGA-II	Continuous
Variant of MOCK [Shirakawa and Nagao 2009]	Adjacency graph/ MST	$Dev(C)$ and $Edge(C)$	SPEA2	Continuous
MOGA (mode) [Mukhopadhyay et al. 2009a]	Mode-based/ Random	Normalized J_m and Fuzzy Sep. $\mathcal{S}$	NSGA-II	Categorical
EMCOC [Ripon and Siddique 2009]	Medoid-based (binary)/ Random	Entropy H and Separation $Sep(C)$	NSGA-II	Continuous
AI-NSGA-II [Kim et al. 2010]	Adjacency graph/ Random	$MinMaxCut$ and Silhouette	NSGA-II	Graph
DYN-MOGA [Folino and Pizzuti 2010]	Adjacency graph/ Random	$CS(C)$ and NMI	NSGA-II	Graph
MOVGA [Mukhopadhyay and Maulik 2011]	Centroid-based (variable)/ Random	Normalized J_m and Fuzzy Sep. $\mathcal{S}$	NSGA-II	Continuous
MOCA [Kirkland et al. 2011]	Centroid-based (variable)/ Random	Average deviation, $ABGSS$ and Conn(C)	NSGA-II	Continuous

Table II. Overview of Different Popular Multiobjective Evolutionary Clustering Algorithms: Crossover Technique, Mutation Technique, and Technique for Obtaining Final Solution

Algorithm	Crossover	Mutation	Obtaining Final Solution
VIENNA [Handl and Knowles 2004]	None	Neighborhood-based	Independent objective-based (F-measure)
MOCK-am [Handl and Knowles 2005c]	Uniform	Neighborhood-based	Knee-based (with null model)
MOKGA [Liu et al. 2005]	Single-point	Probability-based replacement	Independent objective-based (*Dunn*, *DB*, Silhouette C, SD and S_Dbw indices)
MOEA (Dynamic) [Chen and Wang 2005]	Two-point	Centroid perturbation (Gaussian distribution)	Independent objective-based (F-measure)
MOCK [Handl and Knowles 2007]	Uniform	Neighborhood-based	Knee-based (with null model)
VRJGGA [Ripon et al. 2006a]	Jumping gene	Centroid perturbation (Polynomial distribution)	Independent objective-based (Deviation and *Dunn* index)
MOGA [Bandyopadhyay et al. 2007a]	Single-point	Centroid perturbation (Uniform distribution)	Independent objective ($\mathcal{I}$ index)
MOGA (medoid) [Mukhopadhyay and Maulik 2007]	Single-point	Medoid replacement (Point index)	Independent objective (Minkowski score)
GraSC [Demir et al. 2007]	Uniform/ (modified)	Random replacement	Nondomination status
MECEA [Qian et al. 2008]	Uniform	Neighborhood-based	Ensemble-based (Graph-based—M CLA)
MOES (Hybrid) [Won et al. 2008]	Centroid pool	Centroid perturbation (Log-normal distribution)	None
MOGA-SVM [Maulik et al. 2009]	Single-point	Centroid perturbation (Uniform distribution)	Ensemble-based (Majority vote and SVM classifier)
MOCK Variant [Shirakawa and Nagao 2009]	Uniform	Neighborhood-based	Knee-based (without null model)
MOGA (mode) [Mukhopadhyay et al. 2009a]	Single-point	Mode perturbation (Categorical value replacement)	Ensemble-based (Majority vote and k-nn classifier)
EMCOC [Ripon and Siddique 2009]	Jumping gene	None	Independent objective-based (Entropy & Separation)
AI-NSGA-II [Kim et al. 2010]	Uniform	Neighborhood-based	Non-domination status
DYN-MOGA [Folino and Pizzuti 2010]	Uniform	Neighborhood-based	Independent objective-based (Modularity)
MOVGA [Mukhopadhyay and Maulik 2011]	Single-point	Centroid perturbation	Independent objective-based ($\mathcal{I}$ index)
MOCA [Kirkland et al. 2011]	Exchange corresponding prototypes	Centroid pool (add/delete/modify centroids)	Independent objective-based (Rand index)

Table III. Chronological Summary of Major Landmarks of Multiobjective Evolutionary Clustering

Year	Algorithm	Landmark
2004	VIENNA [Handl and Knowles 2004]	—First multiobjective evolutionary clustering
2005	MOCK-am [Handl and Knowles 2005c]	—First multiobjective clustering around medoids —First multiobjective clustering for categorical data —First approach for knee-based solution selection using null model
2007	MOGA [Bandyopadhyay et al. 2007a]	—First centroid-based encoding under multiobjective context —First multiobjective fuzzy clustering —First application in satellite image segmentation
2007	MOCK [Handl and Knowles 2007]	—First locus-based adjacency graph representation —First approach for initializing population with K-means clustering and minimum spanning tree —First neighborhood-based mutation operator
2007	MOGA (medoid) [Mukhopadhyay and Maulik 2007]	—First medoid-based chromosome representation
2008	MOES (Hybrid) [Won et al. 2008]	—First variable-length centroid-based chromosome representation
2009	MOGA-SVM [Maulik et al. 2009]	—First approach for final solution selection based on fuzzy majority voting and SVM classifier —First application in gene expression data clustering
2009	MOGA (mode) [Mukhopadhyay et al. 2009a]	—First multiobjective fuzzy clustering with mode-based representation —First multiobjective fuzzy clustering for categorical data
2009	EMCOC [Ripon and Siddique 2009]	—First medoid-based representation using binary strings
2009	MOGA-CSPA, MOGA-HGPA, MOGA-MCLA [Mukhopadhyay et al. 2009b]	—First approach with three objective functions —First approach for applying graph-based cluster ensemble techniques to obtain the final clustering —First application in MRI brain image segmentation
2010	AI-NSGA-II [Kim et al. 2010]	—First application in social network analysis
2011	MOKGA [Özyer et al. 2011]	—First approach with four objective functions —First application in skyline computation
2013	IMOC [Mukhopadhyay et al. 2013]	—First interactive approach to multiobjective clustering

algorithms. In both the tables, the algorithms have been arranged in increasing order of their publication year to give an idea how the algorithms have evolved over time. For a more clear view of the landmarks in the field of multiobjective evolutionary clustering, we have reported the major milestones in this area in Table III in chronological order.

8. OTHER METAHEURISTIC ALGORITHMS FOR MULTIOBJECTIVE CLUSTERING

As mentioned earlier, most of the multiobjective clustering algorithms are based on multiobjective genetic algorithms. However, there are some other evolutionary algorithms, such as differential evolution (DE), genetic programming (GP), and gene expression programming (GEP), and other related nature-inspired search techniques such as simulated annealing (SA), particle swarm optimization (PSO), and ant colony optimization (ACO) that have also been employed for multiobjective clustering purposes. As this article mainly focuses on evolutionary algorithms, we discuss some other multiobjective clustering algorithms that use other evolutionary computation tools such as DE, GP, and GEP. Besides these, for the sake of completeness, we also mention a few other multiobjective clustering methods that use other metaheuristics such as SA, PSO, and ACO as the underlying optimization framework.

8.1. Multiobjective Clustering Using Differential Evolution

After genetic algorithms, the next most popular evolutionary optimization tool is differential evolution (DE) [Price et al. 2005]. DE algorithms, unlike GAs, usually evolve with one-to-one correspondence between parent and child vectors; a better solution between the parent and child is kept. The child vector is generated using a mutation based on differential operation followed by a crossover operation between the parent and the intermediate vector. DE also has a few variants for multiobjective optimization such as Pareto DE (PDE) [Abbass and Sarker 2002], multiobjective DE (MODE) [Xue et al. 2005], DE for multiobjective optimization (DEMO) [Robic and Filipic 2005], and nondominated sorting DE (NSDE) [Iorio and Li 2004]. Suresh et al. [2009b] have compared all these variants of multiobjective DE for the clustering problem. As DE algorithms are naturally suitable for real-coded vectors, the authors used real-coded, centroid-based encoding (variable-length) for representing the clustering solutions. Two fuzzy validity indices, J_m and XB, have been optimized simultaneously as in Bandyopadhyay et al. [2007a]. The final solution from the Pareto front is selected based on gap statistic as discussed in Section 7.2. The DE-based approaches have also been compared with MOCK [Handl and Knowles 2007] and MOGA [Bandyopadhyay et al. 2007a] clustering methods. The experimental results indicate that the multiobjective DE variants are able to yield comparable results with MOCK and MOGA for a wide variety of datasets. Suresh et al. [2009a] have applied the proposed algorithms for gene expression data clustering. Another similar MODE-based fuzzy clustering has been proposed in Saha and Maulik [2014], in which the same set of validity has been optimized. Here, the authors have applied the proposed algorithm for clustering MR brain images. In Saha et al. [2011b], a modified improved version of multiobjective DE-based fuzzy clustering has been proposed; the authors applied their algorithm for segmentation of remote sensing imagery. A crisp version of the algorithm has been provided in Saha et al. [2011a], in which the authors optimized DB and cluster separation simultaneously and used the Minkowski score to select the final solution from the nondominated front.

8.2. Multiobjective Clustering using Genetic Programming

Genetic Programming (GP) [Banzhaf et al. 1998] is a variation of genetic algorithms, in which the individuals represent computer programs and their fitnesses are measured in terms of their ability to solve the computational program. There are few multiobjective versions of GP that have been used for the multiobjective clustering problem [Sun et al. 2006; Coelho et al. 2010]. The first effort in this area can be found in Sun et al. [2006], in which the authors have proposed a multiobjective clustering method called "Parallel Hybrid Clustering using Genetic Programming and Multi-Objective Fitness with Density (PYRAMID)." This algorithm is a parallel one and thus deployable to multiple processors in parallel. The individuals have a tree representation that encodes a set of rules (blocks). In the tree representation, the rules are denoted by the leaf nodes, and internal nodes represent union operations on the rules. This is a non-Pareto–based approach and optimizes a combination of density-based fitness functions in a single-objective manner. Elitism has also been incorporated. The first Pareto-based GP approach for multiobjective clustering can be found in Coelho et al. [2010], who have proposed a GP-based multiobjective method for consensus partitioning in which the main goal is to evolve a suitable consensus function to ensemble a set of base partitions. The individuals are represented in forms of trees for which each leaf node represents a base partition and internal nodes represent an ensemble technique such as CSPA, HGPA, and MCLA [Strehl and Ghosh 2002]. To assess the goodness of each individual, the two objective functions used are $Dev(C)$ and $Conn(C)$, as in MOCK clustering [Handl and Knowles 2007]. The underlying optimization process follows the strategy adopted in NSGA-II. The results have demonstrated the effectiveness of the proposed technique. Hence this approach gets help from the tree representation of GP to provide a novel way of combining the base partitions in a consensus clustering.

8.3. Multiobjective Clustering Using Gene Expression Programming

Gene Expression Programming (GEP) [Ferreira 2001] is similar to GP; however, in GEP, computer programs are evolved using a genotype-phenotype system. Here, the computer programs of different sizes are represented in linear chromosomes (genotype) of fixed length, from which expressive parse trees (phenotype) are inferred and subsequently decoded to compute the fitness values. We could find only one multiobjective clustering approach based on GEP, in Zheng et al. [2012]. Each chromosome of the proposed multiobjective GEP for clustering (MGEPC) consists of a head part and a tail part. The head part may contain clustering algebraic operations ($\cup$ and $\cap$) and terminal symbols (indices of data points), whereas the tail part consists of the terminal symbols only. The corresponding parse tree represents a possible hierarchical clustering solution. The two objective functions optimized are $Dev(C)$ and $Conn(C)$, as used in MOCK [Handl and Knowles 2007]. The underlying optimization strategy is based on NSGA-II. However, the algorithm also keeps an external archive to store the nondominated solutions. The final solution from the Pareto front is selected based on MOCK's strategy of knee solution selection. It may be noted that GEP-based multiobjective clustering is a promising approach but is still in its early stage. More study is needed to reduce its time complexity since clustering datasets may be large and conversion from genotype to phenotype may need reasonable computational cost.

8.4. Multiobjective Clustering Using Other Metaheuristics

There are several other nature-inspired metaheuristic techniques besides evolutionary algorithms. Recently, these metaheuristic algorithms have been applied for the multiobjective clustering problem. It may be beyond the scope of this article to review all these techniques, but for the sake of completeness, these algorithms are worth mentioning. For example, many of the important multiobjective clustering approaches have used different metaheuristics such as simulated annealing [Saha and Bandyopadhyay 2009, 2013], Particle Swarm Optimization [Agrawal et al. 2008], and Immune Response System [Gong et al. 2007; Liu et al. 2010] as the underlying optimization tools. However, more study is needed to assess the performance of these techniques in comparison with the traditional evolutionary algorithm-based clustering approaches in a systematic way.

9. APPLICATIONS OF MULTIOBJECTIVE EVOLUTIONARY CLUSTERING

The multiobjective evolutionary clustering algorithms developed in various studies have been applied in different real-life applications such as image segmentation, bioinformatics, Web mining, social networks, software engineering, time series analysis, and computer security [Mukhopadhyay et al. 2014b]. In this section, we describe such applications of multiobjective clustering algorithms.

9.1. Applications in Image Segmentation

One of the primary application areas of multiobjective evolutionary clustering algorithms is image segmentation. Usually, the problem of image segmentation can be modeled as a problem of clustering the image pixels in the intensity space

[Bandyopadhyay et al. 2007a]. For multi-spectral images, different bands serve as the different attributes of the dataset. Various types of images have been clustered using multiobjective clustering algorithms. For example, in Shirakawa and Nagao [2009], a number of benchmark color images have been segmented. Maulik and others [Bandyopadhyay et al. 2007a; Mukhopadhyay et al. 2006; Mukhopadhyay and Maulik 2009] have applied multiobjective fuzzy clustering for segmentation of remote sensing imagery of multi-spectral IRS and SPOT satellite images of some popular Indian cities. Besides these, application of multiobjective evolutionary clustering can also be found in segmentation of MRI medical imagery of some parts of human body, such as the brain [Mukhopadhyay et al. 2009b; Mukhopadhyay and Maulik 2011]. Multiobjective clustering has been applied in texture image segmentation as well [Qian et al. 2008].

9.2. Applications in Bioinformatics

Multiobjective evolutionary clustering algorithms have also been applied widely in bioinformatics, in which microarray gene expression datasets are clustered to group co-expressed genes [Shannon et al. 2003; Gasch and Eisen 2002]. Genes that have similar expression patterns are likely to show similar functionality. Therefore grouping genes in terms of their expression profiles is an important problem and can be posed as a multiobjective clustering problem. There have been various studies in this area [Bandyopadhyay et al. 2007b; Maulik et al. 2009; Mukhopadhyay et al. 2009; Handl et al. 2005; Özyer et al. 2005]. Recently, an interactive version of a multiobjective clustering algorithm has been proposed in Mukhopadhyay et al. [2013], which periodically interacts with a human decision maker to automatically evolve the objective functions for any particular gene expression data. Multiobjective clustering has been applied in finding gene markers [Mukhopadhyay et al. 2010; Mukhopadhyay et al. 2010] and gene raking [Mondal et al. 2010] from expression data. Recently, multiobjective clustering has also been used in clustering protein–protein interaction networks [Mukhopadhyay et al. 2012].

9.3. Applications in Web Mining

Multiobjective clustering algorithms have also been applied in Web data mining. For example, in Demir et al. [2007], a Web-recommender system has been built by extracting Web usage patterns based on multiobjective clustering. The work has been extended in Demir et al. [2010], in which different multiobjective clustering approaches have been evaluated and compared for determining a suitable approach for clustering Web user sessions, which contain sequences of Web pages visited by the users.

9.4. Applications in Social Network Analysis

In the past few years, clustering social networks have gained popularity and a multitude of recent studies have employed multiobjective clustering in this regard for detecting strong communities within social networks. In Folino and Pizzuti [2010], a multiobjective clustering algorithm has been proposed for dynamic social network clustering. Another work with a similar objective is proposed in Kim et al. [2010] for developing a suitable multiobjective clustering algorithm for detecting clusters in dynamic time-varying social networks. Recently, a hybrid multiobjective evolutionary clustering algorithm has been proposed in Amiri et al. [2012] for clustering social networks.

9.5. Other Applications

In addition to these applications, multiobjective clustering algorithms have been applied in various other fields such as time series data analysis [Bandyopadhyay et al. 2010], software module clustering [Praditwong et al. 2011], security vulnerability

Table IV. Summary of Applications of Multiobjective Evolutionary Clustering

Applications	Tasks	References
Image Segmentation	Color image segmentation	[Shirakawa and Nagao 2009]
	Satellite image segmentation	[Bandyopadhyay et al. 2007a; Mukhopadhyay and Maulik 2009]
	Medical image segmentation	[Mukhopadhyay et al. 2009b; Mukhopadhyay and Maulik 2011]
	Texture image segmentation	[Qian et al. 2008]
Bioinformatics	Grouping co-expressed genes	[Maulik et al. 2009; Mukhopadhyay et al. 2013; Özyer et al. 2005]
	Clustering samples	[Handl et al. 2005; Mukhopadhyay et al. 2010]
	Protein complex identification	[Mukhopadhyay et al. 2012]
Web mining	Web-recommender system	[Demir et al. 2007, 2010]
Social Network Analysis	Social network clustering	[Folino and Pizzuti 2010; Kim et al. 2010; Amiri et al. 2012]
Other	Time series data analysis	[Bandyopadhyay et al. 2010]
	Software module clustering	[Praditwong et al. 2011]
	Security vulnerability assessment	[Corral et al. 2009]
	Skyline computation	[Özyer et al. 2011]

assessment [Corral et al. 2009], and skyline computation [Özyer et al. 2011]. In Table IV, different applications of multiobjective evolutionary clustering have been summarized.

10. DISCUSSION AND CONCLUSIONS

In the past decade, a multitude of multiobjective evolutionary clustering algorithms have come out. However, there has been no previous systematic effort to make a survey of these algorithms. In this article, we have made such an effort to review these algorithms with respect to many issues. We have first introduced the basic concepts of multiobjective optimization and the basic framework of posing the clustering as a multiobjective optimization problem. Subsequently, we have surveyed a number of multiobjective clustering algorithms with respect to five issues: (1) chromosome encoding policies, (2) objective functions for optimization, (3) evolutionary operators, (4) storing current nondominated solutions, and (5) selection of the final solution from the nondominated front.

There are two broad ways of representing chromosomes in the multiobjective evolutionary clustering algorithms: prototype-based representation and point-based representation. Each type has its own variations. As a whole, for prototype-based representation, the chromosomes are relatively smaller and less redundant compared to that for point-based representation. Thus the application of genetic operators is quicker. However, the prototype-based representation tends to capture round-shaped clusters since each cluster is represented by a single prototype. On the other hand, point-based encoding helps capture clusters with arbitrary shape. We have discussed the different variations of both these encoding strategies with proper references.

Second, the choice of objective functions is an important issue in multiobjective clustering. The algorithms mainly try to optimize a set of cluster validity indices simultaneously. Validity indices can either be based on cluster prototypes or based on the complete cluster labels of the data points. Most of the algorithms have been found to optimize two validity indices. However, a few algorithms are also found to optimize three or four validity indices simultaneously. We have discussed, with suitable references, the combination of validity indices that have been optimized in different existing multiobjective clustering algorithms.

We have described different evolutionary operators such as selection, crossover, and mutation used by various algorithms. The selection operator mainly depends on the multiobjective evolutionary algorithm used as the underlying optimization tool for the clustering algorithm. We have found four different multiobjective evolutionary algorithms: NSGA-II [Deb et al. 2002], PESA-II [Corne et al. 2001], NPGA [Horn and Nafpliotis 1993], and SPEA2 [Zitzler et al. 2001] for this purpose. The crossover and mutation operators are mainly dependent on the encoding strategy used by the corresponding algorithm. We have identified various approaches for crossover and mutation, and discussed those with proper citations.

We have discussed the different approaches for keeping track of the current nondominated front. This mainly depends on the multiobjective evolutionary algorithm that has been used. In NSGA-II and NPGA, the nondominated solutions are kept in the main population itself, whereas, in the case of PESA-II and SPEA2, a separate external archive is used for this purpose. The advantages and disadvantages of these two approaches have been discussed.

One of the most important attributes, for which the different multiobjective clustering algorithms differ with each other, is the method of selection of the final solution. As multiobjective evolutionary algorithms provide a set of nondominated clustering solutions in the final generation, it is important to devise a technique to obtain one solution from that set. Three primary techniques—independent objective-based, knee-based, and ensemble-based methods—have been used for this purpose in various multiobjective clustering algorithms. All these methods, with their variants, have been discussed in this article with proper references. We have also summarized the encoding policies, objective functions, evolutionary operators, and final solution selection of some popular multiobjective evolutionary clustering algorithms in Table I and Table II. In Table III, the major landmarks in the multiobjective evolutionary clustering research have been summarized in chronological order.

Most of the multiobjective evolutionary algorithms used for clustering are in fact genetic algorithm-based approaches. However, a few other kinds of evolutionary algorithms such as differential evolution, genetic programming, and gene expression programming, have also been utilized for multiobjective clustering. We have discussed some of these approaches in subsequent sections. Moreover, we have also mentioned a few other metaheuristics such as simulated annealing, particle swarm optimization, and immune response system applied to multiobjective clustering for the sake of completeness.

Finally, we have described various application fields in which the existing multiobjective clustering algorithms have been applied. In this regard, we have mentioned the multiobjective evolutionary clustering algorithms used in the field of image segmentation, bioinformatics, Web data mining, and social networking. The applications have been summarized in Table IV.

From the discussions in this article, it may be noted that the existing multiobjective evolutionary algorithms differ from one another in various aspects. Although a variety of multiobjective clustering algorithms exist in literature, a systematic and comprehensive comparative study among the algorithms still needs to be done. Moreover, comparison between different encoding schemes, choice of objective functions and evolutionary operators, strategy for storing the current nondominated solutions, and selection of final solution from the nondominated front are important topics for future work. A systematic comparison of the performance can shed light on the use of different algorithms in different suitable situations. Moreover, evolutionary algorithms, in particular multiobjective ones, are criticized for their time consumption. Therefore it is also important to perform research on designing more efficient multiobjective evolutionary algorithms for large-scale clustering problems. Furthermore, classical multiobjective evolutionary algorithms are known to perform badly for many-objective optimization [Saxena et al. 2013]. Therefore, optimizing higher numbers of validity indices simultaneously is not possible using these algorithms. Thus research dedicated to this direction, particularly for objective reduction, is needed. Furthermore, this is the age of big data [Fahad et al. 2014]. Many real-life applications are generating huge amounts of data. Since the existing multiobjective evolutionary algorithms are not well scalable to such large volumes of data, it is a challenge for researchers to devise fast scalable algorithms for multiobjective clustering. It can be concluded that although there has been a sustained development in the area of multiobjective evolutionary algorithms for clustering in the past decade, still there are lots of open areas that deserve serious attention from the researchers.

REFERENCES

H. A. Abbass and R. A. Sarker. 2002. The Pareto differential evolution algorithm. *International Journal on Artificial Intelligence Tools* 11, 4, 531–552.

S. Agrawal, B. K. Panigrahi, and M. K. Tiwari. 2008. Multiobjective particle swarm algorithm with fuzzy clustering for electrical power dispatch. *IEEE Transactions on Evolutionary Computation* 12, 5, 529–541.

B. Amiri, L. Hossain, and J. Crowford. 2012. A multiobjective hybrid evolutionary algorithm for clustering in social networks. In *Proceedings of the 14th International Conference on Genetic and Evolutionary Computation Conference Companion (GECCO Companion'12)*. ACM, New York, NY, 1445–1446.

S. Bandyopadhyay, R. Baragona, and U. Maulik. 2010. Clustering multivariate time series by genetic multiobjective optimization. *Metron - International Journal of Statistics* LXVIII, 2, 161–183.

S. Bandyopadhyay, U. Maulik, and A. Mukhopadhyay. 2007a. Multiobjective genetic clustering for pixel classification in remote sensing imagery. *IEEE Transactions on Geoscience and Remote Sensing* 45, 5, 1506–1511.

S. Bandyopadhyay, A. Mukhopadhyay, and U. Maulik. 2007b. An improved algorithm for clustering gene expression data. *Bioinformatics* 23, 21, 2859–2865.

S. Bandyopadhyay and S. K. Pal. 2007. *Classification and Learning Using Genetic Algorithms: Applications in Bioinformatics and Web Intelligence*. Springer, New York, NY.

S. Bandyopadhyay, S. Saha, U. Maulik, and K. Deb. 2008. A simulated annealing-based multiobjective optimization algorithm: AMOSA. *IEEE Transactions on Evolutionary Computation* 12, 3, 269–283.

W. Banzhaf, F. D. Francone, R. E. Keller, and P. Nordin. 1998. *Genetic Programming: An Introduction: on the Automatic Evolution of Computer Programs and Its Applications*. Morgan Kaufmann, San Francisco, CA.

N. Beume, B. Naujoks, and M. Emmerich. 2007. SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research* 181, 3, 1653–1669.

J. C. Bezdek. 1981. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum, New York, NY.

J. C. Bezdek and N. R. Pal. 1998. Some new indexes of cluster validity. *IEEE Transactions on Systems, Man and Cybernetics* 28, 301–315.

R. Caballero, M. Laguna, R. Marti, and J. Molina. 2006. *Multiobjective clustering with metaheuristic optimization technology*. Technical Report. Leeds School of Business in the University of Colorado, Boulder, CO.

E. Chen and F. Wang. 2005. Dynamic clustering using multi-objective evolutionary algorithm. In *Proceedings of the 2005 International Conference on Computational Intelligence and Security - Volume Part I (CIS'05)*. Springer-Verlag, Berlin, 73–80.

A. L. V. Coelho, E. Fernandes, and K. Faceli. 2010. Inducing multi-objective clustering ensembles with genetic programming. *Neurocomputing* 74, 1–3, 494–498.

C. A. Coello Coello. 2006. Evolutionary multiobjective optimization: A historical view of the field. *IEEE Computational Intelligence Magazine* 1, 1, 28–36.

C. A. Coello Coello, G. B. Lamont, and D. A. van Veldhuizen. 2007. *Evolutionary Algorithms for Solving Multi-Objective Problems* (2nd ed.). Springer, Berlin.

C. A. Coello Coello. 1999. A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge and Information Systems* 1, 3, 129–156.

D. W. Corne, N. R. Jerram, J. D. Knowles, and M. J. Oates. 2001. PESA-II: Region-based selection in evolutionary multiobjective optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S.

Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke (Eds.). Morgan Kaufmann, San Francisco, CA, 283–290.

D. W. Corne, J. D. Knowles, and M. J. Oates. 2000. The Pareto envelope-based selection algorithm for multiobjective optimization. In *Proceedings of the Parallel Problem Solving from Nature VI Conference*. Springer, 839–848.

G. Corral, A. Garcia-Piquer, A. Orriols-Puig, A. Fornells, and E. Golobardes. 2009. Multiobjective evolutionary clustering approach to security vulnerability assesments. In *Proceedings of the 4th International Conference on Hybrid Artificial Intelligence Systems (HAIS'09) (Lecture Notes in Computer Science)*, Emilio Corchado, Xindong Wu, Erkki Oja, lvaro Herrero, and Bruno Baruque (Eds.), Vol. 5572. Springer, 597–604.

D. L. Davies and D. W. Bouldin. 1979. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 1, 224–227.

K. Deb. 2001. *Multi-objective Optimization Using Evolutionary Algorithms*. John Wiley and Sons, England.

K. Deb, A. Pratap, S. Agrawal, and T. Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6, 182–197.

G. N. Demir, A. S. Uyar, and S. G. Ögüdücü. 2007. Graph-based sequence clustering through multiobjective evolutionary algorithms for web recommender systems. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation (GECCO'07)*. ACM, New York, NY, 1943–1950.

G. N. Demir, A. S. Uyar, and S. G. Ögüdücü. 2010. Multiobjective evolutionary clustering of Web user sessions: A case study in Web page recommendation. *Soft Computing* 14, 6, 579–597.

J. Du, E. E. Korkmaz, R. Alhajj, and K. Barker. 2005. Alternative clustering by utilizing multi-objective genetic algorithm with linked-list based chromosome encoding. In *MLDM (Lecture Notes in Computer Science)*, Vol. 3587. Springer, 346–355.

J. C. Dunn. 1974. Well separated clusters and optimal fuzzy partitions. *J. Cyberns.* 4 (1974), 95–104.

K. Faceli, M. C. P. de Souto, and A. C. P. L. F. de Carvalho. 2008. A strategy for the selection of solutions of the Pareto front approximation in multi-objective clustering approaches. In *Proceedings of the 2008 10th Brazilian Symposium on Neural Networks (SBRN'08)*. IEEE Computer Society, Washington, DC, 27–32.

A. Fahad, N. Alshatri, Z. Tari, A. Alamri, I. Khalil, A.Y. Zomaya, S. Foufou, and A. Bouras. 2014. A survey of clustering algorithms for big data: Taxonomy and empirical analysis. *IEEE Transactions on Emerging Topics in Computing* 2, 3, 267–279.

C. Ferreira. 2001. Gene expression programming: A new adaptive algorithm for solving problems. *Complex Systems* 13, 2, 87–129.

F. Folino and C. Pizzuti. 2010. A multiobjective and evolutionary clustering method for dynamic networks. In *Proceedings of the International Conference on Advances in Social Networks Analysis and Mining (ASONAM'10)*. IEEE Computer Society, 256–263.

C. M. Fonseca and P. J. Fleming. 1993. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In *Proceedings of the 5th International Conference on Genetic Algorithms*. Morgan Kaufmann, 416–423.

A. A. Freitas. 2004. A critical review of multi-objective optimization in data mining: A position paper. *SIGKDD Exploration Newsletter* 6, 2, 77–86.

A. P. Gasch and M. B. Eisen. 2002. Exploring the conditional coregulation of yeast gene expression through fuzzy k-means clustering. *Genome Biology* 3, 11, 0059.1–0059.22.

D. E. Goldberg. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, New York.

D. E. Goldberg and P. Segrest. 1987. Finite Markov chain analysis of genetic algorithms. In *Proceedings of the 2nd International Conference on Genetic Algorithms and Their Application*. L. Erlbaum Associates Inc., Hillsdale, NJ, 1–8.

M. Gong, L. Zhang, L. Jiao, and S. Gou. 2007. Solving multiobjective clustering using an immune-inspired algorithm. In *IEEE Congress on Evolutionary Computation*. 15–22.

L. Groll and J. Jakel. 2005. A new convergence proof of fuzzy c-means. *IEEE Transactions on Fuzzy Systems* 13, 5, 717–720.

M. Halkidi, Y. Batistakis, and M. Vazirgiannis. 2001. On clustering validation techniques. *Journal of Intelligent Information Systems* 17, 2/3, 107–145.

M. Halkidi, M. Vazirgiannis, and Y. Batistakis. 2000. Quality scheme assessment in the clustering process. In *Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD'00)*. Springer-Verlag, London, 265–276.

J. Handl and J. D. Knowles. 2004. Evolutionary multiobjective clustering. In *Proceedings of the 8th International Conference on Parallel Problem Solving in Nature (PPSN'04)*. 1081–1091.

J. Handl and J. D. Knowles. 2005a. Exploiting the trade-of—the benefits of multiple objectives in data clustering. In *Proceedings of the 3rd International Conference on Evolutionary Multi-Criterion Optimization (EMO'05)*. Springer-Verlag, Berlin, 547–560.

J. Handl and J. D. Knowles. 2005b. Improvements to the scalability of multiobjective clustering. In *Proceedings of the Congress on Evolutionary Computation (IEEE CEC'05)*. IEEE, 2372–2379.

J. Handl and J. D. Knowles. 2005c. Multiobjective clustering around medoids. In *Proceedings of the IEEE Congress on Evolutionary Computation*, Vol. 1. 632–639.

J. Handl and J. D. Knowles. 2006. Multiobjective clustering and cluster validation. *Computational Intelligence*, Vol. 16. Springer, 21–47.

J. Handl and J. D. Knowles. 2007. An evolutionary approach to multiobjective clustering. *IEEE Transactions on Evolutionary Computation* 11, 1, 56–76.

J. Handl and J. D. Knowles. 2012. Clustering criteria in multiobjective data clustering. In *Parallel Problem Solving from Nature - PPSN XII*, C. A. Coello Coello, V. Cutello, K. Deb, S. Forrest, G. Nicosia, and M. Pavone (Eds.). Lecture Notes in Computer Science, Vol. 7492. Springer, Berlin, 32–41.

J. Handl, J. D. Knowles, and D. B. Kell. 2005. Computational cluster validation in post-genomic data analysis. *Bioinformatics* 21, 15, 3201–3212.

J. Holland. 1975. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI.

J. Horn and N. Nafpliotis. 1993. *Multiobjective Optimization Using Niched Pareto Genetic Algorithm*. Technical Report IlliGAL Report 93005. University of Illinois at Urbana-Champaign, Urbana, IL.

E. R. Hruschka, R. J. G. B. Campello, A. A. Freitas, and A. C. P. L. F. De Carvalho. 2009. A survey of evolutionary algorithms for clustering. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews* 39, 2, 133–155.

A. W. Iorio and X. Li. 2004. Solving rotated multi-objective optimization problems using differential evolution. In *Australian Conference on Artificial Intelligence (Lecture Notes in Computer Science)*, G. I. Webb and X. Yu (Eds.), Vol. 3339. Springer, 861–872.

A. K. Jain and R. C. Dubes. 1988. *Algorithms for Clustering Data*. Prentice-Hall, Englewood Cliffs, NJ.

A. K. Jain, M. N. Murty, and P. J. Flynn. 1999. Data clustering: A review. *Computing Surveys* 31.

K. Kim, R. I. McKay, and B.-R. Moon. 2010. Multiobjective evolutionary algorithms for dynamic social network clustering. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation (GECCO'10)*. ACM, New York, NY, 1179–1186.

O. Kirkland, V. Rayward-Smith, and B. de la Iglesia. 2011. A novel multi-objective genetic algorithm for clustering. In *Intelligent Data Engineering and Automated Learning (IDEAL'11)*, H. Yin, W. Wang, and V. Rayward-Smith (Eds.). Lecture Notes in Computer Science, Vol. 6936. Springer, Berlin, 317–326.

J. D. Knowles and D. W. Corne. 1999. The Pareto archived evolution strategy: A new baseline algorithm for Pareto multiobjective optimisation. In *Proceedings of the IEEE Congress on Evolutionary Computation.* IEEE Press, Piscataway, NJ, 98–105.

L. I. Kuncheva and J. C. Bezdek. 1997. Selection of cluster prototypes from data by a genetic algorithm. In *Proceedings of the 5th European Conference on Intelligent Techniques and Soft Computing*. 1683–1688.

D. Kundu, K. Suresh, S. Ghosh, S. Das, A. Abraham, and Y. Badr. 2009. Automatic clustering using a synergy of genetic algorithm and multi-objective differential evolution. In *Proceedings of the 4th International Conference on Hybrid Artificial Intelligence Systems (HAIS'09)*. Springer-Verlag, Berlin, 177–186.

R. Liu, W. Zhang, L. Jiao, and F. Liu. 2010. A multiobjective immune clustering ensemble technique applied to unsupervised SAR image segmentation. In *CIVR*. 158–165.

Y. Liu, T. Özyer, R. Alhajj, and K. Barker. 2005. Integrating multi-objective genetic algorithm and validity analysis for locating and ranking alternative clustering. *Informatica* 29, 33–40.

N. Matake, T. Hiroyasu, M. Miki, and T. Senda. 2007. Multiobjective clustering with automatic k-determination for large-scale data. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation (GECCO'07)*. ACM, New York, NY, 861–868.

U. Maulik and S. Bandyopadhyay. 2000. Genetic algorithm based clustering technique. *Pattern Recognition* 33, 1455–1465.

U. Maulik and S. Bandyopadhyay. 2002. Performance evaluation of some clustering algorithms and validity indices. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24, 12, 1650–1654.

U. Maulik, S. Bandyopadhyay, and A. Mukhopadhyay. 2011. *Multiobjective Genetic Algorithms for Clustering - Applications in Data Mining and Bioinformatics*. Springer.

U. Maulik, A. Mukhopadhyay, and S. Bandyopadhyay. 2009. Combining Pareto-optimal clusters using supervised learning for identifying co-expressed genes. *BMC Bioinformatics* 10, 27.

U. Maulik, A. Mukhopadhyay, S. Bandyopadhyay, M. Q. Zhang, and X. Zhang. 2008. Multiobjective fuzzy biclustering in microarray data: Method and a new performance measure. In *Proceedings of the IEEE World Congress on Computational Intelligence (WCCI'08)/IEEE Congress on Evolutionary Computation (CEC'08)*. 383–388.

K. C. Mondal, A. Mukhopadhyay, U. Maulik, S. Bandyopadhyay, and N. Pasquier. 2010. MOSCFRA: A multiobjective genetic approach for simultaneous clustering and gene ranking. In *CIBB*. 174–187.

A. Mukhopadhyay, S. Bandyopadhyay, and U. Maulik. 2006. Clustering using multi-objective genetic algorithm and its application to image segmentation. *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC'06)* 3, 2678–2683.

A. Mukhopadhyay, S. Bandyopadhyay, and U. Maulik. 2008. Combining multiobjective fuzzy clustering and probabilistic ANN classifier for unsupervised pattern classification: Application to satellite image segmentation. In *Proceedings of the IEEE World Congress on Computational Intelligence (WCCI 2008)/IEEE Congress on Evolutionary Computation (CEC'08)*. 877–883.

A. Mukhopadhyay, S. Bandyopadhyay, and U. Maulik. 2009. Analysis of microarray data using multiobjective variable string length genetic fuzzy clustering. In *Proceedings of the 11th Conference on Congress on Evolutionary Computation (CEC'09)*. IEEE Press, Piscataway, NJ, 1313–1319.

A. Mukhopadhyay, S. Bandyopadhyay, and U. Maulik. 2010. Multi-class clustering of cancer subtypes through SVM based ensemble of Pareto-optimal solutions for gene marker identification. *PloS One* 5, 11, e13803.

A. Mukhopadhyay and U. Maulik. 2007. Multiobjective approach to categorical data clustering. *Proc. IEEE Congress on Evolutionary Computation (CEC 2007)* (September 2007), 1296–1303.

A. Mukhopadhyay and U. Maulik. 2009. Unsupervised pixel classification in satellite imagery using multiobjective fuzzy clustering combined with SVM classifier. *IEEE Transactions on Geoscience and Remote Sensing* 47, 4, 1132–1138.

A. Mukhopadhyay and U. Maulik. 2011. A multiobjective approach to MR brain image segmentation. *Applied Soft Computing* 11, 872–880.

A. Mukhopadhyay, U. Maulik, and S. Bandyopadhyay. 2009a. Multi-objective genetic algorithm based fuzzy clustering of categorical attributes. *IEEE Transactions on Evolutionary Computation* 13, 5 (2009), 991–1005.

A. Mukhopadhyay, U. Maulik, and S. Bandyopadhyay. 2009b. Multiobjective genetic clustering with ensemble among Pareto front solutions: Application to MRI brain image segmentation. In *Proceedings of the International Conference on Advances in Pattern Recognition (ICAPR'09)*. 236–239.

A. Mukhopadhyay, U. Maulik, and S. Bandyopadhyay. 2010. Simultaneous informative gene selection and clustering through multiobjective optimization. In *IEEE Congress on Evolutionary Computation*. 1–8.

A. Mukhopadhyay, U. Maulik, and S. Bandyopadhyay. 2011. Gene expression data analysis using multiobjective clustering improved with SVM based ensemble. *In Silico Biology* 11, 1–2, 19–27.

A. Mukhopadhyay, U. Maulik, and S. Bandyopadhyay. 2013. An interactive approach to multiobjective clustering of gene expression patterns. *IEEE Transactions on Biomedical Engineering* 60, 1, 35–41.

A. Mukhopadhyay, U. Maulik, S. Bandyopadhyay, and C. A. Coello Coello. 2014a. A survey of multiobjective evolutionary algorithms for data mining: Part I. *IEEE Transactions on Evolutionary Computation* 18, 1, 4–19.

A. Mukhopadhyay, U. Maulik, S. Bandyopadhyay, and C. A. Coello Coello. 2014b. Survey of multiobjective evolutionary algorithms for data mining: Part II. *IEEE Transactions on Evolutionary Computation* 18, 1, 20–35.

A. Mukhopadhyay, S. Ray, and M. De. 2012. Detecting protein complexes in a PPI network: a gene ontology based multi-objective evolutionary approach. *Molecular Biosystems* 8, 11, 3036–3048.

T. Özyer, Y. Liu, R. Alhajj, and K. Barker. 2004. Multi-objective genetic algorithm based clustering approach and its application to gene expression data. In *Proceedings of the 3rd International Conference on Advances in Information Systems (ADVIS'04)*. Springer-Verlag, Berlin, 451–461.

T. Özyer, Y. Liu, R. Alhajj, and K. Barker. 2005. Multi-objective genetic algorithm based clustering approach and its application to gene expression data. In *Advances in Information Systems*, Tatyana Yakhno (Ed.). Lecture Notes in Computer Science, Vol. 3261. Springer, Berlin, 451–461.

T. Özyer, M. Zhang, and R. Alhajj. 2011. Integrating multi-objective genetic algorithm based clustering and data partitioning for skyline computation. *Applied Intelligence* 35, 1, 110–122.

M. K. Pakhira, S. Bandyopadhyay, and U. Maulik. 2004. Validity index for crisp and fuzzy clusters. *Pattern Recognition* 37, 487–501.

N. R. Pal and J. C. Bezdek. 1995. On cluster validity for the fuzzy C-means model. *IEEE Transactions on Fuzzy Systems* 3, 370–379.

K. Praditwong, M. Harman, and X. Yao. 2011. Software module clustering as a multi-objective search problem. *IEEE Transactions on Software Engineering* 37, 2, 264–282.

K. V. Price, R. M. Storn, and J. A. Lampinen. 2005. *Differential Evolution: A Practical Approach to Global Optimization*. Springer-Verlag, Berlin.

X. Qian, X. Zhang, L. Jiao, and W. Ma. 2008. Unsupervised texture image segmentation using multiobjective evolutionary clustering ensemble algorithm. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'08)*. 3561–3567.

K. S. N. Ripon and M. N. H. Siddique. 2009. Evolutionary multi-objective clustering for overlapping clusters detection. In *Proceedings of the 11th Conference on Congress on Evolutionary Computation (CEC'09)*. IEEE Press, Piscataway, NJ, 976–982.

K. S. N. Ripon, C.-H. Tsang, and S. Kwong. 2006a. Multi-objective data clustering using variable-length real jumping genes genetic algorithm and local search method. In *Proceedings of the International Joint Conference on Neural Networks*. 3609–3616.

K. S. N. Ripon, C.-H. Tsang, S. Kwong, and M.-K. Ip. 2006b. Multi-objective evolutionary clustering using variable-length real jumping genes genetic algorithm. In *Proceedings of the International Conference on Pattern Recognition (ICPR'06)*. 1200–1203.

T. Robic and B. Filipic. 2005. DEMO: Differential evolution for multiobjective optimization. *Lecture Notes in Computer Science*, C. A. Coello Coello, A. Hernandez Aguirre, and E. Zitzler (Eds.), Vol. 3410. Springer, 520–533.

P. J. Rousseeuw. 1987. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational Applied Mathematics* 20, 53–65.

I. Saha and U. Maulik. 2014. Multiobjective differential evolution-based fuzzy clustering for MR brain image segmentation. In *Advanced Computational Approaches to Biomedical Engineering*, P. K. Saha, U. Maulik, and S. Basu (Eds.). Springer, Berlin, 71–86.

I. Saha, U. Maulik, and D. Plewczynski. 2011a. Multiobjective differential crisp clustering for evaluation of clusters dynamically. In *Man-Machine Interactions 2*, T. Czachorski, S. Kozielski, and U. Stanczyk (Eds.). *Advances in Intelligent and Soft Computing*, Vol. 103. Springer, Berlin, 307–313.

I. Saha, U. Maulik, and D. Plewczynski. 2011b. A new multi-objective technique for differential fuzzy clustering. *Applied Soft Computing* 11, 2, 2765–2776.

S. Saha and S. Bandyopadhyay. 2009. A new multiobjective simulated annealing based clustering technique using symmetry. *Pattern Recognition Letters* 30, 15, 1392–1403.

S. Saha and S. Bandyopadhyay. 2013. A generalized automatic clustering algorithm in a multiobjective framework. *Applied Soft Computing* 13, 1, 89–108.

D. K. Saxena, J. A. Duro, A. Tiwari, K. Deb, and Q. Zhang. 2013. Objective reduction in many-objective optimization: Linear and nonlinear algorithms. *IEEE Transactions on Evolutionary Computation* 17, 1, 77–99.

H.-P. Schwefel. 1993. *Evolution and Optimum Seeking: The Sixth Generation*. John Wiley & Sons, New York, NY.

S. Z. Selim and M. A. Ismail. 1984. K-means type algorithms: A generalized convergence theorem and characterization of local optimality. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 6, 81–87.

W. Shannon, R. Culverhouse, and J. Duncan. 2003. Analyzing microarray data using cluster analysis. *Pharmacogenomics* 4, 1, 41–51.

S. Shirakawa and T. Nagao. 2009. Evolutionary image segmentation based on multiobjective clustering. In *Proceedings of the 11th Conference on Congress on Evolutionary Computation (CEC'09)*. IEEE Press, Piscataway, NJ, 2466–2473.

N. Srinivas and K. Deb. 1994. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation* 2, 3, 221–248.

A. Strehl and J. Ghosh. 2002. Cluster ensembles - a knowledge reuse framework for combining multiple partitions. In *Machine Learning Research*, Vol. 3. 583–617.

J. Sun, W. Sverdlik, and S. Tout. 2006. Parallel hybrid clustering using genetic programming and multi-objective fitness with density (PYRAMID). In *DMIN*, S. F. Crone, S. Lessmann, and R. Stahlbock (Eds.). CSREA Press, 197–203.

K. Suresh, D. Kundu, S. Ghosh, S. Das, and A. Abraham. 2009b. Data clustering using multi-objective differential evolution algorithms. *Fundamenta Informatica* 97, 4, 381–403.

K. Suresh, D. Kundu, S. Ghosh, S. Das, A. Abraham, and S. Y. Han. 2009a. Multi-objective differential evolution for automatic clustering with application to micro-array data analysis. *Sensors* 9, 5, 3981–4004.

R. Tibshirani, G. Walther, and T. Hastie. 2001. Estimating the number of clusters in a dataset via the Gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 63, 2, 411–423.

W. Wanga and Y. Zhanga. 2007. On fuzzy cluster validity indices. *Fuzzy Sets and Systems* 158, 19, 2095–2117.

J.-M. Won, S. Ullah, and F. Karray. 2008. Data clustering using multi-objective hybrid evolutionary algorithm. In *Proceedings of the International Conference on Control, Automation and Systems*. 2298–2303.

X. L. Xie and G. Beni. 1991. A validity measure for fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 13, 841–847.

F. Xue, A. C. Sanderson, and R. J. Graves. 2005. Multi-objective differential evolution - algorithm, convergence analysis, and applications. *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'05)* 1, 743–750.

Y. Zheng, L. Jia, and H. Cao. 2012. Multi-objective gene expression programming for clustering. *Information Technology and Control* 41, 3, 283–294.

L. Zhu, L. Cao, and J. Yang. 2012. Multiobjective evolutionary algorithm-based soft subspace clustering. In *IEEE Congress on Evolutionary Computation*. 1–8.

E. Zitzler. 1999. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. Ph.D. Dissertation. Swiss Federal Institute of Technology (ETH), Zurich, Switzerland.

E. Zitzler and S. Künzli. 2004. Indicator-based selection in multiobjective search. In *Parallel Problem Solving from Nature - PPSN VIII*, X. Yao et al. (Ed.). Springer-Verlag. *Lecture Notes in Computer Science* Vol. 3242, Birmingham, UK, 832–842.

E. Zitzler, M. Laumanns, and L. Thiele. 2001. *SPEA2: Improving the Strength Pareto Evolutionary Algorithm*. Technical Report 103. Universität Zürich, Zürich, Switzerland.

E. Zitzler and L. Thiele. 1998. *An Evolutionary Algorithm for Multiobjective Optimization: The Strength Pareto Approach*. Technical Report 43. Universität Zürich, Zürich, Switzerland.

Received November 2012; revised October 2014; accepted March 2015