Adaptive Multisubpopulation Competition and Multiniche Crowding-Based Memetic Algorithm for Automatic Data Clustering

Weiguo Sheng, *Member, IEEE*, Shengyong Chen, *Senior Member, IEEE*, Mengmeng Sheng, Gang Xiao, Jiafa Mao, and Yujun Zheng

***Abstract*—Automatic data clustering, whose goal is to recover the proper number of clusters as well as appropriate partitioning of data sets, is a fundamental yet challenging problem in unsupervised learning. In this paper, adaptive multisubpopulation competition (AMC) and multiniche crowding are proposed and incorporated into a memetic algorithm to tackle the problem. The AMC mechanism is developed to ensure a diverse search over solution subspaces corresponding to different numbers of clusters while allowing more promising subspaces to be more intensively searched. In this mechanism, the amount of individuals to be migrated between subpopulations is adaptively controlled according to the performance of subpopulations as well as the diversity of cluster numbers in population. Further, the migration is restricted to occur between subpopulations with relatively similar performances. Additionally, subpopulations with different performances are devised to search their corresponding subspaces with different exploration powers. The adaptive multiniche crowding scheme is designed to promote a diverse search of the subspace while allowing an efficient convergence of the corresponding subpopulation. This is achieved by dynamically adjusting parameter values of a multiniche crowding method to form and maintain diverged niches of high fitness within the subpopulation. The performance of proposed algorithm has been demonstrated through a series of experiments on both artificial and real data, and compared with existing methods. The results reveal that our proposed algorithm can achieve superior clustering performance and outperform related methods.**

***Index Terms*—Data clustering, genetic algorithm (GA), memetic algorithm (MA), multisubpopulation, niching method, parameter adaptation.**

I. Introduction

CLUSTERING, also known as cluster analysis, aims at partitioning a set of data objects into clusters on the basis of their similarity such that objects within the same cluster are similar while objects from different clusters are dissimilar. It is an important tool with applications ranging from astronomy to bioinformatics, machine learning, and pattern recognition. Numerous algorithms [45], [57], [62], [71], [96]–[98] have been proposed in literature. Generally, existing algorithms can be grouped into two categories: 1) hierarchical and 2) partitional clustering [45]. The hierarchical clustering seeks to arrange the data in a hierarchical tree structure based on the similarity between data objects. In this approach, once a data object is allocated to a cluster at an early stage, it cannot be reallocated to a different one at a later stage during clustering process, thus forming clusters in a static manner. Further, the size and global shape of clusters are ignored. In this paper, we focus on partitional clustering, which is a dynamic approach and takes into account the size as well as global shape of clusters. Traditionally, partitional clustering is dealt with under the assumption that the number of clusters existing in a data set is known beforehand. Since such knowledge is generally not available, recovering the proper number of clusters is also required. In such a situation, the problem is often referred to as automatic data clustering, which is our concern in this paper.

For a data set with nontrivial size, it has been found difficult to identify the optimal clustering solution [46]. The problem is made even more difficult when the proper cluster number in the data set is unknown beforehand. The classical approach for identifying the number of clusters in a data set is to apply a given clustering algorithm iteratively over a range of number of clusters and evaluate results using a certain validity criterion [33], [50]. The result, which gives the best validity value, is then selected as the solution. This approach for determining the cluster number in a data set depends on the chosen clustering algorithm, whose performance is usually highly sensitive to initial cluster centers. Since optimal values of validity criteria would correspond to the most valid solutions in terms of the criteria, stochastic optimization techniques have been widely employed to optimize them for simultaneously determining the cluster number as well as the partitioning of a data set [42], [45]. Among various stochastic optimization techniques, a prominent approach is to use genetic algorithms (GAs) [36]. For example, Maulik and Bandyopadhyay [60], Tseng and Yang [90], and Ma *et al.* [57] described several methods based on traditional

Manuscript received August 22, 2014; revised May 7, 2015, August 15, 2015, and November 3, 2015; accepted January 18, 2016. Date of publication February 3, 2016; date of current version November 23, 2016. This work was supported in part by the National Natural Science Foundation of China under Grant 61573316, Grant 61203288, Grant U1509207, Grant 61325019, Grant 61272310, and Grant 61473263, and in part by the Zhejiang Provincial Natural Science Foundation of China under Grant LY15F020032.

W. Sheng, G. Xiao, J. Mao, and Y. Zheng are with Zhejiang University of Technology, Hangzhou 310023, China (e-mail: wsheng@zjut.edu.cn; xg@zjut.edu.cn; maojiafa@zjut.edu.cn; jzheng@zjut.edu.cn).

S. Chen is with Tianjin University of Technology, Tianjin, China, and also with the Zhejiang University of Technology, Hangzhou 310023, China (e-mail: sy@ieee.org).

M. Sheng is with Zhejiang Police College, Hangzhou 310053, China (e-mail: shengmengmeng@zjjcxy.cn).

Digital Object Identifier 10.1109/TEVC.2016.2524555

GAs for automatic data clustering. Nevertheless, traditional GAs suffer from excessively slow convergence. As a result, these methods can take a considerable amount of time to deliver clustering solutions. Recently, hybrid GAs, which incorporate clustering-specific local search operations into GAs, have been developed for automatic data clustering [32], [41], [56], [80]. These studies reveal that the search efficiency of traditional GA-based data clustering methods can be greatly improved by hybridizing them with the local search operation. In literature, hybrid GAs are commonly known as memetic algorithms (MAs) [18], a term which will also be adopted in this paper.

For optimization problems characterized with few local optima, GAs and MAs are generally able to locate the optimal solution. However, they tend to prematurely converge to local optimal solutions on complex optimization problems (such as clustering of large data sets), which may involve a huge number of local optima. This is largely due to GAs and MAs have the difficulty to maintain population diversity during evolution [23], [27]. To deal with this situation, various approaches including multisubpopulation methods [8], [55] and niching methods [24], [74] have been proposed. The premise of multipopulation methods lies in partitioning the population into several subpopulations, which alternate extensive periods of isolated evolution with occasionally episodes of information exchange. By maintaining multiple subpopulations in different subareas of the search space, this approach can improve the population diversity. However, the multiple subpopulations in existing methods are typically used to encode solutions of the same problem and each subpopulation can move around the entire search space. In such cases, several issues (e.g., how to create an appropriate number of subpopulations and how to maintain them in different subareas), which are central to the success of the approach, could be difficult to address.

The niching methods, on the other hand, try to preserve population diversity by fostering niches within the population and are probably the most widely used diversity-preserving approach. Existing niching methods, however, usually require carefully tuning certain parameters in order to obtain a good performance. It is now well understood that there is no one optimal setting for these parameters—rather the situation changes per problem instance. Moreover, optimal parameter values may vary during the run of the method. Setting right values for these parameters is, thus, a hard task [47]. Additionally, these niching methods are typically designed with the purpose to locate multiple optima, regardless of their goodness. Consequently, niches formed in the population based on these methods are not necessarily diverged ones of high fitness. This may significantly reduce their performances in locating the optimal or near optimal solution.

To confront premature convergence, niching-based GAs (NGAs) [21], [44] or niching-based MAs (NMAs) [78] and multisubpopulation-based evolutionary algorithms (MSEAs) [34], [53] have recently been developed for automatic data clustering. Compared with traditional GA/MA-based methods, these schemes can generally deliver clustering solutions with higher quality. However, existing NGA- and NMA-based clustering methods usually employ traditional niching techniques or their variants, which are not able to promote diverged niches of high fitness. Further, parameter values of niching techniques employed in these methods are typically arbitrarily set and remain constant during the run of algorithm. As a result, the search efficiency of these methods can be limited. Existing MSEA-based clustering methods, on the other hand, are usually based on an approach, in which multiple subpopulations are used to encode solutions of the same problem, thus suffering from the issues mentioned above. Further, subpopulations in these methods are designed to work cooperatively to search the solution space while the competition aspect of subpopulations is largely ignored. This, consequently, could render them inefficient and ineffective for automatic data clustering.

From the optimization perspective, the search space of automatic clustering problem can be decomposed into disjoint subspaces, corresponding to different numbers of clusters. By encoding solutions with various numbers of clusters, multiple subpopulations are naturally formed in population, each of which is used to search one particular subspace. Taking the advantage of these naturally formed subpopulations, in this paper, we first propose an adaptive multisubpopulation competition (AMC) mechanism to achieve a diverse search over the subspaces corresponding to different numbers of clusters while allowing promising subspaces to be intensively searched. In the proposed mechanism, the amount of individuals to be migrated between subpopulations at each generation is designed to adaptively vary based on the performance of subpopulations as well as the diversity of cluster numbers in population. Further, we restrict the migration operation to occur only between subpopulations with relatively similar performances. As a result, a diverse search over different subspaces at an early stage and then smoothly shifting to an intensive search of promising subspaces at a later stage of evolution can be achieved. Additionally, subpopulations with low performances are devised to search their corresponding subspaces more aggressively. This is accomplished by dynamically setting the crossover and mutation rates for different subpopulations according to their performance rankings. Finally, to facilitate the migration of individuals between subpopulations with different numbers of clusters, two operators have also been introduced to appropriately adjust the number of clusters encoded in the individuals before migration.

The above mechanism can be used to maintain the diversity with respect to the cluster numbers in population. Such a mechanism, however, may not necessary result in diverse solutions with the same number of clusters in each subpopulation. To promote a diverse search of the subspace while allowing efficient convergence of the corresponding subpopulation, an adaptive niching scheme has been further developed and incorporated into the proposed algorithm. The developed adaptive niching scheme is based on a variant of the multiniche crowding (VMNC) method [16]. To adaptively adjust parameter values of this niching method, a diversity index has been introduced, which measures the fitness-weighted genotypic diversity of subpopulation. As the index can be employed to promote both genotypic diversity and fitness, diverged niches of high fitness can be formed and maintained in the subpopulation during evolution, thus permitting a diverse while efficient search of the corresponding subspace. Based on the above AMC mechanism and adaptive niching scheme, an MA is finally proposed to effectively and efficiently search the solution space of automatic clustering problem. The performance of the proposed algorithm has been evaluated using both artificial and real data and compared with related methods. The results clearly confirm the significance of AMC mechanism and adaptive niching scheme for automatic data clustering.

It should be mentioned that, by incorporating the AMC mechanism and adaptive multiniche crowding scheme, the proposed method represents a significantly extension of our previously developed evolutionary algorithm-based clustering methods in [78]–[81]. Our previous methods are based on the panmictic model, in which the whole population is dealt with as a single pool of individuals. Further, although niching methods have been employed in [79]–[81] for data clustering. These niching methods consider neither dynamically adjusting their parameter values during evolution nor promoting diverged niches of high fitness. While in [78], an adaptive niching method has been devised and employed to evolve the clusters. However, this adaptive niching method is based on the restricted tournament selection technique [40], which has a different rationale and property for inducing niching behavior in evolutionary algorithms as compared to the multiniche crowding method [16]. The major contribution of this paper is therefore developing an AMC mechanism and adaptive multiniche crowding scheme-based MA for effectively and efficiently data clustering.

The rest of this paper proceeds as follows. Section II states the partitional clustering problem and provides a review of related work. An overview of the proposed algorithm is presented in Section III. Sections IV and V describe details of the AMC mechanism and adaptive niching scheme, respectively. The performance of proposed algorithm is evaluated and compared with related work in Section VI. Finally, Section VII concludes this paper with a summary and suggests areas of future work.

II. Partitional Data Clustering and Related Work

Consider a set of n data objects, $X = \{x_1, x_2, \dots, x_n\}$, to be partitioned into K clusters. Each x_i denotes a data object consisting of d real-valued features describing the data object. Partitional clustering deals with the partition matrix $U = [u_{ki}]_{K\times n}$, where u_{ki} represents the membership of data object, x_i, to cluster $C_k(k = 1, \dots, K)$. According to whether the binary or gradual membership is used, partitional clustering algorithms can be classified into two types: 1) hard clustering, in which each object is assigned to only one cluster and 2) fuzzy clustering, in which each object belongs to every cluster with some weight between 0 and 1. Irrespective of the type of algorithm, the goal of partitional clustering is to identify a partition matrix, which can maximize both heterogeneity among different clusters and homogeneity within the same cluster [46].

To deal with partitional data clustering, many methods have been proposed in literature, such as fuzzy c-means [3], [64], k-means [7], [58], self-organizing map-based methods [12], [94], and model-based methods [17], [61]. Most of these methods assume that the number of clusters existing in the data set is known beforehand. Further, they generally employ alternative optimization schemes, which are sensitive to initialization and may easily trap into local optima. Others (e.g., self-organizing map based schemes) do not require the knowledge of the number of clusters. However, they still depend on initial conditions and typically deliver local optimal clustering solutions. Since partitional data clustering can be viewed as a particular case of NP-hard problem [42], [46], this has also stimulated the use of general meta-heuristics [70]. GA and its variants, which are known to be effective for NP-hard problems, have therefore been widely employed to approach the problem. Some of the early work can be found, for instance, in [38], [43], [49], and [60]. These methods generally differ in the encoding scheme (to represent the clustering solution) and/or genetic operators (to generate new candidate solutions). Hruschka and Ebecken [43] devised a method attempting to improve the grouping GA proposed by Falkenauer [30]. In this method, an integer array is used to represent the clustering solution, such that the ith value in the array denotes the cluster membership of ith object. Both mutation and crossover operation in the method are redefined to cope with redundancy and context insensitivity issues of the encoding scheme. Korkmaz [49] designed a crossover operator called group-crossover to exploit the potential of using linear linkage encoding (LLE) scheme [26] for automatic data clustering. The LLE encoding scheme is also based on an integer array. However, the ith value, say j, in the array is interpreted as a link between data objects i and j, and they will be assigned to the same cluster in the resulting solution. Handl and Knowles [38] adopted the LLE scheme with the uniform crossover and neighborhood-biased mutation to perform automatic data clustering by simultaneously optimizing two clustering criteria. While in [60], a real-value array, which is used to represent the cluster centers, was employed as the underlying encoding scheme to search for clusters. In this method, the crossover operation is devised to work by exchanging randomly selected subarrays. Comparing to the clustering methods mentioned above, the GA based clustering methods usually require more computational time. However, the reward of such an approach is that it can generally deliver more promising clustering solutions and makes no *a priori* assumption about the number of clusters. Recently, MAs, that hybrid GAs with local improvement operations, have gained popularity for data clustering. Some of these methods are developed to optimize partitions of the data set for a fixed number of clusters [42], [45], [56] while others aim at simultaneously optimizing the number of clusters and corresponding partitions [41], [80], [93]. In these methods, different local search operators [1], [66] have been introduced to design the MAs, among which the k-means algorithm based operator is perhaps the most commonly used. This operator has been incorporated into traditional GAs in various ways: 1) to improve initial solutions [93]; 2) to fine-tune solutions during evolution [41], [80]; and 3) to substitute the crossover operation [56]. Among these alternatives, the MA which employs the *k*-means algorithm-based operator to fine-tune solutions during evolution has appeared to be the most efficient [42], [45]. Such an MA has thus been adopted as the base for our proposed algorithm in this paper. Contrary to the GA-based clustering approach, MA-based clustering can be more efficient. However, they both suffer from premature convergence.

The premature convergence is a major issue that arises when applying GAs or MAs to complex problems. This issue is mainly caused by the rapid loss of population diversity. Several approaches have been developed to circumvent such an issue by preserving the population diversity during evolution. In the following sections, we review two representative approaches: 1) multisubpopulation methods and 2) niching methods, on which we are focusing in this paper.

A. Multisubpopulation Methods

Multisubpopulation methods manipulate two or more subpopulations. The island model [8], [88], which is closely related to our proposed mechanism, is perhaps the most prevalent model of multisubpopulation approach. In this model, the entire population is divided into several subpopulations, each of which is processed by an isolated GA. During evolution, a certain migration strategy is performed periodically to exchange individuals between subpopulations. The behavior and performance of this model, however, depends on specific elements of the migration strategy used, such as the policy for selecting individuals sent to other subpopulations and to be replaced by received ones as well as the migration size, interval and topology. Consequently, the migration strategy has frequently been the subject of study and many island-model-based multisubpopulation methods have been developed, which differs in certain aspects of their migration strategies. For example, the work presented in [9], [15], [29], and [65] mainly focus on the selection policy: studying the performance of different policies [15], devising new policies [9], [29], and building an adaptive policy [65]. In [10] and [73], the topology issue of migration strategy has been investigated, while the work of [4] and [59] studied the migration size and interval aspects of migration strategy.

The subpopulations in above methods are designed to work cooperatively to search the space, by exchanging individuals among them. The number and size of subpopulations in these methods are kept unchanged during evolution. This, consequently, could significantly reduce their search efficiency to identify the optimal or near optimal solution. Nevertheless, it is possible to let the number and/or size of subpopulations vary according to their characteristics during evolution, thus making them to search the space competitively. A few studies have been carried out toward this direction. Schlierkamp-Voosen and Mühlenbein [91] presented a method, in which the size of subpopulation with the best quality, in terms of a measure called gain criterion, is set to increase while all other subpopulations are reduced if their sizes are greater than a certain lower limit. In this method, each subpopulation loses the same percentage of its individuals. The lower limit of subpopulation size is used to prevent the subpopulation from becoming extinct. Similarly, Srinivasa *et al.* [84] developed a scheme to vary the subpopulation size based on the fitness of the best individual compared to the mean fitness of entire population. Lardeux and Goëffon [51] designed a framework, in which the migration probability of each direction between subpopulations is dynamically decided according to the effect of migration. If a subpopulation, which receives an individual, observes any improvement in terms of the mean fitness then the corresponding migration probability increases, otherwise it is decreased. This framework was later used for adaptively selecting recombination operators in evolutionary algorithms [14]. The idea is to assign different operators to different subpopulations and distribute individuals to subpopulations according to their performances during evolution. By allowing competition among subpopulations, these methods are generally able to improve the search efficiency to identify the optimal or near optimal solution.

Due to the semi-isolated nature of subpopulations, the overall population diversity can be maintained in the above methods. However, multiple subpopulations in these methods are all used to encode solutions of the same problem and each subpopulation can move around the entire search space of the problem. Consequently, to apply such methods, several key issues, for example how to create an appropriate number of subpopulations and how to maintain them in different subareas, have to be addressed. Since *a priori* knowledge of the problem's search space is usually not available, it is difficult to address these issues. Alternative multisubpopulation methods [68], [86], in which the above issues may not arise, have also been designed. In these methods, each subpopulation is used to encode one partial solution of the problem. The best partial solutions obtained at the end of evolution by subpopulations are assembled together to form a full solution. Nevertheless, in this approach, how to make subpopulations work cooperatively to deliver partial solutions, which can be used to form an optimal or near optimal full solution, is typically a difficult problem. In this paper, taking the advantage that multiple subpopulations can be naturally formed in the population and each of which can be used to search one disjoint subspace of the automatic clustering problem, an AMC mechanism, in which the issues mentioned above can be avoided, has been proposed. More importantly, the mechanism is developed in a manner to ensure a diverse search over subspaces while allowing the more promising subspaces to be more intensively searched.

Finally, we should mention that multisubpopulation methods certainly belong to the broader family of structured-population and parallel evolutionary algorithms. Good reviews of these algorithms and their applications can be found in [5], [6], and [55].

B. Niching Methods

While multisubpopulation methods improve the population diversity by evolving multiple spatially separated subpopulations, niching methods do so by fostering niches within the population. Many niching methods have been proposed, examples including fitness sharing [20], [35], clearing [67], [83], and crowding [16], [27], [40]. These methods can be broadly categorized into two major groups. The first group involves methods in which the neighborhood is explicitly determined by using a distance cutoff (i.e., sharing radius). Both sharing- and clearing-based methods belong to this group. The major shortcoming of these methods is that setting proper values of the sharing radius requires *a priori* knowledge about the location and spacing of the optima, which is generally not available [85].

The second group consists of methods, in which the neighborhood is implicit determined, requiring no distance cutoff. Crowding falls into this group. Crowding methods follow from the analogy that dissimilar individuals tend to locate in different niches, such that they usually do not compete. This can be realized by selecting similar parents for mating and/or promoting replacement of similar individuals in population by the newly generated offspring. Based on different selection and/or replacement strategies, various crowding methods have been proposed [24], [74]. Since crowding methods require no explicitly distance cutoff to induce the emergence of niches, the requirement of *a priori* knowledge about the search space can be relaxed, making them attractive to be applied for optimization problems. It is for this reason that the crowding method has been chosen here to study the automatic clustering problem. However, existing crowding methods usually introduce some other parameters, which must be appropriately specified in order to perform well. Clearly, setting proper values for these parameters by performing a trial-and-error search is a tedious task. Also, optimal values of these parameters may vary during evolution. The use of rigid parameter values identified by a trial-and-error search can still result in inferior performance. It would be therefore desirable to adaptively set these parameters depending on the problem instance in question and the phase of evolution. Although several studies [20], [82] have been carried out to deal with parameter issues arising in niching methods, they are designed for either fitness sharing- or clearing-based methods. Little attention has been paid to the parameter issue of crowding based niching methods, which have a rather different niching mechanism.

Niching methods can be used to promote the population diversity. At the same time, they permit the population to simultaneously locate multiple optima in the fitness landscape. This property makes them a popular choice to extend evolutionary algorithms for multimodal optimization, whose task is to identify multiple solutions. For instance, in [52], a niching method was developed and embedded in a GA to make it capable of detecting multiple solutions. In [92], a crowding-based differential evolution incorporating the locality principle was proposed for multimodal optimization and successfully applied to the problems of varied-line-spacing holographic grating design as well as protein structure prediction. Biswas *et al.* [11] presented an information-sharing mechanism to induce the niching behavior in differential evolution for multimodal optimization. While in [95], multimodal optimization is dealt with an ensemble of different niching methods in order to achieve a robust performance. Although these niching methods have been shown to be effective for multimodal optimization, they are generally designed to locate multiple optima, regardless of their goodness. This may limit their search efficiency when the objective is to locate a single optimal or near optimal solution.

Recently, NGAs or NMAs have been suggested for data clustering. For example, in [78], an NMA based on a variant of the standard crowding method [27] was designed for simultaneous feature selection and data clustering. In [21], a variant of the standard crowding method has also been incorporated into the GA, however, to cluster Web documents. Jaafar *et al.* [44] applied the restricted tournament selection method for niching in the GA to cluster railway-driving missions. By incorporating niching technique, the above methods have shown better performance compared with traditional GA- or MA-based clustering schemes. However, niching techniques employed in these methods are suffered from the difficulty of parameter setting and the computational inefficiency to identify the optimal or near optimal solution, as we mentioned above. In this paper, we design an adaptive niching scheme based on a crowding niching method for data clustering, such that its parameter values can dynamically change per clustering problem instance and during evolutionary process. Further, the scheme is designed to promote diverged niches of high fitness, therefore effectively and efficiently searching the solution space.

III. Proposed Algorithm

This section presents the proposed algorithm to effectively and efficiently search the solution space of clustering problem, thus delivering high quality solutions. The proposed algorithm starts with a number of randomly generated subpopulations, which are used to search different subspaces of the automatic clustering problem associated with different numbers of clusters. These subpopulations are evolved by MAs, independent each other, at each generation. After that, an AMC mechanism is developed and employed to control the migration between subpopulations and permit the subpopulations with lower performances to search their corresponding subspaces more aggressively. The mechanism is designed to ensure a diverse search over the solution space while allowing more promising subspaces to be more intensively searched. Further, an adaptive niching scheme is incorporated into the MAs running on subpopulations, whose performances are better than the mean performance of all subpopulations. The adaptive niching scheme is devised to encourage the forming of diverged niches of high fitness within these high performing subpopulations, thus allowing a diverse and efficient search of their corresponding subspaces. The proposed algorithm is implemented to optimize a fitness function consisting of three well-known clustering criteria. The algorithm will be terminated if the fitness of the best solution during evolution does not change for g generations. The output of the algorithm is the best solution found at the end of evolution. The overall evolutionary procedure is illustrated in Algorithm 1.

Algorithm 1 MA Based on AMC and Adaptive Niching for Automatic Data Clustering

```
Step 1. Generate a number of initial subpopulations such that
        individuals in different subpopulations are encoded with
        different numbers of clusters (see Section III.A).
Step 1. Compute the fitness for each individual in the initial popu-
        lation according to Equation (2) (see Section III.C) and then
        calculate the performance index of each initial subpopula-
        tion using Equation (5) (see Section IV.A).
Step 3. Repeat the following process until the termination condition
        is met.
          a) For each subpopulation, if its performance is better
             than the mean performance of subpopulations, then:
               i. Select $t/2$ ($t$ denotes the number of individuals in
                  the subpopulation) parent pairs using the AVMNC
                  selection operation (see Section V).
              ii. Generate offspring by performing crossover on
                  each paired parents and then apply mutation to the
                  offspring (see Section III.B).
             iii. Run a single iteration of $k$-means algorithm (see
                  Section III.B) to refine cluster centers encoded in
                  the offspring.
              iv. Apply the AVMNC replacement operation (see
                  Section V) to select an opponent from the sub-
                  population for each of the offspring.
               v. Compute the fitness for each offspring. If its fit-
                  ness is better than that of the paired opponent, then
                  the opponent in subpopulation is replaced by the
                  offspring.
              vi. Update the performance index of the subpopula-
                  tion.
             Otherwise:
             A similar procedure as above is performed on the
             subpopulation. However, rather than employing the
             AVMNC scheme, during selection, parent pairs are
             selected using the $k$-fold ($k$=2) tournament selec-
             tion method [36] and during replacement, the worst
             individuals are replaced by the offspring.
          b) Implement the adaptive multi-subpopulation competi-
             tion mechanism (see Section IV).
               i. Sort subpopulations according to their perfor-
                  mances and organize them using a chain topology.
              ii. Perform the migration operation between each
                  pair of neighboring subpopulations, in which the
                  amount of individuals to be migrated is adaptively
                  determined (see Section IV.A). Before migration,
                  the number of clusters encoded in each individ-
                  ual is adjusted using the cluster decrement or
                  increment operator (see Section IV.C).
             iii. Dynamically set the crossover and mutation rates
                  for different subpopulations according to their
                  performance rankings (see Section IV.B).
Step 4. Output the solution with the best fitness in terminal popu-
        lation.
```

In the following sections, we first describe how the initial subpopulations are created, how the recombination and local search operations work, and how the fitness function is constructed in the proposed algorithm. The details of the proposed AMC mechanism and adaptive niching scheme are then presented in Sections IV and V, respectively.

A. Solution Representation and Subpopulation Initialization

For optimization problems in continuous domain, real value-based representation is generally preferred to encode solutions in evolutionary algorithms, as it offers higher precision than other alternative representations [24]. Here, we employ the real value-based representation to encode cluster centers of the clustering solution. Based on this representation, each individual solution in each subpopulation is initialized utilizing a vector of $d \times k$ real numbers, where d is the dimension of data to be clustered, and k denotes the cluster number encoded in the solution. Each set of d values in the vector starting from the left represents one center of clustering solution. Its initial values are assigned randomly within the feasible range of features, determined from the data, to which they are assigned. The k is fixed for each subpopulation to generate solutions with the same number of clusters for that subpopulation. By varying the cluster number k, multisubpopulations can then be created. Here the value of k is set to vary from 2 to $k^{\max}$, where $k^{\max}$ denotes the maximum possible cluster number existing in the data set and is taken to be $\sqrt{n}$ (n is the number of data objects in the data set), which has been widely used and demonstrated to be suitable in clustering literature [48]. The suitability of using such a $k^{\max}$ value can be further confirmed by the fact that the numbers of clusters in almost all existing data sets, whose structures are known beforehand, are less than their corresponding $k^{\max}$ values. Certainly, the value of $k^{\max}$ can be increased to as large as n, if necessary. However, this will result in a larger search space. By employing such a $k^{\max}$ value, the number of subpopulations existing in population will therefore equal to $k^{\max} - 1$, covering the cluster numbers ranging from 2 to $k^{\max}$.

B. Recombination and Local Search Operations

During evolution, each paired parents in each subpopulation will be subject to recombination operation (i.e., crossover and mutation). Considering the solutions are encoded with cluster centers, it is desirable to preserve these cluster centers during crossover operation. For this purpose, we restrict crossover positions to lie in between two cluster centers by employing an operator analogous to the traditional two-point crossover operation [36], which is implemented as follows. Given paired parents p_1 and p_2 encoded with k cluster centers, crossover points x_1 and x_2 are generated using the function RandInt$(1, k)$, which returns a natural number between 1 and k. If x_1 is greater than x_2, then their values are swapped to make sure $x_2 > x_1$. After that, data segments between x_1 and x_2 in p_1 and p_2 are exchanged. This crossover operation will be performed on each set of paired parents of each subpopulation with a probability, which is dynamically determined based on the subpopulation's performance rank (see Section IV-B). The generated offspring are subsequently subject to mutation. We perform the Gaussian mutation [36] on each feature value of the offspring with a probability, which is also determined dynamically (see Section IV-B). The Gaussian mutation is defined as

$$v_i^* = v_i + \alpha * N(0, 1) * v_i \tag{1}$$

where v_i and v_i* are feature values before and after, respectively, the mutation operation, $N(0, 1)$ denotes a random Gaussian number with a mean of zero and a standard deviation of one, and $\alpha \in [0.1, 0.2]$ is a randomly generated step size. If the new value falls outside the boundaries of that feature, it is modified to fall on the violated boundary. For instance, suppose the range of a feature is determined to be [0.1, 0.8]. If the mutation operation results in a value greater than the feature's upper boundary, then the value is modified to equal its upper boundary value (i.e., 0.8).

To improve the time efficiency, the *k*-means algorithm is finally employed to refine the clustering solution encoded in the offspring, creating a so-called MA [18] for data clustering. Given an initial set of *k* cluster centers, the *k*-means algorithm attempts to minimize the sum of squared error (SSE) clustering criterion by iterating between two steps: 1) assignment and 2) update. During assignment step, each data object is assigned to a cluster whose center is closest to it. During update step, the center of each cluster is updated by the mean of data points assigned to it. The process of the algorithm converges when no further reassignment improves the SSE. Rather than applying *k*-means on the offspring until convergence, here a single iteration of it will be used to refine cluster centers encoded in the offspring. This is done by assigning each data object to the nearest cluster center encoded in the offspring. After that, each cluster center is updated as the mean of data objects assigned to it. Here, the Euclidean metric is used to measure the distance between data objects and cluster centers.

C. Fitness Function

The fitness function is used to calculate the solution's fitness, which indicates the goodness of the resulting clustering it represents. The widely used clustering criterion, SSE, can be employed for this purpose. However, this criterion requires the number of clusters existing in the data set to be known beforehand. Since such knowledge is usually not available, many other criteria (commonly known as validity functions) have also been proposed, which can be optimized to simultaneously determine the number of clusters as well as partitioning of the data set. Unfortunately, these validity functions are generally sensitive to structures of clustering problems. Different validity functions may deliver rather different solutions on the same data set and it is not clear which one is the best. To deal with this dilemma, employing multiple validity functions to conduct some sort of voting about the best solution has been suggested in clustering literature [80]. A related motivation of using multiple validity functions is to improve the reliability of clustering results.

In this paper, we follow the above idea and construct a fitness function consisting of three validity functions, namely, the Davies–Bouldin (DB) [25], Calinski and Harabasz [13], and *I*-index [60]. The constructed fitness function takes the form of

$$f(x) = \sum_{i=1}^{3} w_i f_i(x) \tag{2}$$

where $f_i(x)$ is the value computed from the *i*th component validity function and w_i is a weighting coefficient denoting the relative significance of that function. Since the relative significance of these three functions for data sets to be clustered is generally unknown beforehand, we consider them to be equally important by setting $w_1 = w_2 = w_3 = 1/3$. The three validity functions employed in our fitness function have different rationales and properties to measure clustering solutions while being computationally efficient. The resulting fitness function is therefore well suitable to serve for our purpose. However, employing this function for fitness computation is not so straightforward. This is due to different functions may return rather different range of possible values and there is no *a priori* information available for normalizing these values. Here, we seek to normalize them dynamically during evolution. Consequently, the fitness of solutions will be also calculated dynamically when needed. Specifically, for each new solution, the value of each component function is calculated and stored in a vector, associated with the solution. Upon calculating the fitness of the solution, stored values in its associated vector are first extracted and each of them is subsequently normalized using

$$f_i(x) = \Big(f_i(x) - f_i^{\min}(x)\Big) \Big/ \Big(f_i^{\max}(x) - f_i^{\min}(x)\Big) \tag{3}$$

where $f_i^{\min}(x)$ and $f_i^{\max}(x)$ are the minimum and maximum values, respectively, of the *i*th component function found so far during evolution. The normalized values are finally used to compute the fitness of the solution according to (2).

IV. Adaptive Multisubpopulation Competition Mechanism

As with traditional GAs, MAs are known to suffer from premature convergence when they are applied to solve complex optimization problems. This behavior is mainly caused by the rapid loss of population diversity. To cope with such a drawback, preserving the population diversity during evolution to ensure a diverse search of the solution space is vital. Specifically, to approach the automatic clustering problem with an MA, one should maintain the diversity with respect to cluster numbers existing in the population as well as the diversity among solutions with the same number of clusters. As the search space of automatic clustering problem can be decomposed into disjoint subspaces corresponding to different numbers of clusters, multiple subpopulations can be naturally created in population, each of which is used to search one particular subspace. Inspired by the diversity-preserving property of using multisubpopulations, in this section, we present an AMC mechanism, which can be used to achieve a diverse search over solution subspaces associated with different numbers of clusters and, at the same time, an intense search of the promising subspaces.

The proposed mechanism organizes multiple subpopulations in a chain topology, in which each subpopulation can send or receive individuals from the next and previous subpopulation in the chain. These subpopulations are first undergoing independent evolution to search their corresponding subspaces at each generation. After that, the performance of each subpopulation is measured and it then competes with its neighbors in the chain for computational resources by migrating individuals from lower performing subpopulations to higher performing ones. This process is repeated until only one single subpopulation remains in the population or the termination condition is met. Unlike in most implementations of the migration operation [8], [55], which exchange individuals between subpopulations while the number as well as size of subpopulations are being kept unchanged during evolution. The individuals to be migrated here will leave their subpopulations and be inserted into receiving subpopulations. As a result, both number and size of subpopulations will vary during evolution.

Obviously, to effectively achieve a diverse search over the subspaces associated with different numbers of clusters as well as an intensive search of the promising subspaces with the above procedure, a few issues have to be properly addressed. First, the amount of individuals, δ, to be migrated between each pair of neighboring subpopulations should be appropriately controlled. For this purpose, we propose an adaptation strategy to dynamically adjust the value of δ based on the performance of subpopulations as well as the diversity of cluster numbers in population. Second, subpopulations in the chain should be adequately arranged to prevent the subpopulations with low performances from disappearing too quickly before their corresponding subspaces are being sufficiently searched. This is accomplished by sorting subpopulations in the chain according to their performances, therefore restricting the migration to occur only between subpopulations with relatively similar performances. Consequently, a diverse search over different subspaces at an early stage and then smoothly shifting to an intensive search of promising subspaces at a later stage of evolution can be achieved. Third, it is important to allow subpopulations with low performances to search their corresponding subspaces more aggressively. We have therefore dynamically assigned different crossover and mutation rates to different subpopulations—a subpopulation with a lower performance rank will be assigned with higher mutation and crossover rates to increase its explorative power. Fourth, since individuals from different subpopulations have different numbers of clusters, the number of clusters encoded in the individuals subjected to migration should be appropriately adjusted to be consistent with the ones in receiving subpopulation. Two operators that can serve for this purpose have also been designed. In the following sections, we shall first describe the details of proposed adaptive migration strategy. Then, the strategy for setting different crossover and mutation rates for different subpopulations is provided. Finally, we describe the operators devised to facilitate the migration between subpopulations.

A. Adaptive Migration Strategy

In the proposed AMC mechanism, different subspaces are searched using different subpopulations. By employing one-way migration between subpopulations, more promising subspaces can be allocated with more search efforts. The distribution of search efforts among the subspaces, therefore, can be controlled by adjusting the amount of individuals, δ, to be migrated, which affects the performance of proposed mechanism. If the δ is too large, subpopulations with low performances can disappear quickly before their corresponding subspaces are being sufficiently searched. If the δ is too small, search efforts will be largely wasted on less promising subspaces, yielding an inefficient search. A relevant question is, therefore, what value of δ should be used to balance between the diverse search over different subspaces and intensive search of promising subspaces. The best value of δ, however, differs from problem to problem and varies as subpopulations evolve within a single problem. Here, we proposed to adaptively adjust the δ between each pair of neighboring subpopulations based on their performances and the diversity of cluster numbers in population during evolution.

The performance of a subpopulation depends on the subspace being searched. For the maximization problem, a subpopulation associated with a promising subspace will generally have a high fitness of the best individual and/or high growth rate of the best individual. Accordingly, the performance of a subpopulation at each generation can be measured in terms of the current fitness of the best individual and its growth rate. Let $f_{i,t}$ and $f_{i,t'}$ be the fitness of the best individual before and after the independent evolution, respectively, of the ith subpopulation at generation t, its growth rate can be computed as

$$G_{i,t} = \left(f_{i,t'} - f_{i,t}\right) \Big/ \left(f^{\max} - f^{\min}\right) \tag{4}$$

where $f^{\max}$ and $f^{\min}$ are the maximum and minimum fitness, respectively, in the population recorded so far during evolution and are used for the normalization purpose. Having obtained the growth rate, performance index of the subpopulation at generation t can then be defined as

$$PI_{i,t} = f_{i,t'} * (G_{i,t} + 1). \tag{5}$$

A high performing subpopulation will thus be the one with a high fitness of the best individual as well as a high growth rate.

Based on the calculated performance index value for each subpopulation, we subsequently set the amount of individuals to be migrated between each pair of neighboring subpopulations at generation t using

$$\delta = \begin{cases} (PI_{i,t} - PI_{i+1,t}) * n_{i+1} * \left(PI_t^{\max} - PI_{i+1,t}\right)/PI_t^{\max} \\ \quad * k_t/(k^{\max} - 1), \ \text{if } PI_{i,t} \geq PI_{i+1,t} \\ (PI_{i+1,t} - PI_{i,t}) * n_i * \left(PI_t^{\max} - PI_{i,t}\right)/PI_t^{\max} \\ \quad * k_t/(k^{\max} - 1), \ \text{otherwise} \end{cases} \tag{6}$$

where n_i denotes the size of ith subpopulation. The equation shows that the amount of individuals to be migrated from a lower performing subpopulation to a higher performing one is proportional to the difference of their performances and the size of the lower performing subpopulation. Further, our strategy for adapting the value of δ is to adjust it to a small value, if the lower performing subpopulation has a relatively high value of performance index, thus allowing its corresponding subspace to be more sufficiently searched. For this purpose, we add a term $(PI_t^{\max} - PI_{i,t})/PI_t^{\max}$ into the equation, where $PI_t^{\max}$ is the largest performance index value obtained among the subpopulations at generation t. Additionally, the value of

δ has been devised to reflect the diversity of cluster numbers in population, by adding another term, $k_t/(k^{\max} - 1)$, into the equation, where k_t is the total number of different numbers of clusters existing in the population at generation t. The basic idea behind this term is that the value of δ is set to gradually reduce along with the decreasing of diversity of cluster numbers in population (i.e., the disappearance of low performing subpopulations), thus putting a more emphasis on exploration of the subspaces associated with high performing subpopulations.

The above adaptation strategy can play its role well to determine the amount of individuals to be migrated between subpopulations. However, the performance of proposed mechanism is also affected by how subpopulations in the chain are arranged. By arranging them in a random order or with respect to their associated numbers of clusters, the subpopulations will quickly disappear if their neighbors have significantly better performances. To address such an issue, we further restrict the adaptive migration operation to occur only between subpopulations with relatively similar performances. This is achieved by sorting subpopulations in the chain according to their performances at each generation. By doing so, we aim to maintain the smoothness of migrating individuals from low performing subpopulations to high performing ones. As a result, the shrinking speed of low performing subpopulations can be appreciably reduced, thus allowing their corresponding subspaces to be sufficiently searched.

Having sorted subpopulations in the chain, the proposed adaptive migration is finally performed on each pair of neighboring subpopulations at each generation. The sequence of application of the migration operation is according to the decreasing order of subpopulation's performance. Before migrating, (6) is first evaluated to determine the value of δ. Then, a random real number r between δf and δc is generated, where δf and δc denote the floor and ceiling of δ. If r is smaller than δ, δf individual(s) will be migrated from the lower performing subpopulation to the higher performing one, otherwise migration of δc individual(s) will occur. To prevent the search capability of losing subpopulation to be deteriorated further, the worst individuals in terms of fitness are selected for migration. In case that the size of losing subpopulation is less than or equal to the amount of individuals to be migrated, then all remaining individuals will be migrated and the empty subpopulation is removed from the chain.

The outline of the adaptive migration procedure is given in Algorithm 2.

```
Algorithm 2 Outline of Adaptive Migration Procedure
Step 1. Calculate the performance of each subpopulation
        according to Equation (5).
Step 2. Organize the subpopulations using a chain topology
        and sort them in descending order according to their
        performances.
Step 3. For each pair of neighboring subpopulations in the
        chain:
        a) Determine the amount of individuals to be migrated
        by evaluating Equation (6).
        b) Perform one-way migration (from the lower performing
        subpopulation to higher performing one)
        by selecting individuals from the lower performing
        subpopulation and adjusting the number of clusters
        encoded in them accordingly via the proposed
        conversion operators.
```

B. Dynamic Mutation and Crossover Rate

The above adaptive migration can be used to control the shrinking speed of low performing subpopulations to maintain appropriately number of individuals to search their subspaces during evolution. To effectively achieve a diverse search over the subspaces, it is also important to allow low performing subpopulations to search their corresponding subspaces more aggressively. With this in mind, we develop a strategy to dynamically adjust the exploration power of MAs running on the subpopulations according to their performance ranking at each generation. The exploration capability of an MA is mainly determined by the rates of recombination operation, i.e., mutation and crossover [36], [76]. Higher rates of mutation/crossover operator will lead to a more explorative search. In our strategy, the MA running on the subpopulation with a lower performance rank will be assigned with higher mutation and crossover rates. Specifically, the following equations define the proposed strategy for setting the mutation and crossover rates ($p_{m,i,t}$ and $p_{c,i,t}$, respectively) of the MA running on the ith subpopulation at generation t:

$$p_{m,i,t} = p_m^{\min} + \left(p_m^{\max} - p_m^{\min}\right) * (\mathrm{Rank}(i,t) - 1)/(s-1) \quad (7)$$

$$p_{c,i,t} = p_c^{\min} + \left(p_c^{\max} - p_c^{\min}\right) * (\mathrm{Rank}(i,t) - 1)/(s-1) \quad (8)$$

where s is the total number of subpopulations existing in population, and $\mathrm{Rank}(i,t)$ is the rank of ith subpopulation at generation t in the sorted subpopulations ranging from 1 to s. Here, $p_m^{\min}$ and $p_c^{\min}$ are taken to be $p_m^{\min} = 0.001$ and $p_c^{\min} = 0.6$, which are referred to as "standard" setting for the mutation and crossover rates [27]. The $p_c^{\max}$ is set to be 1.0, which is the highest possible rate. For the mutation, since a rate larger than 0.1 essentially corresponds to random search [37], $p_m^{\max} = 0.1$ is therefore used here.

From the above rule, we can see that the MA running on the top-ranked subpopulation (with the best performance) at each generation will be assigned with standard crossover and mutation rates. For the rest subpopulations, the lower a subpopulation's rank is, the higher mutation and crossover rates of its associated MA have, resulting in a more explorative search. The multisubpopulation search based on such a rule can therefore be viewed as the one, which exploits promising subspaces while exploring other subspaces. The search is also competitive, as an increase of the rank of some subpopulations will result in other subpopulations with reduced ranks searching their corresponding subspaces more explorative.

C. Conversion Operators

In the proposed mechanism, individuals from different subpopulations are encoded with different numbers of clusters.

Before migration, the number of clusters encoded in the individuals should be appropriately adjusted to be consistent with the ones in receiving subpopulation. Here, two conversion operators, cluster decrement and cluster increment, have been devised to facilitate the migration. The cluster decrement operation is devised to decrease the number of clusters encoded in the solution. This operator works by choosing a cluster (represented by its center encoded in the solution), which, while merging with its nearest cluster, yields the best fitness of the solution, and updating corresponding cluster centers with the new center. The new center is calculated as the center of data objects belonging to the two clusters to be merged. The cluster increment operator, on the other hand, is devised to increase the number of clusters in the solution. In this operator, we select a cluster, which, while splitting into two clusters, yields the best fitness of the solution, and replace its center with the new centers of two separated clusters. Supposing the cluster to be split to be centered at c, the new centers c_1 and c_2 will be formed by adding and subtracting a certain value of v_a and v_b, respectively, to and from the feature of c which shows the maximum standard deviation, denoted as $\lambda_{\max}$, among all features of the cluster c. A simple way to specify the value of v_a and v_b is to make it equal to some fraction of $\lambda_{\max}$. Here, we set $v_a = v_b = \lambda_{\max}/2$.

In case of migrating an individual from a subpopulation with a larger number of clusters to a subpopulation with a smaller number of clusters, the cluster decrement operator will be executed recursively until the number of clusters encoded in the individual is consistent with the destination subpopulation. Otherwise, the cluster increment operator will be applied. These operators can preserve the clustering information in the individuals to be migrated, especially if the migration is happened between subpopulations with similar numbers of clusters. Further, they work in a local improvement manner to refine the number of clusters in individuals. Thus, the resulting individuals usually have fitness values good enough to survive in the destination subpopulation.

V. Adaptive Multiniche Crowding Scheme

The above AMC mechanism can be used to promote a diverse search over subspaces corresponding to different numbers of clusters. However, such a mechanism may not necessarily result in a diverse search within the subspace, which is also critically important. Although existing niching methods can be further employed for this purpose, they usually involve certain parameters, which have to be properly specified in order to perform well. In this section, we develop an adaptive niching scheme based on a VMNC method [16] such that its parameter values can be dynamically adjusted per clustering problem instance and during evolutionary process, resulting a scheme denoted as AVMNC. The adaptation is achieved by utilizing a diversity index, which is introduced to measure the fitness-weighted genotypic diversity of subpopulation. Since the index can be employed to promote both genotypic diversity and fitness, diverged niches of high fitness can thus be formed and maintained in the subpopulation during evolution, making it capable of effectively and efficiently searching the subspace. The developed adaptive niching scheme will be incorporated into MAs running on the subpopulations, whose performances are better than the mean performance of subpopulations. In the following, we first briefly describe the VMNC in Section V-A. Subsequently, a population diversity index is introduced in Section V-B. Employing this index, we finally present an adaptation policy to adaptively set parameter values of the VMNC in Section V-C.

A. VMNC

The VMNC method [16] attempts to induce niches in the population of GA by encouraging mating and replacement between similar individuals during evolution. In the VMNC, during selection, one individual, p_1, is selected randomly from the population and its mate is coming from a group of individuals with the size of s_m, picked randomly from the population. The one, which is the most similar to p_1 in the group, s_m, is chosen as its mate p_2. Note, here the similarity between individuals is measured using the genotype distance. To avoid the issue due to different orderings of cluster centers in individuals, they are reordered such that the closest cluster centers appear at the same position before calculating the similarity. If this results in a group with more than one individual for selection, then an individual is selected at random from the group. The pair is then recombined to produce offspring. After that, each generated offspring is paired with the most similar individual from another group of individuals with the size of s_r, which is also randomly selected from the population. The offspring subsequently competes with its paired opponent to survive based on the fitness. If the offspring has better fitness, then the opponent in population is replaced by the offspring. As some type of crowding is used in both selection and replacement operation, the VMNC can be viewed as a generalized version of crowding-based niching methods. Like most of existing niching methods, the performance of VMNC depends on its parameter values [54], specifically the s_m and s_r in this case. Too large a value of s_r (i.e., considerably larger than the number of high-fitness peaks existing in the search space) will lead to too many niches being preserved in the population and correspondingly decrease the search efficiency to locate the optimal or near optimal solution, while too small a value can result in insufficient population diversity, limiting its capability to effectively search the solution space. Meanwhile, a large value of s_m (i.e., larger than the number of niches existing in the population) will restrict the search within niches whereas a small value tends to encourage global search. Regardless of their importance, no consistent methodology is available to set these parameters and they are typically arbitrarily set within some specific ranges while remaining fixed during entire run of the algorithm. Since appropriate values of these parameters depend on the problem instance and may vary during evolution, this apparently can lead to inferior performance.

B. Fitness-Weighted Genotypic Diversity Index

To achieve a satisfactory performance with the above niching technique, here we first introduce a population diversity index, which will be employed to adaptively adjust its parameter values. A number of indexes [1], [22], [75] have been devised to measure the diversity of population. They are typically based on either phenotypic or genotypic variance of population. Phenotypic diversity indexes are defined by the fitness distribution of population. These indexes may not able to reflect the genotypic diversity of population, as in complex fitness landscapes, many different solutions can have similar fitness values. Genotypic diversity indexes, by contrast, are built from the spread of individuals over the search space. They do not take into account the fitness of individuals. Consequently, a population exhibiting high diversity in terms of genotypic diversity indexes, its individuals may not necessarily lie in highly promising areas across the solution space. To overcome these shortcomings, here we introduce the fitness of individuals into a genotypic diversity index to measure the diversity of a population. Since the proposed adaptive niching scheme will be applied on subpopulations in this paper, the diversity measure is also defined on the subpopulations.

Specifically, for a subpopulation with t individuals, let k denote the cluster number encoded in the individuals and d be the dimension of data. The diversity of subpopulation is computed as follows. We first calculate the mean spatial position of the subpopulation as

$$m_j = \frac{1}{t}\sum_{i=1}^{t} p_{i,j} \tag{9}$$

where $p_{i,j}$ is the jth gene of individual p_i, $p_i = (p_{i,1}, p_{i,2}, \ldots, p_{i,k\times d})$. Then, the diversity contribution of each individual is evaluated based on a fitness-weighted distance from the individual to the mean spatial position as

$$D_i = w_i \times \sqrt{\sum_{j=1}^{k\times d}\left(p_{i,j} - m_j\right)^2}. \tag{10}$$

Here, $w_i = f_i/f_{\text{ave}}, f_i$ and f_{ave} denote the individual's fitness and mean fitness of the subpopulation, respectively. By summing the diversity contribution of all individuals, we can finally define the diversity of the subpopulation as

$$D = \sum_{i=1}^{t} D_i. \tag{11}$$

The above population diversity index is defined by weighting each individual's genotypic diversity contribution by its fitness score. Accordingly, individuals spreading across the space will have a high diversity if these individuals are highly fit. The index is therefore able to used to promote both genotypic diversity and high fitness of the solutions.

C. Adaptation Policy

Based on the above diversity index, we finally dynamically adjust parameter values of VMNC for data clustering. The key idea is to form and preserve diverged niches with high fitness in the subpopulation during evolution, thus effectively and efficiently search the subspace.

The adaptation policy is implemented as follows. At the initial stage of evolution, a high value of s_r is used to promote the emergence of niches in subpopulation. This is further supported with a low value of s_m to encourage global search. During this stage, the diversity of subpopulation in terms of the devised index will generally increases gradually along with the niches of good fitness progressively emerge. This stage will continue until the diversity value ceases to increase within a specific number of generations and the maximum diversity value is recorded. After that, the values of s_r and s_m are dynamically determined according to the rules

$$\begin{cases} s_r = s_r^{\max} - D/D^{\max} \times \left(s_r^{\max} - s_r^{\min}\right) \\ s_m = 2 * s_r \end{cases} \tag{12}$$

where $s_r^{\max}$ and $s_r^{\min}$ denote the upper and lower bound values, respectively, of the parameter s_r, and $D^{\max}$ is the maximum diversity of the subpopulation recorded so far during evolution. Thus, the value of s_r will vary between $s_r^{\min}$ to $s_r^{\max}$ based on the diversity of subpopulation.

The basic principle of this adaptation policy is that, at the beginning of adaptation, where a large number of niches are typically existing in the subpopulation, small values of s_r and s_m are used to boost competition among niches. Then, along with the gradually disappearing of niches with relatively low fitness as well as decreasing of diversity of the subpopulation, the sizes of s_r and s_m will subsequently increase correspondingly. Consequently, diverged niches with high fitness can be formed and preserved, allowing the subspace of automatic clustering problem to be effectively and efficiently searched.

VI. Experiments

This section describes data sets used in experiments and parameter settings for the proposed algorithm. This is followed by a series of experiments, in which we first evaluate the impact of AMC mechanism in the proposed algorithm. Then, the significance of AVMNC scheme is examined. Finally, we assess the performance of the proposed algorithm by comparing it with other related work. All algorithms were implemented on a workstation with an Intel Core i7-3770K CPU at 3.50 GHz and 16 GB RAM running the Windows 7 operating system. Unless otherwise stated, results were obtained by generating ten random populations and executing each algorithm once on each of the population. By doing so, we aim to ensure that different results are not caused by different initial populations. All fitness values reported in this section are normalized using the minimum and maximum values of each component of the fitness function identified in experiments on each data set.

A. Data Sets and Parameter Settings

Both artificial and real data have been used for experiments. The artificial data sets, shown in Fig. 1(a)–(c), are generated with different properties for automatic clustering. The first one, DS_10, is relatively simple, in which one larger cluster surrounded by nine smaller clusters. In the second data set, DS_16, clusters have different volumes and sizes while some of them are overlapped. There are 16 clusters in this data set.

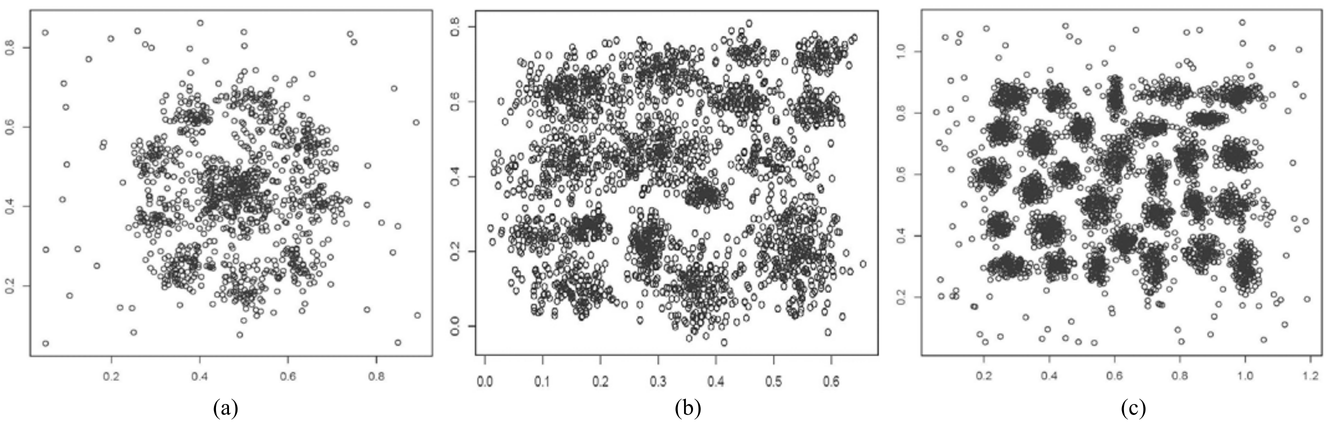


Fig. 1. Artificial data sets used in experiments: (a) DS_10, (b) DS_16, and (c) DS_30.

In the third data set, DS_30, 30 clusters are existed. In both DS_10 and DS_30, many outlier data points have been added to increase the difficulty of problem-solving task further.

A number of real data sets can also be used for our experiments. However, many of them involve a small data size with few clusters. To cluster these data sets, the search spaces usually contain a relatively small number of local optima and therefore can be generally addressed by employing a typical GA- or MA-based algorithm. Here, we mainly concern clustering large data sets, which have complex search spaces with a vast number of local optima, and thus have considered three gene-expression data sets (Subcell2945, Subcell384, and Subcancer) as well as two data sets (optical digit and image segmentation) from the University of California at Irvine (UCI) repository [63]. The Subcell2945 and Subcell384 data are subsets of the yeast cellcycle data given by Cho *et al.* [19]. This data set contains expression profiles (i.e., fluctuation of expression levels) of 6220 genes during the mitotic cell division cycle. Expression levels of the genes are sampled at a 10 min interval resulting in a total of 17 measurements (i.e., features), which covers nearly two yeast cell cycles. The Subcell2945 used in our experiments is picked by Tavazoie *et al.* [89], who selected 2945 most variable genes out of 6220. In this subset, the data at time points of 90 and 100 min have been excluded due to less efficient mRNA labeling during original chip hybridizations. The second subset Subcell384 has 384 genes, whose expression peaks at different time points corresponding to five phases of the cell cycle [19]. The clustering result therefore should reflect this to obtain a five-cluster partitioning. The Subcancer data is a subset of 60 human cancer cell line microarrays, representing 9703 spots corresponding to 6649 unique genes [72]. This subset contains 728 genes, of 60 features, and is selected by Dembele and Kastner [28]. For both Subcell2945 and Subcancer, the *a priori* knowledge about the correct number of clusters is not available. These three data sets are all clustered based on the genes' expression profiles. The optical digit is a data set on optical recognition of handwritten digits. This data set contains 5620 data objects, of 64 features, describing the images of ten handwritten digits (0–9). Hence, ten clusters are present in the data set. While the image segmentation data set consists of 2310 objects with 19 features from seven types of images (sky, brickface, foliage, window, cement, grass, and path). Correspondingly, there are seven clusters in this data set. To cluster the above real data sets (except Subcell384) by optimizing a certain validity function, it could be very difficult to identify the optimal or near optimal solutions due to complex search spaces with a vast number of local optima are involved. In the experiments, artificial data sets and the two UCI data sets are normalized for each feature to have a mean of zero and a standard deviation of one while gene expression data sets are normalized such that each gene has a mean of zero expression value and a standard deviation of one.

To implement the proposed algorithm, although values of several key parameters will be dynamically determined during evolution (as described in Sections IV and V), there are still other parameters whose values need to be specified. These include the $s_r^{\min}$ and $s_r^{\max}$ used for adaptively setting the value of s_r and s_m in AVMNC, the size of subpopulation as well as the value of g used for terminating the evolution. Here, we experimentally determine the values of $s_r^{\min}$ and $s_r^{\max}$ based on the aforementioned data sets. Specifically, for each parameter, five trials of the algorithm are executed for a wide range of values in each case while all other parameter values are held constant. Then, the mean fitness of solutions resulting from the five trials is calculated for each case. The one, that gives the best mean fitness, is subsequently selected as the value of the parameter. As a result of the experiments, the values of $s_r^{\min}$ and $s_r^{\max}$ are set equal to one-tenth and half of the subpopulation size, respectively. These values can yield the best results among a wide range of values of each parameter to be tested. For the size of subpopulation and the value of g, we set them as 20 and 30, respectively. Larger values of either parameter will generally lead to a longer execution time with no significant improvement of clustering solutions. A summary of the values of these parameters as well as other required parameters as specified in the previous sections are shown in Table I.

B. Experimental Results

We first evaluate the impact of AMC mechanism in the proposed algorithm. To this end, we build an MA for automatic data clustering (denoted as MAClustering), which is used as a basis for comparison. In this MA, neither the AMC nor

TABLE I
SUMMARY OF THE PARAMETER VALUES USED IN THE PROPOSED ALGORITHM

Parameter	Value
Size of subpopulation	20
s_r^{min}	2
s_r^{max}	10
g	30
p_m^{min}	0.001
p_m^{max}	0.1
p_c^{min}	0.6
p_c^{max}	1.0
k^{max}	$\sqrt{n}$ (n is the number of data objects in the data set)
α in Gaussian mutation	[0.1,0.2]

AVMNC is employed. All the subpopulations are combined to form one single population. During selection, parent pairs are selected using the k-fold ($k = 2$) tournament selection method, and during replacement, the worst individuals in population are replaced by the offspring. All other operators are the same as the ones used in our proposed algorithm. Then, an algorithm (denoted as AMC_MAClustering) is constructed by incorporating the proposed AMC mechanism into the MAClustering. Further, to assess the operations of performance-based subpopulation sorting as well as dynamic mutation and crossover rate settings for subpopulations in AMC, two variants of AMC_MAClustering have also been built. In the first variant (denoted as AMC_MAClustering1), the AMC mechanism is employed, however, without the performance-based subpopulation sorting operation. Rather, subpopulations in this variant are organized in a descending order according to the number of clusters encoded in them. The second variant (denoted as AMC_MAClustering2) is the AMC_MAClustering without the dynamic mutation and crossover rate settings for subpopulations in AMC. In this variant, mutation and crossover rates for all subpopulations are set with the same value, i.e., mutation rate of 0.001 and crossover rate of 0.6, which are referred to as standard setting for these operators [27]. These algorithms are implemented to cluster the experimental data sets and their performances are compared.

Fig. 2(a)–(d) shows the mean fitness of the best solutions found from ten trials over the runtime of four algorithms on four data sets. It can be observed that MAClustering gives the worst performance among four algorithms to be compared. By incorporating the AMC mechanism, both AMC_MAClustering1 and AMC_MAClustering2 can improve its performance while AMC_MAClustering achieves the best performance. Particularly, the results show that the improvements are generally more significant for data sets with larger and more complex search spaces. By looking closely into evolution processes of the four algorithms, we can find that, for MAClustering, its population diversity with respect to the cluster numbers reduces quickly during evolution. Consequently, the population could be easily trapped into a subspace, which contains local optimal solutions, thus resulting in the worst performance among the four algorithms. The AMC mechanism can restrain the loss of population diversity with respect to the cluster numbers during evolution. However, for AMC_MAClustering1, evolution progresses reveal that the subpopulations with low performances at the early stage of evolution quickly disappear when their neighbors have significantly better performances. This, in turn, negatively affects its performance since some subspaces associated with these quickly disappeared subpopulations may contain high quality solutions. Similarly, for AMC_MAClustering2, subpopulations with low performances may also disappear before their corresponding subspaces are being sufficiently searched, leading to the same drawback as for AMC_MAClustering1. For AMC_MAClustering, the evolutionary process shows that the subspaces containing high quality solutions are gradually identified during early stage of evolution and then intensive searched at a later stage of evolution, thus being capable of effectively searching the space. This can help to explain why AMC_MAClustering has the best performance and to justify the significance of the proposed AMC mechanism.

Next, experiments are carried out to examine the significance of AVMNC in the proposed algorithm. We hence assessed and compared the proposed algorithm, denoted as adaptive multi-subpopulation competition and multi-niche crowding-based memetic algorithm (AMMMA) to AMMMA incorporating no niching method (denoted as AMMMA1), AMMMA incorporating the VMNC with parameters s_r and s_m being set as a quarter and half, respectively, of the subpopulation size (AMMMA2), and AMMMA incorporating the VMNC with experimentally determined optimal parameter values (AMMMA3). In the case of AMMMA1, parent pairs are selected using the k-fold ($k = 2$) tournament selection method, and the worst solutions in the subpopulations are replaced by the generated offspring during replacement.

Fig. 3(a)–(d) shows the results in terms of mean fitness of the best solutions during evolution corresponding to the four algorithms on four data sets. The results reveal, the VMNC in AMMMA2 helps to prevent it from premature convergence to less promising optima, which can happen in AMMMA1. While AMMMA3 shows that, compared with AMMMA2, it can achieve comparable solutions, but more efficiently, on the six data sets. The better performance of AMMMA3, however, comes at the price of a tedious and time-consuming procedure of parameter tuning for each data set. Incorporated with the proposed AVMNC scheme, AMMMA can deliver even better or comparable results than AMMMA3 and at the same time requiring no manually intensive parameter setting. The relatively poor performance of AMMMA3 is primarily due to there is no mechanism, which can be used to form and maintain diverged niches of high fitness in the subpopulation as well as the usage of fixed niching parameter values, which does not reflect the intrinsically dynamic process of evolution. By examining subpopulations with the best performance, we can find that many of niches maintained at the middle stage of evolution for AMMMA3 are actually not highly fit. Further, the number of niches tends to gradually decrease during the late stage of evolution, with few left at the end of evolution. On the other hand, in the case of AMMMA, most of the formed niches are highly fit while many of them can be preserved until the end of evolution. In fact, this is the main reason that AMMMA can outperform AMMMA3, highlighting the merit of the proposed AVMNC scheme.

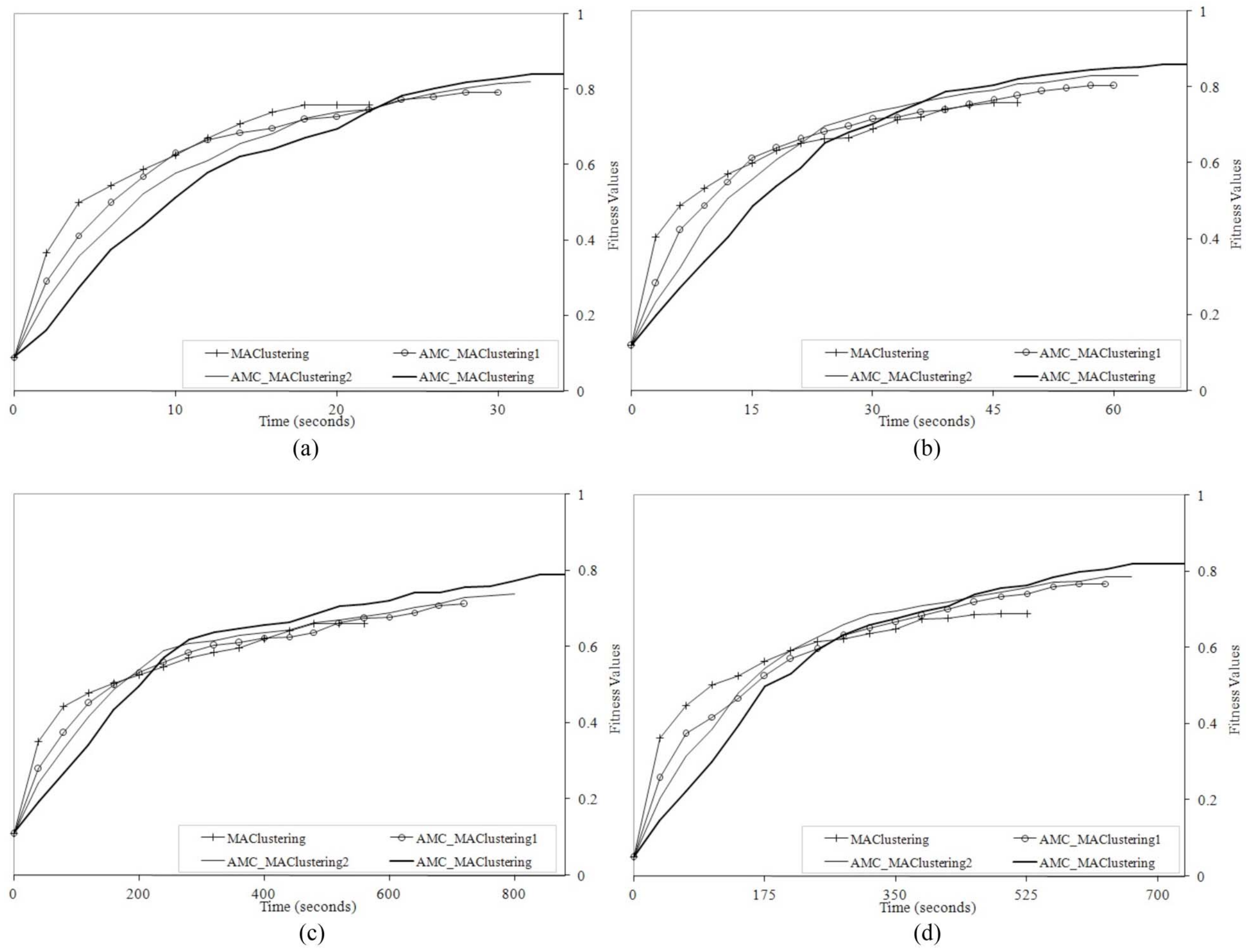
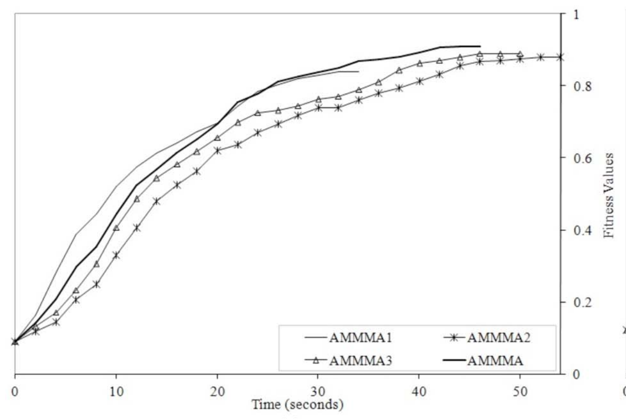
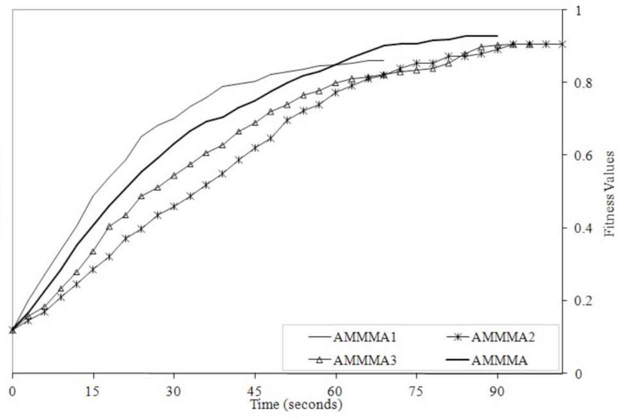


Fig. 2. Mean fitness value of the best solutions over the runtime of the four algorithms to be compared on (a) DS_16, (b) DS_30, (c) Subcell2945, and (d) Subcancer data set.

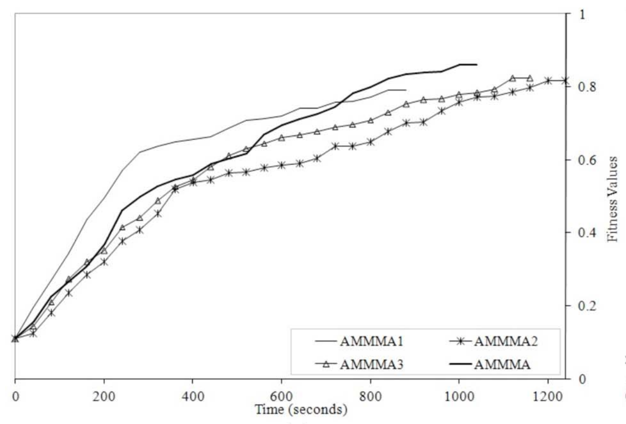
Then, the performance of the proposed algorithm is assessed by comparing it with related work, including the variable string length genetic algorithm (VGA) [60], web document clustering based on a new niching memetic algorithm (WDC-NMA) [21], two-stage genetic clustering algorithm (TGCA) [41], multi-local search and adaptive niching based memetic algorithm (MAMN) [78], and evolutionary algorithm for clustering large probabilistic graphs (EA-CPG) [39]. These methods selected for comparison use different strategies for automatic data clustering. In the VGA [60], a variant of traditional GA using a variable length representation is proposed as the underlying tool to search for clusters by optimizing the *I*-index validity function. In this method, neither local search operation nor diversity-preserving technique is employed. The clustering approach of WDC-NMA [21] is based on a niching MA. In this method, the *k*-means algorithm is used to fine-tune evolving clusters while a variant of the standard crowding technique [27] is incorporated to preserve the solution diversity during evolutionary data clustering. Both DB function and Bayesian information criterion have been used as the clustering criterion in this method. In the TGCA [41], the task of automatic data clustering is accomplished in two stages. In the first stage, the method mainly focuses on identifying the number of clusters in the data set. Then, in the second stage, it proceeds to evolve the partitioning of data based on the identified number of clusters. In this method, both selection and mutation rates are allowed to vary with respect to the numbers of clusters in population. In the MAMN [78], an MA incorporating with a multilocal search mechanism is proposed for automatic data clustering. The multilocal search mechanism employs three local searches (one step of *k*-means, cluster merge and cluster split operation) with different features to cooperatively exploit the clustering search spaces. Further, an adaptive restricted tournament selection technique is designed and incorporated to dynamically preserve the population diversity during the evolutionary clustering process. While the EA-CPG clustering method [39] is based on a multisubpopulation evolutionary algorithm. In this method, multiple subpopulations are used to simultaneously search for multiple clustering solutions. To enhance the search efficiency, initial clustering solutions in this method are improved by adopting a local search algorithm. From the obtained multiple solutions, the best clustering is then identified as the final solution based on the validity functions. The EA-CPG is originally used to cluster the probabilistic graph in [39]. Here, an approach that resembles this method is applied for data clustering and its performance is then compared with our proposed algorithm.



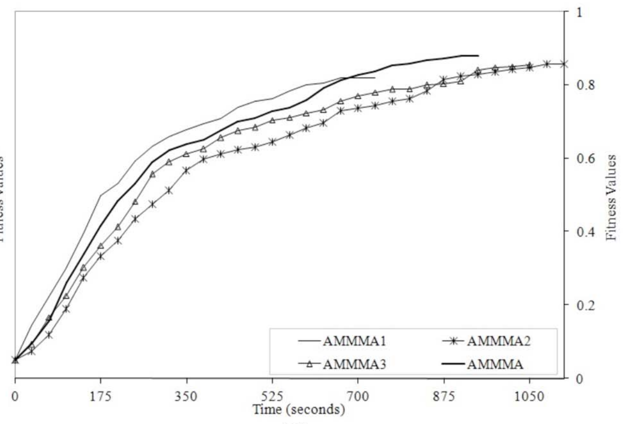
(a)



(b)



(c)



(d)

Fig. 3. Mean fitness value of the best solutions over the runtime of the four algorithms to be compared on (a) DS_16, (b) DS_30, (c) Subcell2945, and (d) Subcancer data set.

In the VGA, web document clustering based on a new niching memetic algorithm (WDC-NMA), TGCA, and EA-CPG, different validity functions were used for fitness computation or selecting the solutions. These individual validity functions, however, could be sensitive to structures of clustering problems. Employing one single validity function as the clustering criterion may therefore result in unreliable clustering solutions. To make the comparisons fair and meaningful, the clustering criterion as employed in our proposed algorithm, which consists of three well-known validity functions, is utilized in all the methods for fitness computation or selecting clustering solutions. Moreover, the population size and termination criterion are set to be the same for all methods to be compared. Additionally, the values of other free parameters of the five methods are determined using the same tuning procedure as described in Section VI-A.

The above methods are compared with respect to the effectiveness and efficiency of recovering high quality solutions in terms of fitness. Since the true cluster structure is *a priori* known for several of the data sets used in our experiments, the accuracy of recovering true clusters on these data sets for the six algorithms can also be compared. Thus, we have reported the mean fitness of solutions, the number of clusters in these solutions, the mean execution time as well as the best solutions identified out of ten trails on the data sets. For those data sets, whose true structure is known beforehand, the mean Rand index (RI) [69] has also been reported. The RI used here is to measure the agreement of clustering solutions delivered by a given algorithm and the true clusters in the data. It returns values in a range of [0, 1], higher values indicating better clustering solutions. The results are given in Table II on the eight data sets. To statistically justify the comparisons, paired Wilcoxon's rank-sum tests comparing our proposed method with the five methods have been conducted for each of the three evaluation indexes (i.e., fitness, runtime, and RI). The Wilcoxon's test is a nonparametric statistical hypothesis test, which does not assume a particular distribution for the data. The results are reported in Table III. As a null hypothesis, it is assumed that there are no significant differences between the two sets of evaluation index values delivered by the two methods, whereas the alternative hypothesis is that they differ from each other in a significant way. If the *p*-value produced by the test is below a certain threshold value, typically 0.05 (5% significance level), then the null hypothesis is rejected in favor of the alternative hypothesis.

The results in Table II show that, in comparison with five previously developed methods, our proposed algorithm is able to identify solutions with the best quality in terms of fitness. For example, on DS_16 data, the VGA, WDC-NMA, TGCA, MAMN, and EA-CPG deliver solutions with a mean fitness of 0.77, 0.87, 0.83, 0.89, and 0.82, respectively, while our proposed algorithm identifies solutions with a mean fitness

TABLE II
COMPARING RESULTS DELIVERED BY THE SIX METHODS ON EIGHT DATA SETS OVER TEN TRIALS. "NOC" IS AN ABBREVIATION FOR "NUMBER OF CLUSTERS." IN CASES THAT DIFFERENT CLUSTER NUMBERS EXIST IN THE SOLUTIONS, THE NUMBER OF TRIALS RESULTING WITH EACH CLUSTER NUMBER ARE SHOWN IN PARENTHESES. THE "COUNT" IN THE EVALUATION INDEX OF "BEST SOLUTION (FITNESS, RI, NOC)/COUNT" DENOTES THE NUMBER OF TRIALS RESULTING IN SUCH A BEST SOLUTION

Methods	**Evaluation indexes**	**Data sets**							
		DS_10	**DS_16**	**DS_30**	**Subcell384**	**Subcell2945**	**Subcancer**	**Optical digit**	**Image seg.**
VGA	**Mean fitness**	0.84	0.77	0.74	0.78	0.66	0.69	0.78	0.75
	Mean RI	0.85	0.83	0.82	0.62	N/A	N/A	0.79	0.75
	NoC (count)	7(1),8(1), 9(4),10(2), 11(1),12(1)	12(1),13(1), 14(3),16(1), 17(2),18(2)	24(1),26(2), 28(1),29(2), 30(1),31(2), 32(1)	2(1),3(3), 5(4),6(2)	2(2),3(3), 4(1),6(2), 8(1),9(1)	2(2),3(3), 4(1),6(1), 7(2),9(1)	9(2),10(2), 11(3),12(2), 13(1)	4(2),5(3), 7(1),8(2), 9(2)
	Best solution (fitness, RI, NoC)/count	(0.93, 0.95, 10)/2	(0.91, 0.95, 16)/1	(0.86, 0.88, 31)/1	(0.86, 0.73, 5)/2	(0.77, N/A, 3)/1	(0.78, N/A, 3)/1	(0.83, 0.81, 11)/1	(0.82, 0.81, 7)/1
	Mean runtime (s)	57.7	217.5	623.1	218.7	14135.5	12133.9	59564.3	12985.1
WDC-NMA	**Mean fitness**	0.89	0.87	0.88	0.83	0.78	0.82	0.85	0.81
	Mean RI	0.91	0.92	0.89	0.70	N/A	N/A	0.84	0.80
	NoC (count)	9(2),10(5), 11(3)	12(3),14(3), 16(4)	28(2),30(2), 31(4),32(2)	3(4),5(6)	3(1),4(4), 6(3),8(2)	2(3),4(3), 5(3),9(1)	10(2),11(3), 12(1),13(4)	5(3),6(1), 7(2),8(4)
	Best solution (fitness, RI, NoC)/count	(0.93, 0.95, 10)/5	(0.91, 0.95, 16)/3	(0.94, 0.94, 30)/1	(0.86, 0.73, 5)/5	(0.86, N/A, 6)/1	(0.86, N/A, 4)/1	(0.87, 0.86, 10)/1	(0.85, 0.84, 7)/1
	Mean runtime (s)	14.9	39.8	92.6	40.9	1136.7	987.6	4531.5	973.4
TGCA	**Mean fitness**	0.87	0.83	0.81	0.80	0.74	0.75	0.83	0.79
	Mean RI	0.87	0.89	0.87	0.64	N/A	N/A	0.82	0.79
	NoC (count)	8(1),9(1), 10(2),11(4), 12(2)	12(2),14(3), 16(3),18(2)	26(2),28(1), 30(2),31(3), 32(2)	2(3),3(3), 5(4)	3(3),4(1), 5(2),7(2), 8(2)	3(1),4(2), 5(3),7(3), 9(1)	9(3),10(2), 11(2),12(3)	4(2),6(1), 7(2),8(2), 9(3)
	Best solution (fitness, RI, NoC)/count	(0.93, 0.95, 10)/3	(0.91, 0.95, 16)/2	(0.91, 0.90, 30)/1	(0.86, 0.73, 5)/1	(0.82, N/A, 5)/1	(0.82, N/A, 5)/1	(0.86, 0.86, 10)/1	(0.85, 0.84, 7)/1
	Mean runtime (s)	14.2	38.1	82.9	37.6	747.7	693.6	4329.6	706.6
MAMN	**Mean fitness**	0.92	0.89	0.90	0.85	0.82	0.85	0.86	0.85
	Mean RI	0.93	0.93	0.91	0.73	N/A	N/A	0.85	0.83
	NoC (count)	9(2),10(8)	14(3),16(7)	28(3),30(5), 31(2)	5(10)	4(3),5(3), 6(2),8(2)	3(2),4(3), 5(2),7(3)	10(3),11(3), 12(4)	6(2),7(4), 8(4)
	Best solution (fitness, RI, NoC)/count	(0.93, 0.95, 10)/7	(0.91, 0.95, 16)/5	(0.94, 0.94, 30)/2	(0.86, 0.73, 5)/7	(0.86, N/A, 6)/1	(0.88, N/A, 4)/1	(0.88, 0.87, 10)/1	(0.86, 0.85, 7)/1
	Mean runtime (s)	17.2	41.4	90.9	42.4	717.6	689.5	3998.5	728.5
EA-CPG	**Mean fitness**	0.85	0.82	0.79	0.79	0.72	0.71	0.79	0.76
	Mean RI	0.86	0.87	0.85	0.63	N/A	N/A	0.80	0.77
	NoC (count)	8(1),9(1), 10(3),11(2), 12(3)	13(2),14(3), 16(2),18(3)	24(1),26(2), 28(2),29(1), 30(2),31(1), 32(1)	2(3),3(2) 5(4),6(1)	3(2),4(1), 6(3),7(1), 8(2),9(1)	3(1),5(3), 6(2),7(1), 8(3)	9(1),10(2), 11(4),12(2), 13(1)	5(3),6(2), 7(1),8(3), 9(1)
	Best solution (fitness, RI, NoC)/count	(0.93, 0.95, 10)/2	(0.91, 0.95, 16)/2	(0.87, 0.89, 30)/1	(0.86, 0.73, 5)/2	(0.80, N/A, 6)/1	(0.81, N/A, 4)/1	(0.83, 0.82, 11)/1	(0.81, 0.81, 8)/1
	Mean runtime (s)	31.8	127.9	294.3	131.7	8093.6	7632.8	30962.4	8154.2
AMMMA	**Mean fitness**	0.93	0.91	0.93	0.86	0.86	0.88	0.90	0.87
	Mean RI	0.95	0.95	0.93	0.73	N/A	N/A	0.88	0.85
	NoC (count)	10(10)	16(10)	30(6),31(4)	5(10)	4(3),6(5), 8(2)	2(2),4(5), 7(3)	10(4),11(4), 12(2)	7(5),8(5)
	Best solution (fitness, RI, NoC)/count	(0.93, 0.95, 10)/10	(0.91, 0.95, 16)/9	(0.94, 0.94, 30)/5	(0.86, 0.73, 5)/10	(0.89, N/A, 6)/1	(0.90, N/A, 4)/1	(0.92, 0.89, 10)/1	(0.89, 0.87, 7)/1
	Mean runtime (s)	18.6	45.3	95.8	43.5	1028.9	919.4	4655.1	930.5

at 0.91. From Table III, except for MAMN on DS_10 and Subcell384 data, we can see that all *p*-values produced by the paired Wilcoxon's rank-sum tests on all data sets with respect to the evaluation criterion of fitness are less than 0.05. This indicates that the better fitness produced by our method is statistically significant and has not occurred by chance.

TABLE III
STATISTICAL SIGNIFICANCE OF COMPARISONS BASED ON PAIRED WILCOXON'S RANK-SUM TESTS BETWEEN OUR PROPOSED METHOD (I.E., AMMMA) AND THE FIVE METHODS (VGA, WDC-NMA, TGCA, MAMN, AND EA-CPG) ON EIGHT DATA SETS. * INDICATES THAT THE PERFORMANCE OF AMMMA IS SIGNIFICANTLY BETTER THAN THE METHOD TO BE COMPARED WITH A CONFIDENCE LEVEL OF 95%

Methods	Evaluation indexes	*P*-values of paired Wilcoxon's rank-sum tests							
		DS_10	DS_16	DS_30	Subcell384	Subcell2945	Subcancer	Optical digit	Image seg.
VGA	Mean fitness	3.25E-3*	2.15E-6*	2.64E-7*	2.96E-3*	4.23E-8*	3.79E-9*	5.78E-6*	2.81E-6*
	Mean RI	1.46E-3*	2.64E-4*	2.89E-5*	4.39E-3*	N/A	N/A	8.13E-5*	3.72E-6*
	Mean runtime	8.72E-13*	3.23E-10*	1.43E-11*	2.24E-10*	8.19E-23*	3.26E-21*	5.59E-23*	7.53E-22*
WDC-NMA	Mean fitness	1.26E-2*	5.18E-3*	8.65E-5*	1.23E-2*	1.03E-5*	4.12E-5*	7.73E-4*	5.13E-4*
	Mean RI	1.98E-2*	7.21E-3*	7.38E-4*	1.41E-2*	N/A	N/A	4.97E-3*	3.42E-3*
	Mean runtime	4.51E-2	1.08E-1	1.37E-1	2.02E-1	9.32E-3	5.43E-2	1.03E-1	1.05E-1
TGCA	Mean fitness	9.82E-3*	1.34E-4*	5.35E-6*	6.48E-3*	8.95E-6*	7.59E-7*	3.89E-5*	4.31E-5*
	Mean RI	4.63E-3*	2.31E-3*	2.20E-4*	9.03E-3*	N/A	N/A	2.51E-5*	3.17E-5*
	Mean runtime	2.96E-2	2.92E-2	2.92E-3	1.45E-2	8.73E-8	8.74E-7	8.93E-5	4.21E-8
MAMN	Mean fitness	5.89E-2	2.29E-2*	4.87E-3*	1.02E-1	2.32E-4*	4.04E-4*	3.27E-2*	3.41E-3*
	Mean RI	6.93E-2	4.61E-2*	8.79E-3*	3.14E-1	N/A	N/A	2.45E-2*	3.71E-3*
	Mean runtime	8.47E-2	6.17E-2	3.12E-2	1.28E-1	6.72E-9	7.61E-8	6.51E-6	6.12E-8
EA-CPG	Mean fitness	7.76E-3*	8.07E-5*	1.27E-6*	4.36E-3*	3.67E-7*	8.39E-8*	2.94E-6*	2.38E-6*
	Mean RI	3.15E-3*	8.03E-4*	7.95E-5*	6.72E-3*	N/A	N/A	5.58E-5*	3.85E-5*
	Mean runtime	2.49E-11*	9.89E-8*	2.11E-11*	8.91E-8*	1.43E-21*	1.62E-21*	4.65E-22*	4.81E-21*

The worse performance of the VGA, TGCA, and EA-CPG is not surprising since they are prone to premature convergence. By incorporating the niching method, both WDC-NMA and MAMN can perform better than the VGA, TGCA, and EA-CPG. However, they are still more vulnerable to less promising local optimal solutions than our proposed algorithm. In terms of efficiency, the results show that the proposed algorithm can generally deliver solutions more efficiently than the VGA as well as EA-CPG and with comparable efficiency to the WDC-NMA, but less efficiently compared with the TGCA and MAMN. For example, on DS_30, the VGA, WDC-NMA, TGCA, MAMN, and EA-CPG need 623.1, 92.6, 82.9, 90.9, and 294.3 s, respectively, while the AMMMA takes 95.8 s on average to converge. This is also confirmed by the paired Wilcoxon's rank-sum test results, which show that the differences of runtimes are significant on all data sets between the AMMMA and the VGA as well as the EA-CPG, while between our method and WDC-NMA they are not significant on most of the data sets. Although the TGCA and MAMN can be efficient, they are still susceptible to less promising local optima, leading to a worse solution quality than our proposed algorithm.

Further, the results in Table II reveal that our proposed algorithm is generally capable of achieving more accurate solutions than the five methods to be compared in terms of the RI values on the data sets, whose true structures are known beforehand. The paired Wilcoxon's rank-sum test results in Table III show that, except for MAMN on DS_10 and Subcell384 data, the differences of RI values are significant for all paired comparisons on all data sets. The poor performance of the five methods is mainly caused by the fact that they may return less promising locally optimal solutions, which typically have low RI values. In terms of the number of clusters identified, Table II shows that our proposed method can generally more consistently recover the correct cluster numbers presented in the data sets with known structure. Particularly, the results show that, on clustering problems with relatively small search spaces, the solutions with correct number of clusters identified by our proposed algorithm usually all correspond to the best-known solutions. This is, however, not the case for those methods to be compared, which can deliver a similar number of solutions with correct number of clusters as our algorithm. For example, on the DS_30 data, although the MAMN can produce five solutions with correct number of clusters, only two of them correspond to the best-known solution with fitness of 0.94 and RI of 0.94. By contrast, five out of the six solutions with correct number of clusters provided by the AMMMA correspond the best-known solution. While, on clustering problems with relatively large search spaces, the solutions with correct number of clusters delivered by our algorithm could be significantly better than those identified by the related methods. For instance, on the image segmentation data, the best solution with correct number of clusters obtained by the MAMN gives the fitness of 0.86 and RI of 0.85, while the AMMMA has the fitness of 0.89 and RI of 0.87. Additionally, it can be noted that the proposed algorithm is capable of recovering solutions with correct number of clusters and high RI values on both DS_10 and DS_30 data sets, which contain many outlier data points. This could indicate that our algorithm is robust to the presence of outliers in the data sets.

Based on the above results, our proposed algorithm is clearly the best alternative for automatic data clustering. The performance improvement of the algorithm over five related methods is mainly attributed to the incorporation of AMC and AVMNC mechanisms. Specifically, the proposed AMC mechanism helps to ensure a diverse search over the subspaces associated with different numbers of clusters and, at the same time, an intense search of the promising subspaces. While the AVMNC mechanism helps to achieve a diverse and efficient search of the promising subspaces by forming and maintaining diverged niches of high fitness within the corresponding subpopulations. These mechanisms make the proposed algorithm

TABLE IV
COMPARING RESULTS DELIVERED BY OUR PROPOSED METHOD AND THE MULTIRUN OF *k*-MEAN ON EIGHT DATA SETS OVER TEN TRIALS. THE VALUES IN PARENTHESES ARE *p*-VALUES OF PAIRED WILCOXON'S RANK-SUM TESTS BETWEEN OUR METHOD AND THE MULTIRUN OF *k*-MEANS. * INDICATES THAT THE PERFORMANCE OF AMMMA IS SIGNIFICANTLY BETTER THAN THE MULTIRUN OF *k*-MEAN WITH A CONFIDENCE LEVEL OF 95%

Methods	Evaluation indexes	Data sets							
		DS_10	DS_16	DS_30	Subcell384	Subcell2945	Subcancer	Optical digit	Image seg.
Multi-run of *k*-means	Mean criterion value	0.85 (7.23E-3)*	0.81 (1.06E-4)*	0.80 (2.18E-6)*	0.80 (3.57E-3)*	0.69 (7.26E-8)*	0.70 (5.46E-8)*	0.77 (7.29E-7)*	0.74 (8.27E-7)*
	Mean RI	0.84 (1.79E-3)*	0.86 (6.17E-4)*	0.86 (1.17E-4)*	0.65 (8.16E-3)*	N/A	N/A	0.78 (1.09E-4)*	0.74 (2.32E-6)*
AMMMA	Mean criterion value	0.93	0.91	0.93	0.86	0.86	0.88	0.90	0.87
	Mean RI	0.95	0.95	0.93	0.73	N/A	N/A	0.88	0.85

capable of effectively and efficiently searching the solution spaces, thus improving its overall performance.

Finally, we evaluate the effectiveness of our proposed method by comparing it with the classical *k*-means algorithm. The *k*-means algorithm is known to be very efficient. However, it generally ends up with locally optimal solutions that are far away from the desired global one. Moreover, this algorithm assumes that the number of clusters existing in the data is known beforehand. By contrast, our proposed method is based on an MA with AMC and AVMNC mechanisms, which aims to deliver the optimal or near-optimal solution. Further, our method requires no *a priori* knowledge about the number of clusters. Although our method can overcome the limitations of *k*-means, it is of interest to compare it with the multirun of *k*-means. For this purpose, while making the comparison more meaningful, we run *k*-means with a range of numbers of clusters from 2 to $k^{\max}$ (the definition of which is shown in Section III-B) on each of the aforementioned data sets and evaluate the results in each case according to the criterion of (2). This process is repeated until it takes approximately the same amount of time as our method on the same data set, and the best solution is selected as the output. Ten trials of multirun of *k*-means are performed on each data set and the mean solution quality in terms of the criterion and RI are reported. The resulting criterion and RI values are then compared with those obtained by our proposed method. The results, as shown in Table IV, reveal that our method can provide better clustering solutions in terms of both criterion and RI values than the multirun of *k*-means in all cases. For instance, on Subcell384, the multirun of *k*-means provides solutions with a mean criterion and RI value of 0.80 and 0.65, while our method gives 0.86 and 0.73, respectively. Further, the *p*-values of paired Wilcoxon's rank-sum tests between our method and multirun of *k*-means in Table IV indicate that the differences of criterion values are statistically significant. Similar results are also obtained for RI comparisons, establishing the significance of the proposed method.

VII. CONCLUSION

In this paper, we propose and implement an AMC and adaptive multiniche crowding-based MA to tackle the automatic data clustering. The objective is to effectively and efficiently identify high quality solutions. To this end, an AMC mechanism along with an adaptive multiniche crowding scheme have been developed and incorporated in the algorithm. The AMC mechanism is designed to ensure a diverse search over solution subspaces associated with different numbers of clusters while allowing more promising subspaces to be more intensively searched. While the adaptive multiniche crowding scheme is devised to promote a diverse search of the subspace by forming and maintaining diverged niches of high fitness within the subpopulation. The significance of the devised AMC mechanism and adaptive niching scheme has been clearly demonstrated with extensive experimental results, which confirm that they are able to work together to properly search complex decision spaces of automatic clustering problems. The proposed algorithm has therefore been capable of effectively and efficiently deliver high quality solutions, outperforming other methods implemented for comparison.

The work presented here can be extended in several directions. First, it will be interesting to verify the usefulness of the proposed AMC mechanism and adaptive multiniche crowding scheme by incorporating them either individually or jointly into other meta-heuristic methods, e.g., particle swarm optimization, for data clustering. Accordingly, a comparison of these developed methods with our proposed algorithm for data clustering can also be conducted. Second, we would be also interested to design other indexes to measure the performance of subpopulation and the diversity of subpopulation in the proposed algorithm. Their influence on the algorithm's performance can then be studied. Third, the proposed algorithm can be suitably modified to make it applicable to other problems (e.g., text mining, scheduling policy design [87], and classification [99]), a direction worth to be investigated. Finally, while we have illustrated the effectiveness of the proposed algorithm to identify a single optimal or near optimal solution, the algorithm is particularly flexible to be extended for multiple optimal solution identification and tracking dynamically changing landscapes, which is desirable for many applications. This thus leads to another interesting topic for future research.

REFERENCES

[1] S. F. Adra and P. J. Fleming, "Diversity management in evolutionary many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 15, no. 2, pp. 183–196, Apr. 2011.

[2] L. E. Agustin-Blas *et al.*, "A new grouping genetic algorithm for clustering problems," *Expert Syst. Appl.*, vol. 39, no. 10, pp. 9695–9703, 2012.

[3] M. N. Ahmed, S. M. Yamany, N. Mohamed, A. A. Farag, and T. Moriarty, "A modified fuzzy *c*-means algorithm for bias field estimation and segmentation of MRI data," *IEEE Trans. Med. Imag.*, vol. 21, no. 3, pp. 193–199, Mar. 2002.

[4] E. Alba and G. Luque, "Theoretical models of selection pressure for dEAs: Topology influence," in *Proc. IEEE Int. Conf. Evol. Comput.*, Edinburgh, U.K., 2005, pp. 214–221.

[5] E. Alba, G. Luque, and S. Nesmachnow, "Parallel metaheuristics: Recent advances and new trends," *Int. Trans. Oper. Res.*, vol. 20, no. 1, pp. 1–48, 2013.

[6] E. Alba and M. Tomassini, "Parallelism and evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 6, no. 5, pp. 443–462, Oct. 2002.

[7] D. Aloise, A. Deshpande, P. Hansen, and P. Popat, "NP-hardness of Euclidean sum-of-squares clustering," *Mach. Learn.*, vol. 75, no. 2, pp. 245–248, 2009.

[8] E. Alba and M. Tomassini, "Parallelism and evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 6, no. 5, pp. 443–462, Oct. 2002.

[9] L. Araujo and J. Merelo, "Diversity through multiculturality: Assessing migrant choice policies in an island model," *IEEE Trans. Evol. Comput.*, vol. 15, no. 4, pp. 456–469, Aug. 2011.

[10] I. Arnaldo, I. Contreras, D. Milln-Ruiz, J. I. Hidalgo, and N. Krasnogor, "Matching island topologies to problem structure in parallel evolutionary algorithms," *Soft Comput.*, vol. 17, no. 7, pp. 1209–1225, 2013.

[11] S. Biswas, S. Kundu, and S. Das, "Inducing niching behavior in differential evolution through local information sharing," *IEEE Trans. Evol. Comput.*, vol. 19, no. 2, pp. 246–263, Apr. 2015.

[12] D. Brugger, M. Bogdan, and W. Rosenstiel, "Automatic cluster detection in Kohonen's SOM," *IEEE Trans. Neural Netw.*, vol. 19, no. 3, pp. 442–459, Mar. 2008.

[13] R. B. Calinski and J. Harabasz, "A dendrite method for cluster analysis," *Commun. Stat.*, vol. 3, no. 1, pp. 1–27, 1974.

[14] C. Candan, A. Goeffon, F. Lardeux, and F. Saubion, "A dynamic island model for adaptive operator selection," in *Proc. Conf. Genet. Evol. Comput.*, Philadelphia, PA, USA, 2012, pp. 1253–1260.

[15] E. Cantú-Paz, "Migration policies, selection pressure, and parallel evolutionary algorithms," *J. Heuristics*, vol. 7, no. 4, pp. 311–334, 2001.

[16] W. Cedeño, V. R. Vemuri, and T. Slezak, "Multiniche crowding in genetic algorithms and its application to the assembly of DNA restriction-fragments," *Evol. Comput.*, vol. 2, no. 4, pp. 321–345, Dec. 1994.

[17] W.-C. Chen and R. Maitra, "Model-based clustering of regression time series data via APECM—An AECM algorithm sung to an even faster beat," *Stat. Anal. Data Min.*, vol. 4, no. 6, pp. 567–578, 2011.

[18] X. S. Chen, Y. S. Ong, M. H. Lim, and T. C. Tan, "A multi-facet survey on memetic computation," *IEEE Trans. Evol. Comput.*, vol. 15, no. 5, pp. 591–607, Oct. 2011.

[19] R. J. Cho *et al.*, "A genome-wide transcriptional analysis of the mitotic cell cycle," *Mol. Cell*, vol. 2, no. 1, pp. 65–73, 1998.

[20] A. D. Cioppa, C. De Stefano, and A. Marcelli, "Where are the niches? Dynamic fitness sharing," *IEEE Trans. Evol. Comput.*, vol. 11, no. 4, pp. 453–465, Aug. 2007.

[21] C. Cobos, C. Montealegre, M. F. Mejia, M. Mendoza, and E. León, "Web document clustering based on a new niching memetic algorithm, term-document matrix and Bayesian information criterion," in *Proc. IEEE Congr. Evol. Comput.*, Barcelona, Spain, 2010, pp. 1–8.

[22] G. Corriveau, R. Guilbault, A. Tahan, and R. Sabourin, "Review and study of genotypic diversity measures for real-coded representations," *IEEE Trans. Evol. Comput.*, vol. 16, no. 5, pp. 695–710, Oct. 2012.

[23] M. Èrepinšek, S. H. Liu, and M. Mernik, "Exploration and exploitation in evolutionary algorithms: A survey," *ACM Comput. Surveys*, vol. 45, no. 3, 2014, Art. ID 35.

[24] S. Dasa, S. Maity, B.-Y. Qub, and P. N. Suganthan, "Real-parameter evolutionary multimodal optimization—A survey of the state-of-the-art," *Swarm Evol. Comput.*, vol. 1, no. 2, pp. 71–88, 2011.

[25] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-1, no. 2, pp. 224–227, Apr. 1979.

[26] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. New York, NY, USA: Wiley, 2001.

[27] K. A. De Jong, "An analysis of the behaviors of a class of genetics adaptive systems," Ph.D. dissertation, Dept. Comput. Commun. Sci., Univ. Michigan, Ann Arbor, MI, USA, 1975.

[28] D. Dembele and P. Kastner, "Fuzzy C-means method for clustering microarray data," *Bioinformatics*, vol. 19, no. 8, pp. 973–980, 2003.

[29] I. De Falco, A. D. Cioppa, D. Maisto, U. Scafuri, and E. Tarantino, "Biological invasion–inspired migration in distributed evolutionary algorithms," *Inf. Sci.*, vol. 207, pp. 50–65, Nov. 2012.

[30] E. Falkenauer, *Genetic Algorithms and Grouping Problems*. New York, NY, USA: Wiley, 1998.

[31] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, no. 5814, pp. 972–976, 2007.

[32] M. L. L. García, R. García-Ródenas, and A. G. Gómez, "Hybrid metaheuristic optimization algorithms for time-domain-constrained data clustering," *Appl. Soft Comput.*, vol. 23, pp. 319–332, Oct. 2014.

[33] I. Gath and A. B. Geva, "Unsupervised optimal fuzzy clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 7, pp. 773–781, Jul. 1989.

[34] K. Georgieva and A. P. Engelbrecht, "A cooperative multi-population approach to clustering temporal data," in *Proc. IEEE Congr. Evol. Comput.*, Cancún, Mexico, 2013, pp. 1983–1991.

[35] D. E. Goldberg and J. Richardson, "Genetic algorithms with sharing for multimodal function optimization," in *Proc. Int. Conf. Genet. Algorithm*, Cambridge, MA, USA, 1987, pp. 41–49.

[36] D. E. Goldberg, *Genetic Algorithm in Search, Optimization and Machine Learning*. Reading, MA, USA: Addison-Wesley Press, 1989.

[37] J. J. Grefenstette, "Optimization of control parameters for genetic algorithms," *IEEE Trans. System, Man, Cybern.*, vol. 16, no. 1, pp. 122–128, Jan. 1986.

[38] J. Handl and J. Knowles, "An evolutionary approach to multiobjective clustering," *IEEE Trans. Evol. Comput.*, vol. 11, no. 1, pp. 56–76, Feb. 2007.

[39] Z. Halim, M. Waqas, and S. F. Hussain, "Clustering large probabilistic graphs using multi-population evolutionary algorithm," *Inf. Sci.*, vol. 317, pp. 78–95, Oct. 2015.

[40] G. R. Harik, "Finding multimodal solutions using restricted tournament selection," in *Proc. Int. Conf. Genet. Algorithms*, Pittsburgh, PA, USA, 1995, pp. 24–31.

[41] H. He and Y. Tan, "A two-stage genetic algorithm for automatic clustering," *Neurocomputing*, vol. 81, pp. 49–59, Apr. 2012.

[42] E. R. Hruschka, R. J. G. B. Campello, A. A. Freitas, and A. C. P. L. F. de Carvalho, "A survey of evolutionary algorithms for clustering," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 39, no. 2, pp. 133–155, Mar. 2009.

[43] E. R. Hruschka and N. F. F. Ebecken, "A genetic algorithm for cluster analysis," *Intell. Data Anal.*, vol. 7, no. 1, pp. 15–25, 2003.

[44] A. Jaafar, B. Sareni, and X. Roboam, "Clustering analysis of railway driving missions with niching," *Int. J. Comput. Math.*, vol. 31, no. 3, pp. 920–931, 2012.

[45] A. K. Jain, "Data clustering—50 years beyond *K*-means," *Pattern Recognit. Lett.*, vol. 31, no. 8, pp. 651–666, 2010.

[46] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," *ACM Comput. Surveys*, vol. 31, no. 3, pp. 264–323, 1999.

[47] G. Karafotias, M. Hoogendoorn, and A. E. Eiben, "Parameter control in evolutionary algorithms: Trends and challenges," *IEEE Trans. Evol. Comput.*, vol. 19, no. 2, pp. 167–187, Apr. 2015.

[48] L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*. New York, NY, USA: Wiley, 1990.

[49] E. E. Korkmaz, "Multi-objective genetic algorithms for grouping problems," *Appl. Intell.*, vol. 33, no. 2, pp. 179–192, 2010.

[50] R. Krishnapuram and C.-P. Freg, "Fitting an unknown number of lines and planes to image data through compatible cluster merging," *Pattern Recognit.*, vol. 25, no. 4, pp. 385–400, 1992.

[51] F. Lardeux and A. Goëffon, *A Dynamic Island-Based Genetic Algorithms Framework* (LNCS 6457). Heidelberg, Germany: Springer, 2010, pp. 156–165.

[52] C.-G. Lee, D.-H. Cho, and H.-K. Jung, "Niching genetic algorithm with restricted competition selection for multimodal function optimization," *IEEE Trans. Magn.*, vol. 34, no. 3, pp. 1722–1725, May 1999.

[53] A. Leitão, F. B. Pereira, and P. Machado, "Enhancing cluster geometry optimization with island models," in *Proc. IEEE Congr. Evol. Comput.*, Brisbane, QLD, Australia, 2012, pp. 1–8.

[54] X. Li, "Niching without niching parameters: Particle swarm optimization using a ring topology," *IEEE Trans. Evol. Comput.*, vol. 14, no. 1, pp. 150–169, Feb. 2010.

[55] T. Y. Lim, "Structured population genetic algorithms: A literature survey," *Artif. Intell. Rev.*, vol. 41, no. 3, pp. 385–399, 2014.

[56] Y. Lu, S. Lu, F. Fotouhi, Y. Deng, and S. J. Brown, "Incremental genetic *k*-means algorithm and its applications in gene expression data analysis," *BMC Bioinformat.*, vol. 28, no. 5, pp. 172–182, 2004.

[57] P. C. H. Ma, K. C. C. Chan, X. Yao, and D. K. Y. Chiu, "An evolutionary clustering algorithm for gene expression microarray data analysis," *IEEE Trans. Evol. Comput.*, vol. 10, no. 3, pp. 296–314, Jun. 2006.

[58] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. Berkeley Symp. Math. Stat. Probabil.*, Berkeley, CA, USA, 1967, pp. 281–297.

[59] Y. Maeda, M. Ishita, and Q. Li, "Fuzzy adaptive search method for parallel genetic algorithm with island combination process," *Int. J. Approximate Reasoning*, vol. 41, no. 1, pp. 59–73, 2006.

[60] U. Maulik and S. Bandyopadhyay, "Nonparametric genetic clustering: Comparison of validity indices," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 31, no. 1, pp. 120–125, Feb. 2001.

[61] X.-L. Meng and D. Van Dyk, "The EM algorithm—An old folk-song sung to a fast new tune," *J. Roy. Stat. Soc. B*, vol. 59, no. 3, pp. 511–567, 1997.

[62] A. Mukhopadhyay, U. Maulik, S. Bandyopadhyay, and C. A. C. Coello, "Survey of multiobjective evolutionary algorithms for data mining: Part II," *IEEE Trans. Evol. Comput.*, vol. 18, no. 1, pp. 20–35, Feb. 2014.

[63] P. M. Murphy and D. W. Aha, "UCI repository for machine learning databases," Dept. Inf. Comput. Sci., Univ. California, Irvine, CA, USA, Tech. Rep. 94-02, 1994. [Online]. Available: http://www.ics.uci.edu/mlearn/MLRepository.html

[64] R. Nock and F. Nielsen, "On weighting clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 8, pp. 1223–1235, Aug. 2006.

[65] E. Noda, A. L. V. Coelho, I. L. M. Ricarte, A. Yamakami, and A. A. Freitas, "Devising adaptive migration policies for cooperative distributed genetic algorithms," in *Proc. IEEE Int. Conf. System Man Cybern.*, vol. 6. Hammamet, Tunisia, 2002, pp. 700–707.

[66] M. M. Paydar and M. Saidi-Mehrabad, "A hybrid genetic-variable neighborhood search algorithm for the cell formation problem based on grouping efficacy," *Comput. Oper. Res.*, vol. 40, no. 4, pp. 980–990, 2013.

[67] A. Petrowski, "A clearing procedure as a niching method for genetic algorithms," in *Proc. IEEE Int. Conf. Evol. Comput.*, Nagoya, Japan, 1996, pp. 798–803.

[68] M. R. N. Raeesi and Z. Kobti, "Heterogeneous multi-population cultural algorithm," in *Proc. IEEE Congr. Evol. Comput.*, Cancún, Mexico, 2013, pp. 292–299.

[69] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *J. Amer. Stat. Soc.*, vol. 66, no. 336, pp. 846–850, 1971.

[70] V. J. Rayward-Smith, "Metaheuristics for clustering in KDD," in *Proc. IEEE Congr. Evol. Comput.*, Edinburgh, U.K., 2005, pp. 2380–2387.

[71] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science*, vol. 344, no. 6191, pp. 1492–1496, 2014.

[72] D. T. Ross *et al.*, "Systematic variation in gene expression patterns in human cancer cell lines," *Nat. Genet.*, vol. 24, no. 3, pp. 227–235, 2000.

[73] M. Ruciñski, D. Izzo, and F. Biscani, "On the impact of the migration topology on the island model," *Parallel Comput.*, vol. 36, nos. 10–11, pp. 555–571, 2010.

[74] B. Sareni and L. Krähenbüh, "Fitness sharing and niching methods revisited," *IEEE Trans. Evol. Comput.*, vol. 2, no. 3, pp. 97–106, Sep. 1998.

[75] S. B. Gee, K. C. Tan, V. A. Shim, and N. R. Pal, "Online diversity assessment in evolutionary multiobjective optimization: A geometrical perspective," *IEEE Trans. Evol. Comput.*, vol. 19, no. 4, pp. 542–559, Aug. 2015.

[76] M. Serpell and J. E. Smith, "Self-adaptation of mutation operator and probability for permutation representations in genetic algorithms," *Evol. Comput.*, vol. 18, no. 3, pp. 491–514, 2010.

[77] R. Sharan and R. Shamir, "CLICK: A clustering algorithm with application to gene expression analysis," in *Proc. Conf. Intell. Syst. Mol. Biol.*, 2000, pp. 307–316.

[78] W. Sheng, S. Y. Chen, F. Michael, X. Gang, and J. Mao, "Multilocal search and adaptive niching based memetic algorithm with a consensus criterion for data clustering," *IEEE Trans. Evol. Comput.*, vol. 18, no. 5, pp. 721–741, Oct. 2014.

[79] W. Sheng, X. Liu, and M. Fairhurst, "A niching memetic algorithm for simultaneous clustering and feature selection," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 7, pp. 868–879, Jul. 2008.

[80] W. Sheng, S. Swift, L. Zhang, and X. Liu, "A weighted sum validity function for clustering with a hybrid niching genetic algorithm," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 35, no. 6, pp. 1156–1167, Dec. 2005.

[81] W. Sheng, A. Tucker, and X. Liu, "A niching genetic *k*-means algorithm and its applications to gene expression data," *Soft Comput.*, vol. 14, no. 1, pp. 9–19, 2010.

[82] O. M. Shir, M. Emmerich, and T. Bäck, "Adaptive niche radii and niche shape approaches for niching with the CMA-ES," *Evol. Comput.*, vol. 18, no. 1, pp. 97–126, 2010.

[83] G. Singh and K. Deb, "Comparison of multi-modal optimization algorithms based on evolutionary algorithms," in *Proc. Genet. Evol. Comput. Conf.*, Seattle, WA, USA, 2006, pp. 1305–1312.

[84] K. G. Srinivasa, K. R. Venugopal, and L. M. Patnaik, "A self-adaptive migration model genetic algorithm for data mining applications," *Inf. Sci.*, vol. 177, no. 20, pp. 4295–4313, 2007.

[85] C. Stoean, M. Preuss, R. Stoean, and D. Dumitrescu, "Multimodal optimization by means of a topological species conservation algorithm," *IEEE Trans. Evol. Comput.*, vol. 14, no. 6, pp. 842–864, Dec. 2010.

[86] C.-H. Su and T.-H. Hou, "Using multi-population intelligent genetic algorithm to find the Pareto-optimal parameters for a nano-particle milling process," *Expert Syst. Appl.*, vol. 34, no. 4, pp. 2502–2510, 2008.

[87] N. Su, M. Zhang, M. Johnston, and K. C. Tan, "Automatic design of scheduling policies for dynamic multi-objective job shop scheduling via cooperative coevolution genetic programming," *IEEE Trans. Evol. Comput.*, vol. 18, no. 2, pp. 193–208, Apr. 2014.

[88] R. Tanese, "Distributed genetic algorithms," in *Proc. 3rd Int. Conf. Genet. Algorithms*, Fairfax, VA, USA, 1989, pp. 434–439.

[89] S. Tavazoie, D. Hughes, J. M. J. Campbell, R. J. Cho, and G. M. Church, "Systematic determination of genetic network architecture," *Nat. Genet.*, vol. 22, no. 3, pp. 281–285, 1999.

[90] L. Y. Tseng and S. B. Yang, "A genetic approach to the automatic clustering problem," *Pattern Recognit.*, vol. 34, no. 2, pp. 415–424, 2001.

[91] D. Schlierkamp-Voosen and H. Mühlenbein, "Strategy adaptation by competing subpopulations," in *Proc. Parallel Probl. Solving Nat.*, Jerusalem, Israel, 1994, pp. 199–208.

[92] K.-C. Wong, C.-H. Wu, R. K. P. Mok, C. Peng, and Z. Zhang, "Evolutionary multimodal optimization using the principle of locality," *Inf. Sci.*, vol. 194, pp. 138–170, Jul. 2012.

[93] J. Xiao, Y. P. Yan, J. Zhang, and Y. Tang, "A quantum-inspired genetic algorithm for *k*-means clustering," *Expert Syst. Appl.*, vol. 37, no. 7, pp. 4966–4973, 2010.

[94] L. Xu, T. W. S. Chow, and E. W. M. Ma, "Topology-based clustering using polar self-organizing map," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 4, pp. 798–807, Apr. 2015.

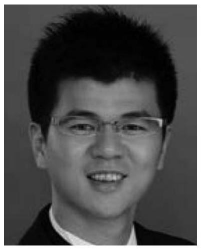
[95] E. L. Yu and P. N. Suganthan, "Ensemble of niching algorithms," *Inf. Sci.*, vol. 180, no. 15, pp. 2815–2833, 2010.

[96] S. Yu *et al.*, "Optimized data fusion for kernel *k*-means clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 5, pp. 1031–1039, May 2012.

[97] M. Yuwono, S. W. Su, B. D. Moulton, and H. T. Nguyen, "Data clustering using variants of rapid centroid estimation," *IEEE Trans. Evol. Comput.*, vol. 18, no. 3, pp. 366–377, Jun. 2014.

[98] Y. Zheng, J. Byeungwoo, D. Xu, Q. M. Wu, and H. Zhang, "Image segmentation by generalized hierarchical fuzzy *C*-means algorithm," *J. Intell. Fuzzy Syst.*, vol. 28, no. 2, pp. 961–973, 2015.

[99] Y.-J. Zheng, H.-F. Ling, J.-Y. Xue, and S.-Y. Chen, "Population classification in fire evacuation: A multiobjective particle swarm optimization approach," *IEEE Trans. Evol. Comput.*, vol. 18, no. 1, pp. 70–81, Feb. 2014.



Weiguo Sheng (M'13) received the M.Sc. degree in information technology from University of Nottingham, Nottingham, U.K., in 2002 and the Ph.D. degree in computer science from Brunel University, Uxbridge, U.K., in 2005.

He was a Researcher with University of Kent, Canterbury, U.K., and Royal Holloway, University of London, Egham, U.K. In 2011 he moved to Zhejiang University of Technology, Hangzhou, China, where he is currently a Professor of Computer Science. His research interests include evolutionary computations, data mining/clustering, pattern recognition, and machine learning.

Shengyong Chen (M'01–SM'10) received the Ph.D. degree in computer vision from City University of Hong Kong, Hong Kong, in 2003.

He joined Zhejiang University of Technology, Hangzhou, China, in 2004, where he is currently a Professor with the Department of Computer Science. He was with University of Hamburg, Hamburg, Germany, from 2006 to 2007. He has published over 100 scientific papers in international journals and conferences. His research interests include computer vision, robotics, and image analysis.

Dr. Chen received the National Outstanding Youth Foundation Award of China in 2013 and the Fellowship from the Alexander von Humboldt Foundation of Germany. He is a fellow of IET and a Committee Member of IET Shanghai Branch.

Mengmeng Sheng received the M.Sc. degree in computer science from University of Hong Kong, Hong Kong, in 2008.

She was a Data Engineer with China Telecom, Hong Kong. In 2013 she moved to Zhejiang Police College, Hangzhou, China, where she is currently a Lecturer of Computer Science. Her research interests include data mining, machine learning, and computer vision.

Gang Xiao received the master's degree from Tsinghua University, Beijing, China, in 1992 and the Ph.D. degree from the Zhejiang University of Technology, Hangzhou, China, in 2011.

In 1985 he joined Zhejiang University of Technology, where he is currently a Professor with the School of Computer Science and Technology. His research interests include digital image processing, water quality testing, and computer-aided design.

Jiafa Mao received the Ph.D. degree in pattern recognition from East China University of Science and Technology, Shanghai, China, in 2009.

He was a Post-Doctoral Researcher with Beijing University of Posts and Telecommunications, Beijing, China. In 2011 he joined Zhejiang University of Technology, Hangzhou, China, where he is currently an Associate Professor with the School of Computer Science and Technology. He has published over 30 papers in the scientific literature. His research interests include pattern recognition, digital image processing, and information hiding.

Yujun Zheng received the Ph.D. degree from the Institute of Software, Chinese Academy of Sciences, Beijing, in 2010.

He is an Associate Professor and a Ph.D. Advisor with Zhejiang University of Technology, Hangzhou, China. He has published over 60 scientific papers in journals, including IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, IEEE TRANSACTIONS ON FUZZY SYSTEMS, and IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS. His research interests include nature-inspired computation and its applications.

Dr. Zheng received the 2014 IFORS Prize for Development (Runner-Up) due to his work on intelligent scheduling of emergency engineering rescue in China. He is a member of ACM.