Appl Intell (2012) 36:788–799
DOI 10.1007/s10489-011-0295-y

A two-leveled symbiotic evolutionary algorithm for clustering problems

Kyoung Seok Shin · Young-Seon Jeong · Myong K. Jeong

Published online: 8 July 2011
© Springer Science+Business Media, LLC 2011

Abstract Because of its unsupervised nature, clustering is one of the most challenging problems, considered as a NP-hard grouping problem. Recently, several evolutionary algorithms (EAs) for clustering problems have been presented because of their efficiency for solving the NP-hard problems with high degree of complexity. Most previous EA-based algorithms, however, have dealt with the clustering problems given the number of clusters (K) in advance. Although some researchers have suggested the EA-based algorithms for unknown K clustering, they still have some drawbacks to search efficiently due to their huge search space. This paper proposes the two-leveled symbiotic evolutionary clustering algorithm (TSECA), which is a variant of coevolutionary algorithm for unknown K clustering problems. The clustering problems considered in this paper can be divided into two sub-problems: finding the number of clusters and grouping the data into these clusters. The two-leveled framework of TSECA and genetic elements suitable for each sub-problem are proposed. In addition, a neighborhood-based evolutionary strategy is employed to maintain the population diversity. The performance of the proposed algorithm is compared with some popular evolutionary algorithms using the real-life and simulated synthetic data sets. Experimental results show that TSECA produces more compact clusters as well as the accurate number of clusters.

K.S. Shin · M.K. Jeong (✉)
Rutgers Center for Operations Research (RUTCOR), Rutgers, The State University of New Jersey, Piscataway, USA
e-mail: mjeong@rci.rutgers.edu

Y.-S. Jeong (✉) · M.K. Jeong
Department of Industrial and Systems Engineering, Rutgers, The State University of New Jersey, Piscataway, USA
e-mail: ysjeong@eden.rutgers.edu

Keywords Clustering · Symbiotic evolutionary algorithm · Two-leveled structure · Coevolutionary algorithm

1 Introduction

Clustering is to group objects into subsets that have some meaning in the context of a particular problem and has been widely applied in variety of fields such as pattern recognition, image processing, and biotechnology [20, 51]. The aim of clustering is to maximize the similarity within the groups and the dissimilarity between two different groups, respectively. Given N input patterns $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ where $\mathbf{x}_j = (x_{j1}, x_{j2}, \ldots, x_{jp})$, K clusters are represented by $\mathbf{C} = \{C_1, C_2, \ldots, C_K\}$ with the following properties [51]:

(1) $C_i \neq \phi$ for $i = 1, 2, \ldots, K$.
(2) $C_i \cap C_j = \phi$ for $i = 1, 2, \ldots, K$, $j = 1, 2, \ldots, K$ and $i \neq j$.
(3) $\bigcup_{i=1}^{K} C_i = \mathbf{X}$.

Clustering techniques can be generally divided into two categories: hierarchical and non-hierarchical [10, 22]. Hierarchical clustering techniques proceed by either agglomerative hierarchical methods or divisive hierarchical methods. While the agglomerative hierarchical methods merge each similar group with initial many groups, the divisive hierarchical methods divide an initial single group of objects into each subgroup so that the objects in one subgroup are dissimilar to the objects in the other. The simple and popular methods for agglomerative hierarchical clustering include single linkage, compete linkage, and average linkage techniques [24].

One of the popular algorithms in non-hierarchical methods is the K-means algorithm proposed by Tou and Gonzalez [46]. This algorithm has been utilized in a variety of

applications because of its simple implementation and easy applicability. However, the big drawback of this algorithm is that it is too sensitive to an initial solution so that it is highly possible to converge to a local optimal solution [2]. As for searching a global optimal clustering method, there have been many other techniques such as fuzzy set, neural networks, genetic algorithm, and ant-based clustering methods [17, 18, 30, 38, 39, 48]. Please refer to some survey papers for details [20, 51].

From an optimization perspective, the problem of clustering N objects into K clusters can be viewed as a particular kind of NP-hard grouping problem [20, 51]. As the size of data sets with high dimension increases, it is practically infeasible to find the best number of clusters and the corresponding subsets optimizing the given objective function. Due to this reason, several researches have focused on developing heuristic algorithms for the near optimal solution such as simulated annealing [7, 43], tabu search [3, 44], and evolutionary (genetic) algorithm [20].

Among them, evolutionary algorithm (EA), which is probabilistic search technique modeled by the principles of evolution and natural selection, has widely been adopted to the clustering problems because they are effective on providing near optimal solutions in acceptable time [20]. To apply EA to a particular problem, some components, such as solution encoding scheme, selection scheme, and crossover/mutation methods, should be developed or chosen because the performance of EA depends on these components [13]. EA-based clustering approaches differ mainly in the encoding scheme, representing a potential solution of the problem as a chromosome, which is a critical part of EA. Kuncheva and Bezdek [31] proposed an EA-based clustering with a binary encoding. In this scheme, a clustering solution (partition) is represented as a binary string of length N, where N is the number of input data. If the value of the i-th gene is 1, then i-th point is the center of a cluster, and then the other points corresponding to genes with 0 are assigned to one of the clusters using an appropriate rule. Murthy and Chowdhury [37] and Lu et al. [32] presented an integer encoding scheme. A chromosome by the integer scheme consists of N integer genes. Each gene position corresponds to a particular point (object) and each gene value indicates the cluster number to which the point belongs. This scheme is simple and natural but has a drawback of redundancy. There could be, that is, many different chromosomes representing the same solution.

In addition, EA-based methods using the real number scheme have been suggested [5, 12, 34]. In the real number scheme, a chromosome contains the coordinates of the cluster centers. Accordingly, for clustering p-dimensional data into K clusters, the length of a chromosome is pK. Maulik and Bandyopadhyay [34] proposed an EA-based clustering algorithm where EA is used to search for the appropriate cluster centers. They used the sum of the Euclidean distances of the points from their respective cluster centers as the clustering metric. Bandyopadhyay and Maulik [5] developed the KGA-clustering algorithm, which combined EA with K-means algorithm. In KGA-clustering, EA was used to search for the initial cluster centers, and then K-means algorithm exploited optimal clustering centers, avoiding the major drawback of the K-means algorithm in which the clustering tends to be dependent on the choice of the initial cluster centers. Garai and Chaudhuri [12] proposed the two-phase genetic clustering algorithm. At the first phase, in their algorithm, the original data set is decomposed into some number of fragmented clusters and at the second phase, some of those fragmented clusters are combined into complete K cluster by using an iterative EA process. The advantage of this algorithm is to reduce the complexity of problem so that overall computational time could be reduced.

Even though previous EA-based clustering algorithms have showed good performance for clustering, these algorithms are only available when the number of clusters is known a priori. Unfortunately, however, the number of clusters in general is unknown, and when the number of clusters is not easy to guess, clustering becomes a tedious trial-and-error work and the clustering result is often not promising. Several EA-based approaches to handle the unknown K clustering have been suggested. Tseng and Yang [47] proposed a novel evolutionary clustering algorithm using a binary representation scheme. In their approach, the nearest-neighbor algorithm searches for a proper number of clusters and EA classifies the points into these clusters at the same time. However, this algorithm had a tendency to shrink the solution space because data points of near distance were grouped and they were considered as one point for clustering. Bandyopadhyay and Maulik [6] suggested an EA-based algorithm that adopted the real number scheme. In the paper, the number of clusters is assumed to be in the range of the minimum and maximum number of clusters. To fix the length of a chromosome, a 'do not care' symbol # is used to fill in chromosomes whose K is less than the maximum number of clusters. In addition, EA-based clustering algorithms using integer representation scheme were suggested [8, 21]. However, those conventional representation schemes and the frameworks of standard EA have a limitation in searching the solution space efficiently because the integrated problem comprising two sub-problems is very complex and has huge search space [28].

On the other hand, the clustering problem in the paper can be viewed as a multi-objective problem. For the clustering objectives, there could be inter-cluster distance, intra-clustering distance, and the number of clusters, and so on. One of the most popular methods for dealing with a multi-objective problem is to transform it into the single-objective

problem with weighted objective functions such as DB index [9], considering both inter-cluster distance and intra-cluster distance.

This paper deals with the clustering problem finding the number of clusters and grouping the data into the clusters at the same time while minimizing a particular objective function. To solve the unknown K clustering problem, we propose a new efficient algorithm, named Two-leveled Symbiotic Evolutionary Clustering Algorithm (TSECA), which is a variant of coevolutionary algorithm. Coevolutionary algorithm is known as a very efficient tool to solve the integrated optimization problems with high degree of complexity compared to classical ones [28, 36]. A common hypothesis for coevolutionary algorithm is that several parallel searches for different pieces of the solution are more efficient than single search for the entire solution. Based on the good property of coevolutionary algorithm, TSECA decomposes the clustering problem into two sub-problems; finding the number of clusters and grouping the data into these clusters. In the lower level, there exist two populations for finding the number of clusters and grouping the data into each cluster, respectively. The chromosomes in the populations are separated, but interact with each other while evolving. At the upper level, the population consists of chromosomes that are combined with those in the lower level, each of which represents the complete solution to the entire problem. Since the good combinations of the chromosomes obtained from the interaction within the lower level are transmitted to the upper level, it is highly possible to finally obtain a good solution (the number of clusters and the data set of each cluster) in the upper level. To our best knowledge, this is the first study on considering coevolutionary algorithm with clustering problems. In addition, a neighborhood-based evolutionary strategy is employed to maintain the population diversity. In order to show the effectiveness of the proposed algorithm, we perform some experiments with a real-life application, automatic clustering of spatial defects on wafer, and a variety of synthetic clustering problems with different number of input variables.

The remainder of this paper is organized as follows. Section 2 reviews a conventional evolutionary and coevolutionary algorithm. Section 3 presents the concept and the structure of the proposed algorithm, and the components consisting of the proposed algorithm is described in Section 4. Section 5 contains experimental results with real life and simulated examples. Finally, we present conclusions and some remarks on the future research in Section 6.

2 Review of evolutionary and coevolutionary algorithm

Evolutionary algorithm (EA) is a stochastic search method that takes its inspiration from natural selection and survival of the fittest in the biological world [14, 15, 35]. The origins of EA can be traced back to the late 1950s, and since the 1970s several evolutionary methodologies have been proposed, mainly genetic algorithms, evolutionary programming, and evolution strategies [1, 4, 11]. All of these approaches operate on a set of candidate solutions that are represented by a chromosome, i.e. a solution to the problem. As in the case of biological evolution, EA has a mechanism of selecting fitter chromosomes at each generation. To simulate the process of evolution, the selected chromosomes undergo genetic operations, such as crossover and mutation. This evolving process is repeated until a termination condition is satisfied. EA has received considerable attention and has been successfully applied in many real-life optimization problems [13]. The outline of conventional EAs is as follows;

1. Initialization: Generate a population of chromosomes.
2. Evaluation: Calculate the fitness of each chromosome in the population.
3. Evolution: Create new chromosomes (offspring) by applying selection, crossover, and mutation to current chromosomes (parents).
4. Termination condition: If the condition is met, then stop the algorithm and return the best chromosome. If not, go to 2.

The process of evolving in conventional EA assumes that one population of individuals is alone in adapting to a fixed environment. However, the environment is actually a composite consisting of both the physical environment and other independently action biological populations of individuals which are simultaneously actively adapting to their environment.

Coevolutionary algorithm is a search algorithm that imitates the biological coevolution that is a series of reciprocal changes in two or more interacting species [29]. In traditional EA, the individuals evolve while interacting with each other within a species to which they belong, not considering adaptation between interacting species. It is reported that the coevolutionary algorithms offer a prospective alternative to the standard evolutionary algorithms for problems that can be decomposed into subtasks [40]. Although there are many variants of coevolutionary algorithms, they are typically classified into two main forms: cooperative coevolutionary algorithm and competitive coevolutionary algorithm, which imitate symbiosis and parasitism, respectively.

Cooperative coevolutionary algorithm [26, 28, 40], also called symbiotic evolutionary algorithms, is based on positive fitness interactions between individuals of different populations. In the algorithm, a success on one individual improves the chances of survival of the other. The populations may reciprocally enhance the adaptability to complex environments by the symbiotic relationships and coevolution. On the other hand, competitive coevolutionary algorithm

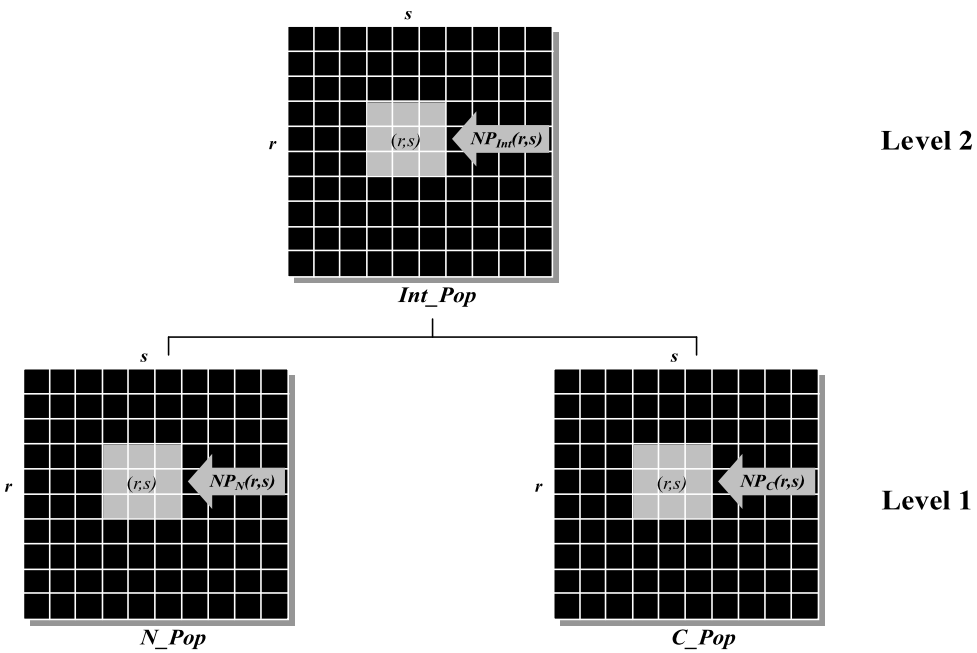


Fig. 1 The population structure of TSECA

[19, 41] is based on inverse (negative) fitness interactions between individuals of the different populations. A success on one side implies a failure of the other side to which one must respond in order to maintain one's chances of survival. Competing populations may reciprocally drive one another to increasing levels of complexity by producing an evolutionary arms race.

Based on the advantages of coevolutionary algorithm, in this research, we propose a two-leveled symbiotic evolutionary algorithm for unknown K clustering problems. The specific description of the proposed algorithm will be given in the next section.

3 Two-leveled symbiotic evolutionary clustering algorithm (TSECA)

3.1 The basic concept and structure

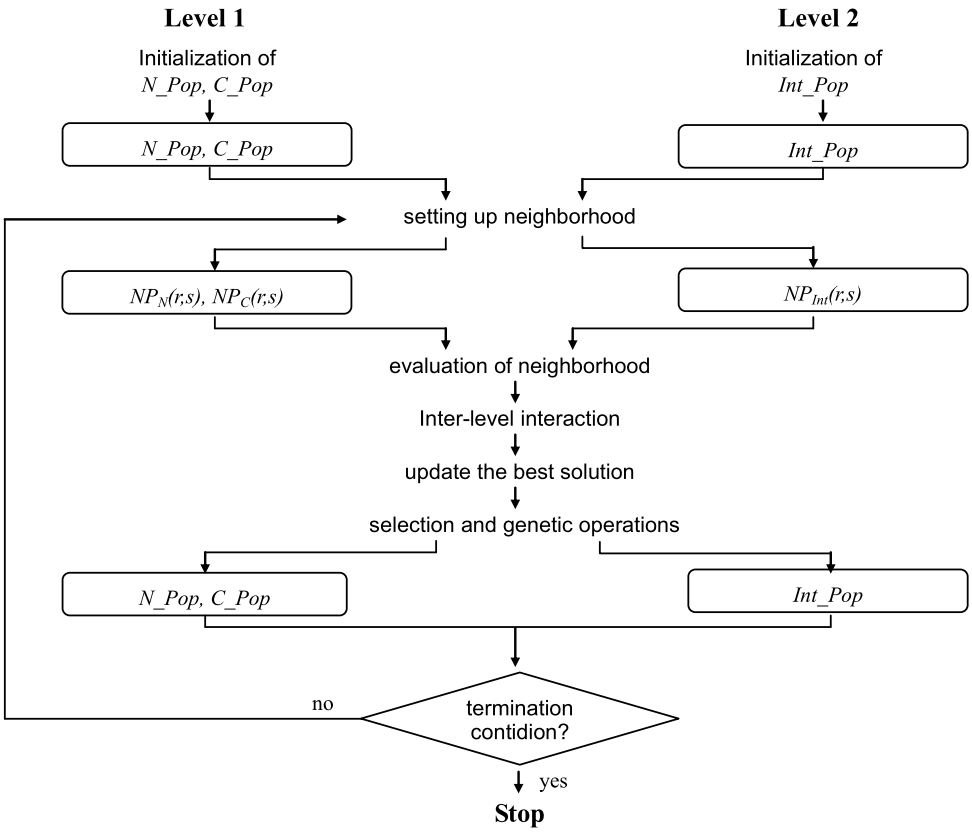
In TSECA, a two-leveled structure is maintained throughout the execution of the algorithm, as shown in Fig. 1. There are three populations in the algorithm. Each population at Level 1 and 2 forms a two-dimensional structure of square lattice. In the symbiotic evolutionary algorithm, each subproblem is treated as a distinct species, and a population is maintained for each of the species.

At Level 1, there exist two populations (*N_Pop* and *C_Pop*) that represent the two sub-problems: finding the number of clusters and grouping the data into each cluster, respectively. A chromosome in a population at Level 1 represents a partial solution to the entire problem. An entire solution is constructed by combining two partial solutions. Therefore, fitness evaluation of a chromosome in the populations at Level 1 requires selecting a chromosome in another population. This chromosome is called as the symbiotic partner, which represents a sub-problem solution. The chromosomes in the populations are separated, but interact with each other in other populations while evolving. This process simulates symbiosis in nature [28].

The population, *Int_Pop* at Level 2, consists of chromosomes that are combined with those in the lower level, each of which represents the complete solution to the entire problem. The chromosomes in *Int_Pop* evolve by themselves while keeping up their combined form within the population. Evaluation at Level 2, therefore, is the same as that of standard evolutionary algorithm. The good combinations of the chromosomes obtained from the interaction within Level 1 are transmitted to Level 2. The interactions within and between the levels imitate the natural process of endosymbiotic evolution. The theory of endosymbiotic evolution, first proposed by Margulis [33], explains the evolution process of eukaryotes from prokaryotes. In this theory, relatively simple structured prokaryotes enter into a larger host prokaryote. Thereafter, they live together in symbiosis and evolve to a eukaryote [25, 27].

To maintain the diversity of solutions in a population, a neighborhood-based evolution strategy is adopted. Unlike a standard EA, which evolves with whole population, neighborhood-based strategy evolves with parts of population. It has been reported that the strategy promotes diverse and good chromosomes to form niches, so that it can lessen premature convergences, facilitating an efficient exploration of the solution space [26, 28]. The structure and size of neighborhood can be defined in many different ways. We use the 3×3 structure here. The neighborhood at po-

Fig. 2 The outline of TSECA



sition (r, s), $NP(r, s)$ denotes the chromosome at (r, s) and its eight neighbors in a population, as shown in Fig. 1. We will present in Section 3.2 the detailed explanation of a neighborhood-based evolution strategy.

In addition, the evolution process for a population is based on a type of steady-state genetic algorithm in which one chromosome produced from its parents is updated in one evolving cycle [45, 50]. The algorithm is known to provide better performance in solution quality and computational time than a generational standard EA.

3.2 The procedure of TSECA

In this subsection, we will present the detailed procedure of the proposed algorithm, TSECA. During one evolution cycle, each population evolves independently, except for the evaluation of two populations at Level 1. Note that nine members of each population, not all members, take part in the evolving process. This evolving process is repeated until the termination condition is met. Figure 2 shows the flowchart of the proposed algorithm, and the overall procedure is described below.

Step 1: *Initialization and evaluation of initial populations*

Step 1.1: *Initialize all populations, N_Pop and C_Pop in Level 1 and Int_Pop in Level 2.*

Step 1.2: *Evaluate the initial fitness of all chromosomes in N_Pop and C_Pop. The fitness is calculated using the combination of chromosomes in the same position in N_Pop and C_Pop. Set* Ind^* *and* f^* *to be the best combination and its fitness, respectively.*

Step 1.3: *Evaluate all chromosomes in Int_Pop at Level 2 and set* Ind_{best}, f_{best} *to be the best chromosome and its fitness, respectively. If* $f^* > f_{\text{best}}$, *then set* $f_{\text{best}} = f^*$ *and* $Ind_{\text{best}} = Ind^*$.

Step 2: *Setting up neighborhood*

Choose an arbitrary position (r, s) *and set up the neighborhoods* $NP_N(r, s), NP_C(r, s)$ *of N_Pop, C_Pop and* $NP_{\text{Int}}(r, s)$ *of Int_Pop.*

Step 3: *Fitness evaluation and inter-level interaction*

Step 3.1: *For each neighborhood* $NP_N(r, s), NP_C(r, s)$ *in Level 1,*

(a) *Evaluate the fitness of chromosomes in* $NP_N(r, s)$, $NP_C(r, s)$, *by combining a symbiotic partner selected arbitrarily from another population.*

(b) *Let* Ind^* *be the best combination chromosome. Replace the worst chromosome in* $NP_{\text{Int}}(r, s)$ *of Int_Pop at Level 2 with* Ind^*.

Step 3.2: *Evaluate the fitness of chromosome in* $NP_{\text{Int}}(r, s)$. *Let* Ind_{old} *and* f_{old} *be the best chromosome in* $NP_{\text{Int}}(r, s)$.

If $f_{\text{old}} > f_{\text{best}}$, *then update* Ind_{best} *and* f_{best}, *that is,* $f_{\text{best}} = f_{\text{old}}$ *and* $Ind_{\text{best}} = Ind_{\text{old}}$.

Step 4: *Evolution of neighborhoods*

Step 4.1: *For each* $NP_N(r, s)$, $NP_C(r, s)$, *and* $NP_{\text{Int}}(r, s)$,

(a) *Produce two offspring by applying a crossover operation to two parent chromosomes which are selected from the neighborhood based on their fitness.*
(b) *Replace two chromosomes having the worst fitness from the neighborhood with the two offspring produced in* (a).
(c) *Mutate chromosomes in the neighborhood based on the mutation rate.*

Step 5: *Termination condition*
If the termination condition is met, then stop the algorithm. Otherwise, go to Step 2.

The initialization in *Step 1* randomly generates chromosomes of all populations in each level and computes their initial fitness. More specific description of the creation and fitness calculation of initial chromosomes will be given in the next section.

Step 2 sets up the neighborhood, which is the basic unit of evolution operations in the algorithm. The neighborhood $NP_P(r, s)$ indicates the chromosomes including one at position (r, s) and its eight neighbors in population P.

Fitness evaluation at Level 1 needs a strategy for selecting symbiotic partners. Although many strategies are available, there is no significant difference in the performance of the algorithms [26]. Therefore, we select randomly the partners (*Step 3.1*). In *Step 3.2*, the chromosomes in two levels interact with each other. Through the harmonic evolution between standard and coevolutionary mechanism, we expect to improve the capability of searching diverse and good solutions.

Step 4 evolves the chromosomes in the neighborhood of every population that has been set up in *Step 2*. For selection mechanism, we use binary tournament selection, which is independent of fitness scaling. In a binary tournament selection process, two individuals are selected at random and their fitness is compared. The individual with better fitness is selected as a parent. The other parent is selected in the same way. Genetic operators (crossover and mutation) used here are explained in Section 4.3 in detail.

Since TSECA uses a form of steady-state genetic algorithm, it always preserves the best chromosome among those that have been found (elitism). In addition, other parameters including the size of populations, the rate of mutation, and termination condition are explained in Section 5.

4 Evolutionary components for TSECA

As mentioned earlier, TSECA imitates the process of symbiotic evolution in nature. To construct it properly, some components and evolution mechanisms are needed. These involve the representation scheme of a potential solution, fitness evaluation, and genetic operators. In this section, we will present those elements for unknown K clustering problems in detail.

4.1 Genetic representation and initial populations

To apply a symbiotic evolutionary algorithm to clustering problems, it is required to represent the solution to the subproblems and the entire problem as genetic chromosomes. It is desirable that the chromosome representation in evolutionary algorithms is natural, clear, and not redundant. There are two populations, *N_Pop* and *C_Pop* in Level 1. For *N_Pop*, a chromosome is represented by binary strings with the length of $K_{\max}$ where $K_{\max}$ is the maximum number of possible clusters. The number of 1-bit in a string means the number of clusters. On the other hand, a chromosome of *C_Pop* is represented by a sequence of real numbers denoting the cluster centers, and appropriately determines cluster centers. For a P-dimensional space, the length of the chromosome is $K_{\max} \times P$, where the first P genes represent the first cluster center with P dimensions, the next P genes represent those of the second cluster center, and so on. These representation schemes are clear and natural, therefore it is easy to decode and apply some genetic operators which are well known. An example of chromosome representation at Level 1 is shown in Fig. 3. In Level 2, a chromosome in *Int_Pop* consists of the combinations of two different chromosomes of *N_pop* and *C_Pop*.

In Level 1, decoding of a chromosome in *N_Pop* (*C_Pop*) is achieved after forming an entire chromosome by combining a chromosome in *C_Pop* (*N_Pop*). The number of 1-bit in *N_Pop* chromosome is decoded as the number of clusters, and the values in the same position as the 1-bit of *C_Pop* chromosome are decoded as the centers of the clusters. The 0-bit of *N_Pop* chromosome and the same position of *C_Pop* chromosome are discarded in the decoding process. Clustering is done by assigning each data point to the closest center obtained previously. Thus, we can determine the number of clusters and the cluster to which each data point belongs. Decoding of a chromosome in *Int_Pop* at Level 2 is also achieved in the same way as in Level 1. Each chromosome in it can be directly decoded because the chromosomes represent the entire problem. As shown in the example in Fig. 3, the number of clusters is 4 and their centers are (54.91, 15.97), (17.81, 33.26), (41.83, 70.80), and (32.29, 79.18).

The chromosomes for all initial populations are generated randomly. For each chromosome in *N_Pop*, first generate the random number k in the range of $[1, K_{\max}]$ and

(0 1 1 0 0 1 0 1 0)
(a) An example of *N_Pop* chromosome

{(50.11, 18.62) (54.91, 15.97) (17.81, 33.26) (17.29, 67.71) (29.86, 21.15) (41.83, 70.80) (8.944, 58.71) (32.29, 79.18) (63.88, 75.10)}
(b) An example of *C_Pop* chromosome

Fig. 3 The chromosome representation of the sub-problems

choose the k genes from a chromosome at random, then set the value 1 to the chosen genes, 0 to the rest genes. For *C_Pop*, the initial chromosomes are created by randomly generating $K_{\max}$ of center points which have values ranging in $[x_d^{\min}, x_d^{\max}]$, where $x_d^{\min}$ and $x_d^{\max}$ denote the minimum and maximum value of d-th attribute in data set, respectively.

4.2 Fitness evaluation

In general, the fitness of a chromosome is evaluated using the objective function of the problem. The objective of the clustering problem considered in this paper is to maximize the similarity within each cluster and the dissimilarity among clusters. Many measures to evaluate the result of clustering have been suggested [16, 42]. In this paper, the Davies-Bouldin index (*DB*) is used to evaluate the fitness of a chromosome. The *DB* is used to find clusters which are compact and well separated by minimizing the intra-cluster distance while maximizing the inter-cluster distance. This index (*DB*) for K clusters is defined as [9]:

$$DB = \frac{1}{K}\sum_{i=1}^{K} \max_{j=1,\ldots,K, j\neq i} \frac{S_i + S_j}{d(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j)} \tag{1}$$

where S_i can be computed as

$$S_i = \frac{1}{|C_i|}\sum_{\mathbf{x}\in C_i} d(\mathbf{x}, \boldsymbol{\mu}_i)$$

$$d(\mathbf{a}, \mathbf{b}) = (\mathbf{a}^T\mathbf{b})^{1/2}$$

In (1), S_i is the intra-cluster distance, which is the average distance of all data points within cluster C_i to its cluster center $\boldsymbol{\mu}_i$, and $d(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j)$ is the inter-cluster distance which is measured by the distance between the centers of two clusters, i.e. the distance between i-th cluster center $\boldsymbol{\mu}_i$ and j-th cluster center $\boldsymbol{\mu}_j$. A cluster with low scatter and high distance from other clusters has small value of *DB*. Consequently, DB index will have a small value for a good clustering. The optimal number of clusters can be estimated by minimizing DB for different values of K. To assign higher fitness value to better chromosome, we use the inverse of DB as the fitness for a chromosome. Thus, the fitness value is defined as follows.

$$\textit{fitness} = \frac{1}{\textit{DB index}} \tag{2}$$

4.3 Genetic operations

Genetic operators are generally divided into two classes: crossover and mutation [14]. It is important to design genetic operators that are able to extract good genetic information from the parents and inherit it to offspring. Since binary (*N_Pop*) and real number (*C_Pop*) representation are used in our algorithm, existing genetic operators can be applied without any modifications. For all the populations in Level 1, we apply a two-point crossover operator [35]. Figure 4 shows the process in which one child is produced by two-point crossover for each type of chromosome. At Level 2, the genetic operators used at Level 1 can be utilized in the same manner. Recall that a chromosome at Level 2 is composed of two chromosomes in Level 1.

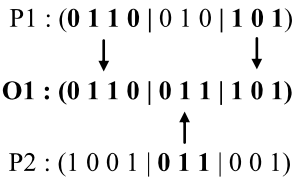
For each chromosome in *N_Pop*, the mutation is performed by selecting one or more genes with a probability equal to the mutation rate and changing the gene value 0 (1) to 1 (0). For *C_Pop* representing the center of clusters, we use the Gaussian mutation as follows [14];

$$z_{kj}^{t+1} = z_{kj}^{t} + N(0, 1) \tag{3}$$

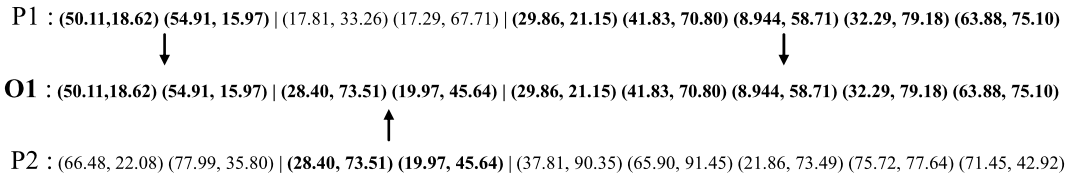
where z_{kj}^t is the j-th attribute of the k-th cluster center at generation t, $N(0, 1)$ is a Gaussian random number with a mean of zero and standard deviation of 1. For the chromosomes in *Int_Pop* at Level 2, the same operators as in Level 1 are used for each part of a chromosome.

5 Experimental results

This section presents a real-life and simulated synthetic example to evaluate the performance of the proposed algorithm. The proposed algorithm, TSECA, is compared to Genetic Clustering Unknown K-clustering (GCUK) [5]. For the validity index for clustering, we utilize Davies-Bouldin (DB) index. The algorithms were implemented in JAVA language and executed on IBM-PC with a 3.0 GHz Intel Core2 Duo CPU. The computational times are calculated using the CPU time function in JAVA. In EA-based algorithms, several control parameters must be determined. The parameters are set to those that give good results in preliminary experiments. A 10×10 grid structure is established for each population in Level 1 and 2 of TSECA, and thus the population



(a) Two-point crossover in *N_Pop*



(b) Two-point crossover in *C_Pop*

Fig. 4 An example of a crossover process

size is 100 (same to GCUK). In TSECA, the crossover rate is not required because TSECA uses the steady-state strategy in the evolution process. The crossover rate for GCUK is set to 0.7. The mutation rate for both algorithms is set to 0.05. The total number of reproduced chromosomes is used for the termination condition. When the number reaches 7,000, the algorithm is terminated.

5.1 Automatic clustering of spatial defects on wafer

Spatial defects shown on the wafer tend to clusters, which contain important information that can assist process engineers in their understanding of the ongoing manufacturing processes [23, 52]. The defects on wafers can be divided into global defects and local defects. Global defects are generated by random causes, which are difficult to remove, while local defects in clusters are produced by assignable causes, which should be removed to improve yield rates in semiconductor manufacturing. Therefore, it is crucial to automatically identify local defect clusters as well as recognize the position of clusters because different locations of clustering can lead to different root causes in manufacturing processes.

Defect recognition on wafers can be considered as a two-dimensional clustering problem. Given the coordinates of defects by an automated defect scanning tool, which scans wafer surfaces to identify the locations of defects, clustering methods take those coordinates of defects as input variables, calculate the similarity (e.g., distance) between two defects, and divide those defects into several groups. For this experiment, we generated the simulated wafers followed by the procedure presented by Yuan and Kuo [52]. We have applied the spatial filtering technique, which is popular for de-noising [49, 52], to remove the global defects from the local defects. Among total nine simulated wafers, Fig. 5 shows three representative patterns and clustering results by TSECA. The black circle, in Fig. 5, indicates the center point of each cluster.

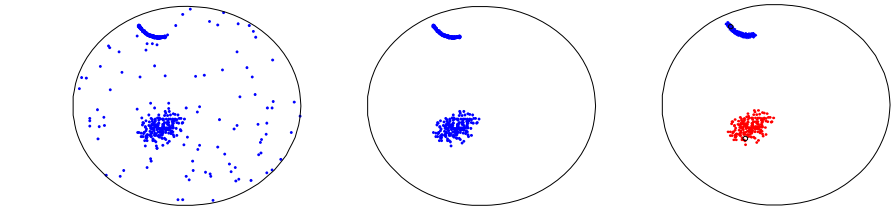
Table 1 summarizes the number of clusters, DB index, and computational time obtained by TSECA and GCUK. The values of DB index in Table 1 are the minimum and average values of 10 time runs. The result indicates that even though both TSECA and GCUK successfully find the correct number of defect clusters on wafers, average DB index of TSECA is smaller than that of GCUK, implicating that the clustering results of TSECA are more compact. In addition, p-value shows that the experimental results are statistically significant.

5.2 Simulated synthetic data set

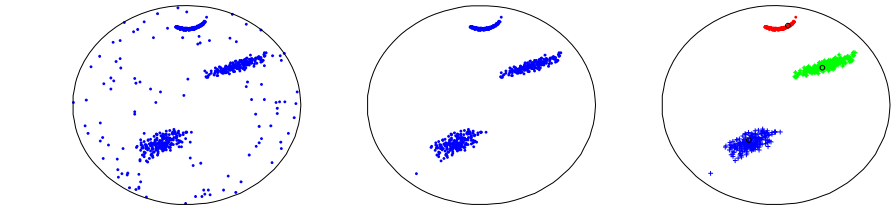
In this subsection, we explore the effectiveness of the proposed algorithm for synthetic data sets with 3 to 10 dimensions of an input vector. The i-th data set is generated by normal random number with mean μ_i and standard deviation σ_i where $0 \leq \mu_i \leq 100$ and $0 \leq \sigma_i \leq 5$. The range of the number of data points in each cluster is [50, 450]. To verify the effectiveness of the two-leveled structure of TSECA, we compare the result of TSECA with that of GCUK and Single-leveled Symbiotic Evolutionary Clustering Algorithm (SSECA), which has a single level structure with symbiotic processes (Level 1 in TSECA). The control parameters for SSECA are same as those of TSECA. Table 2 shows the performance of each algorithm with diverse dimensional data set. The values of DB index in Table 2 present the average values of 20 experiments with the same data set. In Table 2, the improved rate is calculated by

$$\frac{(\textit{average DB index}_{\text{GCUK/SSECA}} - \textit{average DB index}_{\text{TSECA}})}{\textit{average DB index}_{\text{GCUK/SSECA}}} \times 100(\%).$$

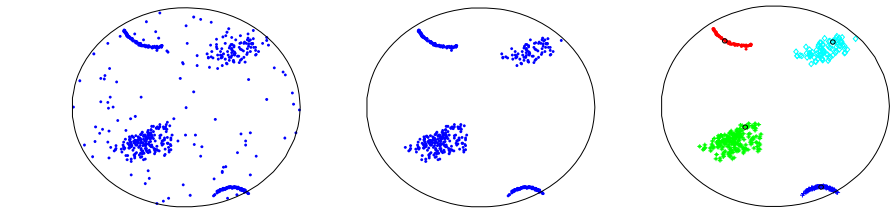
Table 2 shows that in most of cases, the proposed TSECA shows the effectiveness for clustering problems compared



(a) (i) original defect patterns, (ii) defect patterns after filtering, (iii) clustering by TSECA



(b) (i) original defect patterns, (ii) defect patterns after filtering, (iii) clustering by TSECA



(c) (i) original defect patterns, (ii) defect patterns after filtering, (iii) clustering by TSECA

Fig. 5 Performance of TSECA: (**a**) two clusters, (**b**) three clusters, and (**c**) four clusters

Table 1 Summary of the defect clustering results on wafer maps

Wafer ID	Actual no. of clusters	GCUK				TSECA				p-value
		No. of clusters*	DB index		Time (s)	No. of clusters*	DB index		Time (s)	
			Mean	STD+			Mean	STD+		
1	2	2	**0.234**	0.011	4.4	2	0.243	0.001	7.4	0.026
2	2	2	0.191	0.006	3.9	2	**0.189**	0.003	6.9	0.022
3	2	2	0.218	0.007	5.3	2	**0.216**	0.003	9.2	0.018
4	3	3	0.362	0.020	6.6	3	**0.349**	0.004	11.2	0.002
5	3	3	0.359	0.024	6.4	3	**0.326**	0.001	11.4	0.013
6	3	3	0.235	0.007	9.0	3	**0.234**	0.003	15.6	0.034
7	4	4	0.331	0.049	7.4	4	**0.281**	0.019	12.4	0.013
8	4	4	0.321	0.016	8.1	4	**0.303**	0.005	14.1	0.005
9	4	4	0.333	0.021	7.5	4	**0.283**	0.005	12.8	0.000

* The number of clusters at minimum value of DB index
+ STD: standard deviation

with standard GA-based approaches. The reason is that a symbiotic evolution at Level 1 provides a better diversity of solutions by increasing the capability of parallel search and the process at Level 2 speeds up the solution convergence by utilizing the information of entire solutions. The experimental results implicate that several parallel searches with a pieces of sub-problems are more efficient than single search for the entire problem. In Table 2, p-value shows that the proposed TSECA is a competitive approach for clustering problems with statistical significance. As an aspect of computational burden, the proposed TSECA takes more time than SSECA. This is because the TSECA is modeled with a two-leveled structure and each chromosome in Level 1 is combined with other chromosomes in other sub-populations and its fitness is then evaluated. SSECA, meanwhile, requires a little more computational time than TSECA because SSECA only conducts the process of symbiotic evolution. Note that minimizing the DB index does not always

Table 2 Performance comparison of each algorithm (standard deviation in parentheses)

Problem	DB index			Improved rate (%)		*p*-value		Computational time (s)		
(D, C, N)*	GCUK	SSECA	TSECA	vs. GCUK	vs. SSECA	vs. GCUK	vs. SSECA	GCUK	SSECA	TSECA
(3, 3, 759)	0.130 (0.005)	0.130 (0.005)	**0.120** (0.007)	7.8	7.8	0.000	0.000	9.4	27.4	20.0
(3, 5, 1242)	0.332 (0.051)	0.349 (0.038)	**0.267** (0.023)	19.5	23.4	0.000	0.000	16.8	46.3	33.3
(3, 7, 1734)	0.324 (0.056)	**0.296** (0.050)	**0.296** (0.040)	8.4	0.0	0.084	0.999	25.2	67.3	48.6
(3, 10, 2725)	0.413 (0.038)	0.427 (0.040)	**0.407** (0.038)	1.6	4.8	0.587	0.096	40.2	106.1	76.5
(5, 3, 990)	0.142 (0.006)	0.142 (0.006)	**0.136** (0.008)	4.5	4.5	0.000	0.000	20.6	60.0	45.0
(5, 5, 1369)	0.243 (0.079)	0.276 (0.074)	**0.197** (0.018)	19.1	28.5	0.019	0.000	29.0	86.3	63.9
(5, 7, 2248)	0.273 (0.078)	0.288 (0.103)	**0.203** (0.050)	25.7	29.7	0.002	0.002	51.9	140.5	104.0
(5, 10, 2197)	0.515 (0.072)	0.466 (0.095)	**0.448** (0.057)	13.0	3.9	0.002	0.471	50.4	138.2	101.7
(7, 3, 592)	0.148 (0.001)	0.165 (0.073)	**0.143** (0.006)	3.4	13.1	0.000	0.188	19.9	50.4	35.8
(7, 5, 1377)	0.267 (0.082)	0.312 (0.116)	**0.251** (0.033)	6.1	19.6	0.283	0.029	47.1	123.1	89.2
(7, 7, 1495)	0.357 (0.079)	0.350 (0.089)	**0.306** (0.045)	14.3	12.6	0.017	0.058	50.0	132.8	97.5
(7, 10, 1991)	0.417 (0.071)	0.459 (0.063)	**0.390** (0.060)	6.3	14.9	0.213	0.001	64.1	177.8	129.2
(10, 3, 1016)	0.194 (0.006)	0.194 (0.006)	**0.184** (0.009)	5.1	5.1	0.000	0.000	49.6	121.1	90.0
(10, 5, 1047)	0.233 (0.081)	0.324 (0.126)	**0.213** (0.028)	8.8	34.3	0.300	0.001	47.7	133.1	106.2
(10, 7, 1581)	0.372 (0.122)	0.391 (0.131)	**0.312** (0.065)	16.1	20.4	0.063	0.022	74.0	200.4	148.4
(10, 10, 2894)	0.488 (0.093)	0.460 (0.089)	0.403 (0.113)	17.4	12.2	0.014	0.089	136.9	360.5	273.1

* D: number of dimension, C: number of cluster, N: total number of data

guarantee the correct number of clusters and the same number of clusters on each run because EA-based algorithms are a kind of the stochastic search method, which takes their inspiration from natural selection and survival of the fittest.

Even though we used only datasets that have 3 to 10 dimensions of an input vector, similar conclusion can be made for the high dimensional datasets because the higher dimension of data does not influence on the effectiveness of chromosome encoding/decoding method in EA algorithms, only increasing computational times. In summary, based on the two experimental results, it is concluded that the proposed TSECA algorithm is a promising alternative for clustering problems with unknown cluster numbers.

6 Conclusions

This paper proposes a two-leveled symbiotic evolutionary clustering algorithm (TSECA) for unknown K clustering problems. Unlike conventional evolutionary algorithms, the proposed algorithm provides parallel exploring optimal solutions for two sub-problems such as finding the number of clusters and grouping the data into these clusters. In addi-

tion, the framework of coevolutionary algorithm and genetic elements suitable for each sub-problem are proposed. Experimental results with a real-life application and simulated synthetic data set demonstrate that TSECA produces more compact clusters as well as the accurate number of clusters. This result implicates that the proposed algorithm is a very promising alternative for unknown K clustering problems.

As for further researches, even though coevolutionary algorithm shows better performance than conventional evolutionary algorithms, much computational time is a drawback of coevolutionary-based algorithms. Therefore, we need to develop an efficient genetic representation scheme and fitness evaluation in order to reduce a computational time. In addition, the clustering problem in the paper can be viewed as a multi-objective problem. For the clustering objectives, there could be inter-cluster distance, intra-clustering distance, and the number of clusters, and so on. Evolutionary algorithms (EAs) are known to be promising for finding diverse solutions close to the true Pareto optimal solutions in multiple objective optimization problems since EAs can search for many solutions in parallel by virtue of maintaining a population of solutions. This area is another branch of EA, called Multi-Objective Evolutionary Algorithm (MOEA). There have been a huge number of studies on MOEA and its applications so far. Therefore, applying MOEA and the coevolutionary concept to the clustering problem can be an interesting future research area.

Acknowledgements This work was supported by the Korea Research Foundation Grant funded by the Korean Government [KRF-2008-357-D00288]. The part of this work was supported by the National Science Foundation (NSF) grant number CMMI-0853894.

References

1. Abraham A, Jain L, Goldberg R (2005) Evolutionary multiobjective optimization. Springer, Berlin
2. Al-Harbi SH, Rayward-Smith VJ (2006) Adaptive k-means for supervised clustering. Artif Intell 24:219–226
3. Al-Sultan K (1995) A tabu search approach to the clustering problem. Pattern Recognit 28(9):1443–1451
4. Bäck T, Hammel U, Schwefel HP (1997) Evolutionary computation: comments on the history and current state. IEEE Trans Evol Comput 1(1):3–17
5. Bandyopadhyay S, Maulik U (2002a) An evolutionary technique based on K-means algorithm for optimal clustering in $\mathbb{R}^N$. Inf Sci 146:221–237
6. Bandyopadhyay S, Maulik U (2002b) Genetic clustering for automatic evolution of clusters and application to image classification. Pattern Recognit 35:1197–1208
7. Brown D, Huntley C (1992) A practical application of simulated annealing to clustering. Pattern Recognit 25(4):401–412
8. Cowgill MC, Harvey RJ, Watson LT (1999) A genetic algorithm approach to cluster analysis. Comput Math Appl 37:99–108
9. Davies DL, Bouldin DW (1979) A cluster separation measure. IEEE Trans Pattern Recognit Mach Intell 1(2):224–227
10. Everitt B, Landau S, Leese M (2001) Cluster analysis. Arnold, Sevenoaks
11. Fogel DB (2006) Evolutionary computation: toward a new philosophy of machine intelligence. IEEE Press, New York
12. Garai G, Chaudhuri BB (2004) A novel genetic algorithm for automatic clustering. Pattern Recognit Lett 25:173–187
13. Gen M, Cheng R (1997) Genetic algorithms and engineering design. Wiley, New York
14. Gen M, Cheng R (2000) Genetic algorithm and engineering optimization, Wiley series in engineering design and automation. Wiley, New York
15. Goldberg DE (1989) Genetic algorithms in search, optimization, and machine learning. Addison Wesley, Reading
16. Halkidi M, Batistakis Y, Vazirgiannis M (2001) On clustering validation techniques. J Intell Inform Syst 17:107–145
17. Geva A (1999) Hierarchical unsupervised fuzzy clustering. IEEE Trans Fuzzy Set 7(6):723–733
18. Handl J, Meyer B (2007) Ant-based and swarm-based clustering. Swarm Intell 1(2):95–113
19. Hillis WD (1990) Co-evolving parasites improve simulated evolution as an optimization procedure. Physica D 42:228–234
20. Hruschka ER, Campello RGB, Freitas AA, Carvalho APL (2009) A survey of evolutionary algorithms for clustering. IEEE Trans Syst Man Cybern Part C 39(2):133–155
21. Hruschka ER, Ebecken NFF (2003) A genetic algorithm for cluster analysis. Intell Data Anal 7:15–25
22. Jain A, Murty M, Flynn P (1999) Data clustering: a review. ACM Comput Surv 31(3):264–323
23. Jeong YS, Kim SJ, Jeong MK (2008) Automatic identification of defect patterns in semiconductor wafer maps using spatial correlogram and dynamic time warping. IEEE Trans Semicond Manuf 21(4):625–637
24. Johnson RA, Wichern DW (2002) Applied multivariate statistical analysis. Prentice-Hall, New York
25. Kim JY, Kim Y, Kim YK (2001) An endosymbiotic evolutionary algorithm for optimization. Appl Intell 15(2):117–130
26. Kim YK, Kim JY, Kim Y (2000) A coevolutionary algorithm for balancing and sequencing in mixed model assembly lines. Appl Intell 13:247–258
27. Kim YK, Kim JY, Shin KS (2007) An asymmetric multileveled symbiotic evolutionary algorithm for integrated FMS scheduling. J Intell Manuf 18:631–645
28. Kim YK, Park KT, Ko JS (2003) A symbiotic evolutionary algorithm for the integration of process planning and job shop scheduling. Comput Oper Res 30:1151–1171
29. Koza JR (1992) Genetic programming. MIT Press, Cambridge
30. Korkmaz EE (2010) Multi-objective Genetic Algorithms for grouping problems. Appl Intell 33:179–192
31. Kuncheva LI, Bezdek JC (1997) Selection of cluster prototypes from data by a genetic algorithm. In: Proceedings of 5th European congress on intelligent techniques and soft computing, pp 1683–1688
32. Lu Y, Lu S, Fotouhi F, Deng Y, Brown SJ (2004) FGKA: a fast genetic K-means clustering algorithm. In: Proceedings of ACM symposium on applied computing, pp 622–623
33. Margulis L (1970) Origin of eukaryotic cells. Yale University Press, New Haven
34. Maulik U, Bandyopadhyay S (2000) Genetic algorithm-based clustering technique. Pattern Recognit 33:1455–1465
35. Michalewics Z (1996) Genetic algorithms+data structures= evolution programs, 3rd edn. Springer, Berlin
36. Moriarty DE, Miikkulainen R (1997) Forming neural networks through efficient and adaptive coevolution. Evol Comput 5:373–399
37. Murthy CA, Chowdhury N (1996) In search of optimal clusters using genetic algorithms. Pattern Recognit Lett 17:825–832
38. Ozyer T, Zhang M, Alhajj R (2009) Parallel clustering of high dimensional data by integrating multi-objective genetic algorithm with divide and conquer. Appl Intell 31:318–331

39. Ozyer T, Zhang M, Alhajj R (2011) Integrating multi-objective genetic algorithm based clustering and data partitioning for skyline computation. Appl Intell 35:110–122
40. Potter MA (1997) The design and analysis of a computational model of cooperative coevolution. PhD dissertation, George Mason University
41. Rosin CD, Belew RK (1997) New methods for competitive coevolution. Evol Comput 5:1–29
42. Saitta S, Raphael B, Smith IFC (2008) A comprehensive validity index for clustering. Intell Data Anal 12:529–548
43. Selim S, Alsultan K (1991) A simulated annealing algorithm for the clustering problems. Pattern Recognit 24(10):1003–1008
44. Sung C, Jin H (2000) A Tabu-search-based heuristic for clustering. Pattern Recognit 33:849–858
45. Syswerda G (1991) A study of reproduction in generational and steady-state genetic algorithms. In: Foundations of genetic algorithms. Morgan Kaufmann, San Mateo, pp 94–101
46. Tou JT, Gonzalez RC (1974) Pattern recognition principles. Addison-Wesley, Reading, MA
47. Tseng LY, Yang SB (2001) A genetic approach to the automatic clustering problem. Pattern Recognit 34:415–424
48. Vesanto J, Alhoniemi E (2000) Clustering of the self-organizing map. IEEE Transactions on Neural Networks **11**(3):586-600
49. Wang CH (2008) Recognition of semiconductor defect patterns using spatial filtering and spectral clustering. Expert Syst Appl 34:1914–1923
50. Whitley D (1989) The genitor algorithm and selection pressure: why rank-based allocation of reproductive trials is best. In: Proceedings of the third international conference on genetic algorithms. Morgan Kaufmann, San Mateo, pp 116–121
51. Xu R, Wunsch D (2005) Survey of clustering algorithms. IEEE Trans Neural Netw 16(3):645–678
52. Yuan T, Kuo W (2008) Spatial defect pattern recognition on semiconductor wafers using model-based clustering and Bayesian inference. Eur J Oper Res 190:228–240