Swarm and Evolutionary Computation 50 (2019) 100485

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

journal homepage: www.elsevier.com/locate/swevo

Multi-objective artificial immune algorithm for fuzzy clustering based on multiple kernels

Ronghua Shang [a,*], Weitong Zhang [a], Feng Li [a], Licheng Jiao [a], Rustam Stolkin [b]

[a] Key Laboratory of Intelligent Perception and Image Understanding of Ministry of Education, International Research Center for Intelligent Perception and Computation, Joint International Research Laboratory of Intelligent Perception and Computation, School of Artificial Intelligence, Xidian University, Xi'an, Shaanxi Province, 710071, China

[b] Extreme Robotics Lab, University of Birmingham, UK

ARTICLE INFO

Keywords:
Fuzzy c-means (FCM)
Multiple kernel learning
Multi-objective optimization
Artificial immune algorithm

ABSTRACT

This paper presents a multi-objective artificial immune algorithm for fuzzy clustering based on multiple kernels (MAFC). MAFC extends the classical Fuzzy C-Means (FCM) algorithm and improves some of its important limitations, such as vulnerability to local optima convergence, which can lead to poor clustering quality. MAFC unifies multi-kernel learning and multi-objective optimization in a joint clustering framework, which preserves the geometric information of the dataset. The multi-kernel method maps data from the feature space to kernel space by using kernel functions. Additionally, the introduction of multi-objective optimization helps to optimize between-cluster separation and within-cluster compactness simultaneously via two different clustering validity criteria. These properties help the proposed algorithm to avoid becoming stuck at local optima. Furthermore, this paper utilizes an artificial immune algorithm to address the multi-objective clustering problem and acquire a Pareto optimal solution set. The solution set is obtained through the process of antibody population initialization, clone proliferation, non-uniform mutation and uniformity maintaining strategy, which avoids the problems of degradation and prematurity which can occur with conventional genetic algorithms. Finally, we choose the best solution from the Pareto optimal solution set. We use a semi-supervised method to achieve the final clustering results. We compare our method against state-of-the-art methods from the literature by performing experiments with both UCI datasets and face datasets. The results suggest that MAFC is significantly more efficient for clustering and has a wider scope of application.

1. Introduction

Clustering analysis is an important problem in data analysis, in which diverse data must be divided into different classes or clusters [1]. Objects in the same cluster should exhibit significant similarities, while objects in different clusters should exhibit significant dissimilarities [2]. Clustering technology is widely used in different fields, such as pattern recognition, image segmentation, text clustering, etc [2–6]. Over the past several decades, a variety of clustering algorithms have been proposed, like k-means clustering [3], hard c-means clustering (HCM) [4], fuzzy c-means (FCM) clustering [5], and others. In many real world problems, clusters may exhibit a significant degree of overlap, and an object may belong to multiple clusters simultaneously [4]. This kind of data can be handled by replacing hard clustering methods with Fuzzy clustering.

Fuzzy c-means (FCM) [5] is a typical algorithm of fuzzy clustering, which enables a single object to belong to two or more clusters by introducing the concept of membership. The FCM algorithm has been widely used and achieves good clustering results. However, K-means and FCM are sensitive to the initialization of their cluster centers. Additionally, Euclidean distance is typically used in these algorithms, which makes such algorithms only effective for spherical clusters and well-separated data [7], performing much less well for more general clusters. Kernel learning methods have been proposed to address these problems.

In recent decades, kernel learning methods have received increasing attention in the literature, due to their abilities to improve the applicability of FCM algorithms, and handle non-linear relationships between data. Camstra and Verri presented kernel-based clustering algorithm by combining k-means and kernel learning methods [8]. Experimental results on UCI data show the effectiveness of the algorithm. By assigning

* Corresponding author.
E-mail address: rhshang@mail.xidian.edu.cn (R. Shang).

https://doi.org/10.1016/j.swevo.2019.01.001
Received 1 February 2018; Received in revised form 10 December 2018; Accepted 7 January 2019
Available online 7 January 2019
2210-6502/© 2019 Elsevier B.V. All rights reserved.

each data point different weights, Dhillon and Guan proposed the weight-based k-means and spectral clustering algorithm [9]. Kim et al. compared the kernel-based K-means and kernel-based FCM algorithms and concluded that these two algorithms exhibit equal clustering performance when using a Gaussian kernel, but both of them produce better clustering results than non–kernel methods [10]. There are two main forms of kernel-based fuzzy clustering [11]: in the first form, prototypes remain in the feature space and are mapped to the kernel space by using kernel functions (KFCM-F) [12,13], while in the second form, prototypes are built in the kernel space [14]. The second form can be problematic, in that we must use the mapping from kernel space to feature space to obtain prototypes in the feature space. Zhang et al. proposed a kernel-based FCM algorithm in which prototypes resided in the feature space [13]. The aforementioned kernel-based clustering algorithms all require a kernel function, and selection of a suitable kernel function is key to improving the effectiveness of an algorithm. However, the appropriate selection of a kernel function remains an open problem [11]. Zhao et al. handled the selection of kernel functions effectively by proposing a kernel-based maximum edge clustering algorithm, which extends single kernel clustering to multiple kernel clustering [15]. Huang et al. further improved the algorithm of Zhao et al. and presented multiple kernel fuzzy clustering (MKFC), which transforms multiple hard clustering into multiple fuzzy clustering [7].

The clustering algorithms mentioned above only consider one objective function during the clustering process, which causes the clustering results to deteriorate significantly as the number of clusters increases, and can also increase the likelihood of local optima convergence, resulting in poor clustering quality. Multi-objective optimization problems take into account multiple objective functions simultaneously, and can consider overall geometry information of data distribution and improve the quality of the clustering. For this reason, a variety of multi-objective clustering algorithms have been proposed. In multi-objective clustering algorithms, frequently used validity indices are J_m [16], *XB* [17], *OS* [18], fuzzy cluster separation *S* [19], the number of clusters *C* and total within-cluster variance [20]. Handl and Knowles proposed a multi-objective evolutionary algorithm that optimizes overall cluster deviation and cluster connectedness simultaneously [21]. In Ref. [22], Demir et al. did some experiments with several multi-objective evolutionary algorithms, and determined a suitable approach for clustering Web user sessions. In Ref. [23], the conceptual advantages of the multi-objective formulation were discussed. Mukhopadhyay and Maulik utilized a multi-objective fuzzy clustering algorithm on satellite image pixel classification by minimizing J_m and *XB* [24]. In Refs. [25,26], total within-cluster variance and the number of clusters *C* are minimized to optimal clustering. Mukhopadhyay and Maulik also made use of a multi-objective fuzzy clustering on cancer classification in Ref. [27] and optimized three indices: τ [28], J_m and *XB* simultaneously. Zhu et al. combined soft subspace clustering with a multi-objective clustering algorithm and minimized J_m and *XB* simultaneously. Experimental results showed that Zhu's multi-objective algorithm significantly improved over all compared single-objective algorithms [29]. In multi-objective clustering algorithms, two indices: J_m and *XB* are usually used. The *XB* index is defined as the ratio of the sum of pixel fuzzy mean square distance to the minimum separation of clustering center. The smaller the value of *XB* index is, the better the division is. The J_m index is the sum of the global fuzzy mean square distance of pixels. It can be seen that J_m is the index to measure the global partition of pixels, while the *XB* index is the product of separability between J_m index and nearest neighbor class. Bandyopadhyay and Maulik proved that this index combination can obtain a set of mutually exclusive non-dominant antibodies [30]. Since they are simple and have strong interpretation, the work described in our paper also employs these two indices.

A multi-objective optimization problem is less tractable than a single-objective problem, because it does not have a unique determined solution [31–33]. In recent decades, people have tried to use knowledge of biology to solve multi-objective optimization problems [34–36]. Fonseca and Fleming proposed a method which utilized a genetic algorithm (GA) to solve a multi-objective optimization problem [37]. However, this conventional GA lacks elite solutions, which leads to the loss of good solutions. Deb et al. put forward NSGA-II [38], an effective solution to handle the above problems and improve the efficiency of the algorithm. Owing to the randomness of mutation and crossover in GA algorithms, it is sometimes difficult to find optimal solutions, which can increase the number of iterations of the algorithms [39]. Algorithms based on artificial immune system can be a good solution to this problem. When an antibody can better identify the invading antigen and eliminate the antigen, the antibody is selected to be de-cloned and propagated through the cloning selection process. Then, the super-mutation can produce better affinity through the change of antibody itself to cater to the change of antigen in the body, so that the antibody can produce more mature population effectively. Finally, some antibodies with better affinity will be retained to prevent the same antigen from invading again. In the multi-objective immune algorithm, the optimization problem and the corresponding constraints can be regarded as antigens, while a possible solution vector can be regarded as an antibody. The process of processing information can increase the rate of convergence and preserve the diversity of antibody populations. Numerical optimization is one of the early applications of artificial immune system, which includes single objective optimization and multi-objective optimization. Yoo and Hajela first introduced some concepts of immunity to multi-objective optimization in 1999 [40]. Coello Coello and other scholars have proposed a more complete immune multi-objective algorithm at the first International Conference on human immune system, held at the University of Kent in 2002 [41]. Campelo and other scholars designed a multi-target clonal selection algorithm by combining the non dominated sorting and clonal selection [42]. Shang et al. solved multi-objective problems using an artificial immune algorithm in Ref. [43]. The algorithm performs well in convergence, diversity and breadth of the solution distribution. Yang et al. made use of an artificial immune algorithm for multi-objective SAR image segmentation [44]. We also employed an artificial immune algorithm in SAR image change detection, and experimental results show that the algorithm improves the optimal solution's local search capability [39]. In general, the research and application of artificial immune multi-objective optimization need to be further studied.

- In order to widen the scope of the FCM algorithm and avoid local optima convergence, this paper proposes a multi-objective artificial immune algorithm for fuzzy clustering based on multiple kernels. The main contributions of this work are as follows: The proposed algorithm takes into account between-cluster separation and within-cluster compactness simultaneously. Thus So it can consider overall geometry information of the data distribution and which enables effective avoidance of local optima convergence. Since the original FCM algorithm only applies to spherical distribution data, we introduce a multiple kernel learning method which uses kernel function mapping to generate significant linear relationships from data which are non-linear in the input space, enabling improved classification accuracy. In our proposed algorithm, prototypes reside in the feature space. This not only avoids the mapping (ϕ^{-1}) from the kernel space to the feature space, but also assists the encoding process and reduces the time complexity of the artificial immune algorithm.
- We utilize an artificial immune algorithm to handle the multi-objective clustering problem. Since the weights of kernel functions in the MKFC algorithm can easily become trapped at local optima, we encode kernel weights and cluster centers simultaneously in the encoding process. Non-uniform mutation is used in the mutation process. This method of mutation makes relates mutation range relate with to the evolution generation. This mutation method is able to avoid the degradation phenomenon of conventional genetic algorithms. In the clone selection process, we take into account degree of

affinity and crowding of the solutions simultaneously to ensure the diversity and competitiveness of the populations.

The remainder of this paper is organized as follows. We describe the related algorithms and concepts in the second section. We introduce the multi-objective artificial immune algorithm for fuzzy clustering based on multiple kernels in detail in the third section. The experimental results and analyses are presented in the forth section. Finally we summarize the paper and provide concluding remarks in the fifth section.

2. Related algorithm and concepts

For easy understanding, in this section, we briefly review relevant knowledge of FCM and multi-objective optimization.

2.1. Fuzzy C-means

Fuzzy C-means (FCM) is a well known fuzzy clustering algorithm. Firstly, we review the process of the operation briefly. Given a dataset consisting of N samples, $X = [x_1; x_2; \ldots; x_N]$, $\boldsymbol{x}_i \in R^l$. The number of clusters is C. FCM obtains the membership $u_{ik}(1 \le i \le N, 1 \le k \le C)$ i.e. the possibility that sample $\boldsymbol{x}_i$ belongs to the k-th cluster by minimizing the objective function J_m [45]:

$$\begin{aligned} & J_m(\boldsymbol{U}, \boldsymbol{V}) = \sum_{i=1}^{N} \sum_{k=1}^{C} u_{ik}^m \left\| \boldsymbol{x}_i - \boldsymbol{v}_k \right\|^2 \\ & s.t. \qquad \sum_{k=1}^{C} u_{ik} = 1 \quad \forall\ i \\ & \qquad u_{ik} \ge 0 \quad \forall\ i,k \\ & \qquad \sum_{i=1}^{N} u_{ik} > 0 \ \forall\ k \end{aligned} \tag{1}$$

where $V = [v_1; v_2; \ldots; v_C]$, v_k denotes the k-th cluster center; $U = [u_{ik}]_{i=1\ldots N, k=1\ldots C}$ is the membership matrix; m is the fuzzification degree, which generally takes $m = 2$; $||\cdot||$ represents the Euclidean distance.

FCM obtains the final clustering results by updating the membership matrix $\boldsymbol{U}$ and cluster centers $\boldsymbol{V}$ constantly. The two updated formulas are as follows:

$$u_{ik} = \frac{1}{\sum_{k'=1}^{C} \left(\frac{||\boldsymbol{x}_i - \boldsymbol{v}_k||}{||\boldsymbol{x}_i - \boldsymbol{v}_{k'}||} \right)^{\frac{2}{m-1}}} \tag{2}$$

$$\boldsymbol{v}_k = \frac{\sum_{i=1}^{N} u_{ik}^m \boldsymbol{x}_i}{\sum_{i=1}^{N} u_{ik}^m} \tag{3}$$

At the time $||U' - U|| < \varepsilon$, i.e. the iterative update condition is not satisfied, we can get the final membership $\boldsymbol{U}$ and cluster centers $\boldsymbol{V}$.

2.2. Related concepts of multi-objective problem

To facilitate understanding of the proposed work, we first briefly review the related concepts of a multi-objective problem [46].

(1) Multi-objective optimization problem can be described as:

$$\min\ \boldsymbol{F}(x) = [f_1(x), f_2(x), \ldots, f_n(x)]^T \tag{4}$$

$$s.t. \qquad [g_1(x), g_2(x), \ldots, g_p(x)] \le 0 \tag{5}$$

$$[h_1(\boldsymbol{x}), h_2(\boldsymbol{x}), \ldots, h_q(\boldsymbol{x})] = 0 \tag{6}$$

where $\boldsymbol{F}(\boldsymbol{x}) = [f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x})]^T$ denotes n objective functions; $\boldsymbol{x} = [x_1, x_2, \ldots, x_m]^T$ is an m-dimensional decision vector, which satisfies p inequality constraints $[g_1(x), g_2(x), \ldots, g_p(x)] \le 0$ and q equality constraints $[h_1(x), h_2(x), \ldots h_q(x)] = 0$.

(2) Pareto optimal solution:

Decision vector $\boldsymbol{x}$ is called Pareto optimal solution in a multi-objective minimization problem if and only if there is no other decision vector x', which can dominate $\boldsymbol{x}$, i.e. $(\forall i \in \{1, 2, \ldots, n\}, f_i(\boldsymbol{x}') \le f_i(\boldsymbol{x}))\ \&(\exists\ j\ f_j(\boldsymbol{x}') < f_j(\boldsymbol{x}))$.

(3) Pareto optimal solution set:

All Pareto optimal solutions compose the Pareto optimal set.

Multi-objective optimization problem always exists in people's life, production and scientific research. Typical algorithms for solving multi-objective problems are as follows: (1) Multi-objective optimization algorithm based on particle swarm optimization [47–50]. The particle swarm optimization algorithm is easy to implement, and the parameters of the algorithm are simple. There is also no complex adjustment required. (2) Multi-objective optimization algorithm based on artificial immune system [51–53]. In which the non-dominated neighborhood immune algorithm (NNIA) is a very effective evolutionary multi-objective algorithm. This method has relatively large advantages in solving high-dimensional multi-objective optimization problem. (3) Multi-objective optimization algorithm based on distribution estimation algorithm [54,55]. This kind of algorithms had no traditional crossover and mutation operation but a new evolutionary model. The superior performance of the distribution estimation algorithm in solving some problems has promoted more relevant algorithms. (4) Multi-objective evolutionary algorithm based on decomposition [56,57]. A decomposition based multi objective evolutionary algorithm (MOEA/D) is used to decompose the problem of the whole Pareto Front into a certain number of single objective optimization problems. The decomposition method commonly used in mathematical programming is successfully introduced into the domain of evolutionary multi-objective, and the fitness score of the single objective optimization problem can be directly applied to the evolutionary algorithm. At present, how to introduce a new dominant mechanism and how to efficiently solve the multi-objective optimization problem remain topical and challenging problems in the field of multi-objective optimization. This paper focuses on the application of a multi-objective artificial immune algorithm to the clustering problem.

3. Multi-objective artificial immune algorithm for fuzzy clustering based on multiple kernels

In this paper, an artificial immune algorithm is utilized to solve this multi-objective clustering problem which is based on multiple kernels. Firstly, MAFC initializes the population via encoding chromosomes. In the encoding process, initial cluster centers and weights of kernel functions are encoded simultaneously, so that the weights and the cluster centers update together. After obtaining an initial antibody population, we employ a clone proliferation operator to increase the number of antibodies. Then a non-uniform mutation operator is utilized to increase the diversity of the antibody population, followed by a clone selection operator for mutated antibodies to select non-dominated antibodies. Finally the non-dominated antibodies are further updated according to a uniformity maintaining strategy, which can maintain the uniformity of the solutions and increase the diversity of antibody populations. We next explain the objective function of the multi-objective clustering problem.

3.1. Objective function of multi-objective clustering based on multiple kernels

The multi-objective clustering algorithm in this paper optimizes two different clustering validity criteria, which are $J_m(\boldsymbol{w},\boldsymbol{U},\boldsymbol{V})$ and $MKXB$. So the multi-objective clustering problem can be described as:

$$\min \quad \boldsymbol{f}(\mathbf{v}) = [f_1(\mathbf{v}), f_2(\mathbf{v})] \tag{7}$$

where,

$$f_1(\boldsymbol{v}) = J_m(w, \boldsymbol{U}, \boldsymbol{V}) = \sum_{i=1}^{N}\sum_{j=1}^{C} u_{ij}^m D(\boldsymbol{x}_i, \boldsymbol{v}_j) \tag{8}$$

$$f_2(\mathbf{v}) = MKXB = \frac{\sum_{i=1}^{N}\sum_{j=1}^{C} u_{ij}^m D(\boldsymbol{x}_i, \boldsymbol{v}_j)}{N \times dis(\boldsymbol{v})} \tag{9}$$

$$dis(\boldsymbol{v}) = \frac{1}{C}\cdot\sum_{s=1}^{C}\sum_{l=1,l\neq s}^{C} u_{sl}^m D(\boldsymbol{v}_s, \boldsymbol{v}_l) \tag{10}$$

$$D(\boldsymbol{x}_i, \boldsymbol{v}_j) = \left(\phi(\boldsymbol{x}_i) - \phi(\boldsymbol{v}_j)\right)^T\left(\phi(\boldsymbol{x}_i) - \phi(\boldsymbol{v}_j)\right) \tag{11}$$

$$D(\boldsymbol{v}_s, \boldsymbol{v}_l) = (\phi(\boldsymbol{v}_s) - \phi(\boldsymbol{v}_l))^T(\phi(\boldsymbol{v}_s) - \phi(\boldsymbol{v}_l)) \tag{12}$$

$$\phi(\boldsymbol{x}) = w_1\phi_1(\boldsymbol{x}) + w_2\phi_2(\boldsymbol{x}) + \cdots + w_M\phi_M(\boldsymbol{x}) \tag{13}$$

$$u_{ij} = \frac{1}{\sum_{j=1}^{C}\left(\frac{D(\boldsymbol{x}_i, \boldsymbol{v}_j)}{D(\boldsymbol{x}_i, \boldsymbol{v}_j)}\right)^{\frac{1}{m-1}}} \tag{14}$$

$$u_{sl} = \frac{1}{\sum_{l=1,l\neq s}^{C}\left(\frac{D(\boldsymbol{v}_s, \boldsymbol{v}_l)}{D(\boldsymbol{v}_s, \boldsymbol{v}_l)}^{\frac{1}{m-1}}\right)} \qquad s \neq l \tag{15}$$

where $v = [w, v_1, v_2, \ldots, v_C]$, $\boldsymbol{v}_k$ ($k = 1,2, \ldots, C$) is the k-th cluster center of antibody $\boldsymbol{v}$; $\boldsymbol{w} = [w_1, w_2, \ldots, w_M]$, w_t($t = 1,2, \ldots, M$) is the weight of a kernel function, satisfying $w_1 + w_2 + \cdots + w_M = 1$, $w_t \in [0,1] \quad \forall\ t \in \{1, 2, \ldots, M\}$; N is the number of data simples in a dataset; C is the number of cluster centers; u_{ij} represents the possibility that data sample $\boldsymbol{x}_i$ belongs to the j-th cluster, which can be calculated by Eq. (14); m is the fuzzification degree, which generally takes $m = 2$; $D(\cdot)$ denotes the distance between two vector; $\boldsymbol{x}_i = [x_{i1}, x_{i2}, \ldots, x_{il}]$ denotes the i-th data sample, and $\boldsymbol{x}_i \in R^l$; $dis(\boldsymbol{v})$ represents the mean distance of cluster centers corresponding to antibody $\boldsymbol{v}$; u_{sl} represents the possibility that cluster center $\boldsymbol{v}_s$ belongs to the l-th cluster center, and it can be calculated by Eq. (15); $\phi = \{\phi_1, \phi_2, \ldots, \phi_M\}$ are M kinds of kernel mappings. By mapping ϕ_t, the l-dimensional vector $\boldsymbol{x}$ in its feature space becomes an L-dimensional vector $\phi_t(\boldsymbol{x})$ in a new space. $\{K_1, K_2, \ldots, K_M\}$ are Mercer kernels corresponding to M kinds of kernel mappings respectively:

$$\kappa_t(\boldsymbol{x}_i, x_j) = \phi_t(\boldsymbol{x}_i)^T\phi_t(\boldsymbol{x}_j), t = 1, 2, \ldots, M \tag{16}$$

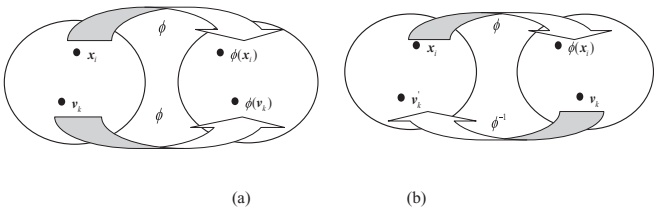
$f_1(\boldsymbol{v})$ is proposed by improving the multiple kernel fuzzy clustering algorithm (MKFC) presented by Huang et al. [7]. In MKFC:

$$J_m(w, \boldsymbol{U}, \boldsymbol{V}) = \sum_{i=1}^{N}\sum_{j=1}^{C} u_{ij}^m D(\boldsymbol{x}_i, \boldsymbol{v}_j) \tag{17}$$

where $D(\boldsymbol{x}_i, \boldsymbol{v}_j) = (\phi(\boldsymbol{x}_i) - \boldsymbol{v}_j)^T(\phi(\boldsymbol{x}_i) - \boldsymbol{v}_j)$.

This paper introduces a multiple kernel learning method, which can improve the application of the FCM algorithm and make non-linear relationships among data easier to be discovered. It can improve the clustering quality and also make performance less dependent on the particular choice of kernels. We reside prototypes in the feature space, in contrast with MKFC, where prototypes are established in the kernel space. Two different models are shown in Fig. 1 [58].

It can be seen from Fig. 1(b) that, if we want to obtain the cluster center $\boldsymbol{v}_k^{'}$ in the feature space, we need to employ the inverse mapping (ϕ^{-1}) from the kernel space to the feature space. However this operation



(a) (b)

Fig. 1. Feature space and kernel space of (a) MAFC and (b) MKFC.

is often difficult to carry out [11]. For the encoding problem in the artificial immune algorithm described in this paper, this inverse problem would undoubtedly increase the complexity of the proposed algorithm. Therefore, our algorithm uses the approach shown in Fig. 1(a) to establish prototypes so that the operations, such as antibody population initialization, are simple and have strong interpretability.

MKFC is a single-objective clustering algorithm. In the clustering process, it only takes into account the objective function J_m. Although optimizing this function can improve clustering to some extent, it is unable to take into account the overall data distribution, causing the clustering performance to significantly deteriorate with increasing numbers of clusters. To avoid this situation, on the basis of $f_1(\boldsymbol{v})$, another objective function $f_2(\boldsymbol{v})$ is introduced in this paper. $f_2(\boldsymbol{v})$ is an improved *XB* indicator. The *XB* indicator [17] is defined as:

$$XB = \frac{\sum_{i=1}^{N}\sum_{j=1}^{C} u_{ij}^m d(\boldsymbol{x}_i, \boldsymbol{v}_j)}{N \times \left(\min_{i\neq j}\{d(\boldsymbol{v}_i, \boldsymbol{v}_j)\}\right)} \tag{18}$$

where $d(x_i, v_j) = \left|\left|x_i - v_j\right|\right|^2$, $||\cdot||$ denotes Euclidean distance between two vectors.

By Eq. (18), it can be seen that *XB* indicator takes the form of a ratio. It takes into account the separation and compactness of clusters, which can effectively avoid local optima and improve the clustering performance. This paper introduces a multiple kernel learning method based on *XB*, and the Euclidean distance between any two vectors is instead represented by kernel distance, yielding the objective function $f_2(\boldsymbol{v})$. Rules related with kernel functions in $f_1(\boldsymbol{v})$ are equally applicable to $f_2(\boldsymbol{v})$.

Through the above analysis, $D(\boldsymbol{x}_i, \boldsymbol{v}_j)$ and $D(\boldsymbol{v}_i, \boldsymbol{v}_j)$ can be further denoted as:

$$\begin{aligned} D(\boldsymbol{x}_i, \boldsymbol{v}_j) &= \left(\phi(\boldsymbol{x}_i) - \phi(\boldsymbol{v}_j)\right)^T\left(\phi(\boldsymbol{x}_i) - \phi(\boldsymbol{v}_j)\right) \\ &= \phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}_i) - 2\phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{v}_j) + \phi(\boldsymbol{v}_j)^T\phi(\boldsymbol{v}_j) \\ &= \sum_{k=1}^{M} w_k^2\kappa_k(\boldsymbol{x}_i, \boldsymbol{x}_i) - 2\sum_{k=1}^{M} w_k^2\kappa_k(\boldsymbol{x}_i, \boldsymbol{v}_j) + \sum_{k=1}^{M} w_k^2\kappa_k(\boldsymbol{v}_j, \boldsymbol{v}_j) \end{aligned} \tag{19}$$

$$\begin{aligned} D(\boldsymbol{v}_i, \boldsymbol{v}_j) &= \left(\phi(\boldsymbol{v}_i) - \phi(\boldsymbol{v}_j)\right)^T\left(\phi(\boldsymbol{v}_i) - \phi(\boldsymbol{v}_j)\right) \\ &= \phi(\boldsymbol{v}_i)^T\phi(\boldsymbol{x}_i) - 2\phi(\boldsymbol{v}_i)^T\phi(\boldsymbol{v}_j) + \phi(\boldsymbol{v}_j)^T\phi(\boldsymbol{v}_j) \\ &= \sum_{k=1}^{M} w_k^2\kappa_k(\boldsymbol{v}_i, \boldsymbol{v}_i) - 2\sum_{k=1}^{M} w_k^2\kappa_k(\boldsymbol{v}_i, \boldsymbol{v}_j) + \sum_{k=1}^{M} w_k^2\kappa_k(\boldsymbol{v}_j, \boldsymbol{v}_j) \end{aligned} \tag{20}$$

In Eq. (8), $f_1(\boldsymbol{v})$ denotes the sum of overall data fuzzy mean square distances, which measures the within-cluster compactness. Therefore, a small value of $f_1(\boldsymbol{v})$ indicates a good clustering result. From Eq. (9), it can be seen that $f_2(\boldsymbol{v})$ is a ratio function, which is equal to the ratio between the sum of fuzzy mean square distances ($f_1(\boldsymbol{v})$) and the minimum square distance between cluster centers. Similarly, a smaller value of $f_2(\boldsymbol{v})$ indicates a better clustering result. Since the value of $f_2(\boldsymbol{v})$ depends on the value of $f_1(\boldsymbol{v})$ and the square distance between the nearest clusters, the between-cluster separation and within-cluster compactness must be considered simultaneously. This method significantly improves the accuracy of clustering, but it also increases the difficulty of solving the

clustering problem. The following section describes the process of solving these two objective functions.

3.2. Antibody population initialization and clone proliferation

In this paper, an antibody refers to a candidate solution of the clustering problem, and the antigen refers to the constraints and the problem, i.e. the objective function. In this artificial immune algorithm, real numbers are utilized to encode chromosomes. The size of the initial antibody population is n. Population $\boldsymbol{P}(t) = [\boldsymbol{v}^{(1)};\boldsymbol{v}^{(2)}; \ldots;\boldsymbol{v}^{(n)}]$, where t denotes the current generation; $\boldsymbol{v}^{(i)}$ represents the i-th antibody, i.e. a solution to the clustering problem. The encoding method of antibody $\boldsymbol{v}^{(i)}$ is as follows: randomly select C data samples from the dataset as initial cluster centers $\boldsymbol{V} = [\boldsymbol{v}_1;\boldsymbol{v}_2; \ldots;\boldsymbol{v}_C]$; then randomly initialize weight $w_i \in [0,1](1 \le i \le M)$ which satisfies $w_1 + w_2 + \cdots + w_M = 1$; then the antibody $\boldsymbol{v}^{(i)} = [\boldsymbol{w}^{(i)},\boldsymbol{v}_1^{(i)},\boldsymbol{v}_2^{(i)},\cdots,\boldsymbol{v}_C^{(i)}]$, where $\boldsymbol{w}^{(i)} = [w_1^{(i)},w_2^{(i)},\cdots,w_M^{(i)}]$, $\mathbf{v}_k^{(i)} = [v_{k1}^{(i)}, v_{k2}^{(i)}, \cdots, v_{kl}^{(i)}]$ denotes the k-th cluster center of antibody $\boldsymbol{v}^{(i)}$. Fig. 2 shows the encoding form of antibody $\boldsymbol{v}^{(i)}$.

Algorithm 1 describes the initialization steps of the antibody population.

Algorithm 1
The initialization of antibody population for MAFC.

```
N ← total number of data in the dataset
C ← number of clusters
n ← size of initial antibody population
l ← data dimension
data← dataset
M← the number of kernels
for(i = 1, …,n) do
  sum←0
  for (j = 1, …, M) do
    y_ij ← GenerateRandomNumber(0,1)/* Generate a random number between 0 and 1 */
    sum ← sum + y_ij
  end for
  for(j = 1, …,M)
    v^(i)(j) ← y_ij/sum/* normalize y_ij */
  end for
  for (j = 1, …,C) do
    Index ← GenerateRandomNumber(1, N)/* Generate random integer between 1 to N */
      for(k = 1 … l) do
        v^(i)(M + j*l + k) ← data_index k
      end for
    end for
end for
```

After initializing antibody population $\boldsymbol{P}(t)$, the clone proliferation operation is done to increase the size of the antibody population. Then antibody population $\boldsymbol{P}(t)$ becomes $\boldsymbol{P}^{(1)}(t)$:

$$P^{(1)}(t) = \left[v^{(1)1}; v^{(1)2}; \ldots; v^{(1)q}; v^{(2)1}; v^{(2)2}; \ldots; v^{(2)q}; \ldots; v^{(n)1}; v^{(n)2}; \ldots; v^{(n)q}\right] \quad (21)$$

where q denotes the multiples of cloning and our experiments take $q = 5$. Each antibody in $\boldsymbol{P}(t)$ has the same clonal rate in this paper.

3.3. Non-uniform mutation

After clone proliferation, we obtain antibody population $\boldsymbol{P}^{(1)}(t)$. Then each antibody in $\boldsymbol{P}^{(1)}(t)$ undergoes non-uniform mutation [43], which makes the mutation combine with evolution generation. The non-uniform mutation operator relates the mutation range relate to the evolution generation. This enables searching the space globally in the early stages and very locally at later stages [59]. This avoids the degradation phenomenon of conventional genetic algorithms. Therefore, the non-uniform mutation operator is especially useful in bounded optimization problems [60]. Mutation range is large in the early evolution of the population to ensure the algorithm searches in the global scope, and the mutation range in last stage of the evolution of the population decreases, which increases the algorithm's ability to perform local search. The mutation operator is as follows:

$$y(t,h) = h \cdot \left(1 - r^{(1-t/T)^{\sigma}}\right) \quad (22)$$

where, r is a random number, belonging to [0, 1]; t represents the current evolution generation; T denotes the maximum evolution generation, and this paper takes $T = 50$; $1 - \mathrm{t/T}$ is the mutation rate which changed with the number of iterations; σ is a parameter, which determines the extent of non-uniformity and it plays a role in adjusting the local search area, and this paper takes $\sigma = 2$. From Eq. (20), we can see that when t is small, i.e., in the early evolution of the population, the function value y is large, which ensures a large range of mutation; with the increase of evolution generation, i.e. as t increases, $y(t,h)$ decreases, and then the range of mutation narrows.

Assume $\boldsymbol{v}^{(i)} = [v^{(i)}(1), v^{(i)}(2), \ldots, v^{(i)}(m)]$ (m denotes the length of chromosome, $m = M + l \times C$) is an antibody in $\boldsymbol{P}^{(1)}(t)$. We randomly select the k-th position $\boldsymbol{v}^{(i)}(k)$ in $\boldsymbol{v}^{(i)}$ to make it mutate, then the mutated antibody becomes:

$$\boldsymbol{v}^{(i)'} = \left[v^{(i)}(1), v^{(i)}(2), \ldots, v^{(i)}(k-1), v^{(i)'}(k), \ldots, v^{(i)}(m)\right] \quad (23)$$

where $v^{(i)'}(k) = \begin{cases} v^{(i)}(k) + y(t,h) & \mathrm{kmod2} = 0 \\ v^{(i)}(k) - y(t,h') & \mathrm{kmod2} = 1 \end{cases}$**;** $h = b_k - v^{(i)}(k)$**,** $\boldsymbol{b_k}$ **is the maximum value in the** k**-th position of** $\boldsymbol{v^{(i)}}$**;** $h' = v^{(i)}(k) - a_k$**,** $\boldsymbol{a_k}$ **is the minimum value in the** k**-th position of** $\boldsymbol{v^{(i)}}$**. Data are normalized first in the algorithm, so here** $\boldsymbol{b_k = 1}$ **and** $\boldsymbol{a_k = 0}$**.** Algorithm 2 **describes the process of the non-uniform mutation.**

Algorithm 2
The non-uniform mutation of antibody population $P^{(1)}(t)$.

```
v^(i) ← an antibody in P^(1)(t)
m ← the length of chromosome
t ← the current evolution generation
T ← the maximum evolution generation
NQ ← the size of P^(1)(t)
for (i = 1, …, NQ) do
  k ← GenerateRandomNumber(1, m)/* Generate random number between 1 to m */
  r ← GenerateRandomNumber(0,1)/* Generate random number between 0 and 1*/
  if mod(k,2) = 0/*k mod 2 equals 0*/
    v^(i)'(k) ← v^(i)(k) + (1 − v^(i)(k))*(1 − r^((1−t/T)^σ))
  end if
  else if
```

(continued on next page)

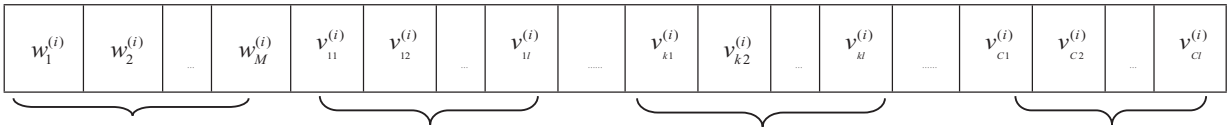


Fig. 2. The specific encoding form of antibody $v^{(i)}$.

Algorithm 2 (*continued*)

$v^{(i)'}(k) \leftarrow v^{(i)}(k) - v^{(i)}(k)*(1 - r^{(1-t/T)^r})$
end
end for

After the above non-uniform mutation, antibody population $\boldsymbol{P}^{(1)}(t)$ becomes:

$$P^{(2)}(t) = \left[v^{(1)'1}; v^{(1)'2}; \ldots; v^{(1)'q}; v^{(2)'1}; v^{(2)'2}; \ldots; v^{(2)'q}; \ldots; v^{(n)'1}; v^{(n)'2}; \ldots; v^{(n)'q}\right] \tag{24}$$

3.4. Clone selection

The purpose of clone selection is to select outstanding antibodies from the antibody population to form a new population. To avoid losing population information and ensure the diversity of population, the proposed algorithm reserves parent population $\boldsymbol{P}^{(1)}(t)$, and combines it with population $\boldsymbol{P}^{(2)}(t)$ together to form a new population $\boldsymbol{P}^{(3)}(t) = [\boldsymbol{P}^{(2)}(t); \boldsymbol{P}^{(1)}(t)]$, which is the antibody population to be selected. Here, we take the two objective functions in 3.1 as the antibody-antigen affinity function. Firstly, antibodies are divided into dominated antibodies and non-dominated antibodies according to the value of antibody-antigen affinity $f(\boldsymbol{v}) = [f_1(\boldsymbol{v}); f_2(\boldsymbol{v})]$. Affinity of a dominated antibody is set to 1, and a non-dominated antibody's affinity is 0. During the cloning process only the non-dominated antibodies are cloned, thus this ensures the competitiveness and diversity of antibodies. Assume $\boldsymbol{v}^{(j)}$ is a non-dominated antibody, which must satisfy:

$$\forall \boldsymbol{v}^{(i)} \neq \boldsymbol{v}^{(j)} \in \boldsymbol{P}^{(3)}(t):$$

$$\left(\left(f_1\left(\boldsymbol{v}^{(j)}\right) \leq f_1\left(\boldsymbol{v}^{(i)}\right)\right) \& \ \left(\exists\ i\ satisfies\ f_2\left(\boldsymbol{v}^{(j)}\right) < f_2\left(\boldsymbol{v}^{(i)}\right)\right)\right) or \left(\left(f_2\left(\boldsymbol{v}^{(j)}\right) \leq f_2\left(\boldsymbol{v}^{(i)}\right)\right) \ \& \ \left(\exists\ i\ satisfies\ f_1\left(\boldsymbol{v}^{(j)}\right) < f_1\left(\boldsymbol{v}^{(i)}\right)\right)\right)$$

After clone selection, we obtain non-dominated antibody population $P^{(4)}(t) = [v^{(1)''}; v^{(2)''}; \ldots; v^{(N)''}]$ utilized to solve the multi-objective clustering problem based on multiple kernels, where N denotes the size of non-dominated antibody population. Algorithm 3 describes the process of clone selection.

Algorithm 3
Clone selection for MAFC.

$\boldsymbol{P}^{(3)}(t) \leftarrow$ antibody population to be selected
for each $\boldsymbol{v}^{(j)} \in \boldsymbol{P}^{(3)}(t)$
  Aff($\boldsymbol{v}^{(j)}$) $\leftarrow$ 1/* set affinity of each antibody to 1 */
  for each $\boldsymbol{v}^{(i)} \in P^{(3)}(t)$ $(i \neq j)$
    if $((f_1(\boldsymbol{v}^{(j)}) \leq f_1(\boldsymbol{v}^{(i)})) \& \ (\exists\ i\ satisfies\ f_2(\boldsymbol{v}^{(j)}) < f_2(\boldsymbol{v}^{(i)}))) or ((f_2(\boldsymbol{v}^{(j)}) \leq f_2(\boldsymbol{v}^{(i)})) \ \& \ (\exists\ i\ satisfies\ f_1(\boldsymbol{v}^{(j)}) < f_1(\boldsymbol{v}^{(i)})))$
      Aff($\boldsymbol{v}^{(j)}$) $\leftarrow$ 0/* set affinity of each non-dominated antibody to 0 */
    end if
  end for
  if Aff($\boldsymbol{v}^{(j)}$) == 0 *then*/* affinity of antibody is 0 constantly */
    $\boldsymbol{P}^{(4)}(t) \leftarrow \boldsymbol{v}^{(j)}$/* select non-dominated antibody $\boldsymbol{v}^{(j)}$ to compose antibody population $\boldsymbol{P}^{(4)}(t)$*/
  end if
end for

3.5. Uniformity maintaining strategy

After obtaining non-dominated antibody population $\boldsymbol{P}^{(4)}(t)$, in order to ensure uniformity of the solutions and diversity of the population, we propose the incorporation of a uniformity maintenance strategy. If the size of the non-dominated antibody population $\boldsymbol{P}^{(4)}(t)$ is N, which satisfies $N > N^*$(N^* is the maximum size of non-dominated antibody population), then non-dominated antibody population $\boldsymbol{P}^{(4)}(t)$ undergoes a congestion degree analysis. The most crowded antibodies are removed until the population size satisfies $N = N^*$. Otherwise, non-dominated antibody population $\boldsymbol{P}^{(4)}(t)$ obtained from Algorithm 3 is regarded as the new population to be cloned directly. Algorithm 4 describes the steps to maintain uniformity of the population. Unlike NSGA-II which prefers the point with the lower non-domination rank between two solutions with differing non-domination ranks, we remove the most crowded antibody in the uniformity maintaining strategy.

After performing the uniformity maintenance strategy, we obtain the final antibody population $P^{(5)}(t) = [v^{(1)'''}; v^{(2)'''}; \ldots; v^{(N^*)'''}]$ used for cloning for solving the multi-objective clustering problem.

Algorithm 4
Uniformity maintaining strategy for MAFC.

$\boldsymbol{P}^{(4)}(t) \leftarrow$ non-dominant antibody population
$\boldsymbol{f} \leftarrow$ the value of objective functions of non-dominant antibodies in $\boldsymbol{P}^{(4)}(t)$
$N \leftarrow$ the number of non-dominated antibodies in $\boldsymbol{P}^{(4)}(t)$
$N^* \leftarrow$ the maximum number of non-dominated population
While $(N > N^*)$
  for each $v^{(i)} \in P^{(4)}(t)$
    $d_i \leftarrow 0$/* initialize the crowded distance */
  end
  for $(j = 1,2)$ *do*
    $\boldsymbol{F} \leftarrow sort(\boldsymbol{f},j)$/*sort $\boldsymbol{f}$ according to the value of the j-th objective function*/
    $\boldsymbol{P}^{(4)}(t) \leftarrow sort(\boldsymbol{P}^{(4)}(t),\boldsymbol{F}_j)$/* sort population $\boldsymbol{P}^{(4)}(t)$ according to the value of the j-th objective function */
    $d_1 = d_N = \infty$
    for $(i = 2, \ldots, N\text{-}1)$ *do*
      $d_i = d_i + (F_j(\boldsymbol{v}^{(i+1)}) - F_j(\boldsymbol{v}^{(i-1)}))/(F_j^{\max} - F_j^{\min})$
    end for
  end for
  $i \leftarrow \min(\boldsymbol{d})$/* calculate the index of the minimum value of $\boldsymbol{d}$*/
  delete $\boldsymbol{v}^{(i)}$/* remove the most crowded antibody */
  $N \leftarrow N\text{-}1$
end while

3.6. Select the final solution

Through the above artificial immune algorithm, we obtain a Pareto optimal solution set in the final generation, where each antibody represents one possible clustering result. It is difficult to say whether one solution is better than another, because of the multiple competing objective functions.

The proposed algorithm is applicable to clustering problems with datasets for which we know the true class labels for part of the data samples. Therefore, we employ a semi-supervised method to select the final solution. We select the final solution based on clustering accuracy rate of those data samples for which we know the true class labels a priori. Clustering accuracy (ACC) is defined as follows [61]:

$$ACC = \frac{\sum_{i=1}^{n} \delta(g_i, map(p_i))}{n} \tag{25}$$

where g_i denotes the true class label of data sample $\boldsymbol{x}_i$; p_i is the obtained cluster index of data sample $\boldsymbol{x}_i$; n is the number of data samples; $map(\cdot)$ denotes the matching function, which matches the true class label and

obtained cluster index; $\delta(\cdot,\cdot)$ is a Boolean function, if $x_1 = x_2$, $\delta(x_1,x_2) = 1$, otherwise, $\delta(x_1,x_2) = 0$; The larger the *ACC* value, the better the clustering results. The solution that has the largest *ACC* in Pareto optimal solution set is selected as the optimal solution in our method.

Above all, the flowchart for MAFC algorithm as shown in Fig. 3:

3.7. Complexity analysis

In this section, we analyze the time complexity of the proposed and other algorithms.

The time complexity of MAFC algorithm is mainly composed of three parts: The computational complexity of the inconsistency mutation operation; the computational complexity of clone selection operation; and the computational complexity of *non-uniform mutation*. Given: a size of the antibody group n; the dimension of the target vector 2; the size of the non-dominant antibody group N; the expected size of the non-dominant antibody group N_a; the clone ratio q: the time complexity of initializing the antibody group is $o(n)$; the time complexity of calculating the objective function value of each antibody in the antibody group is $o(2n)$; the time complexity of clone operation is $o(Nq)$; the time complexity of nonuniform mutation operation is $o(Nq)$; the time complexity of clone selection operation is $o(Nq)$; and the time complexity of antibody group updating operation is $o(N_aN^2q)$. In each iteration, the total worst time complexity is: $o(3n+3Nq + N_aN^2q)$.

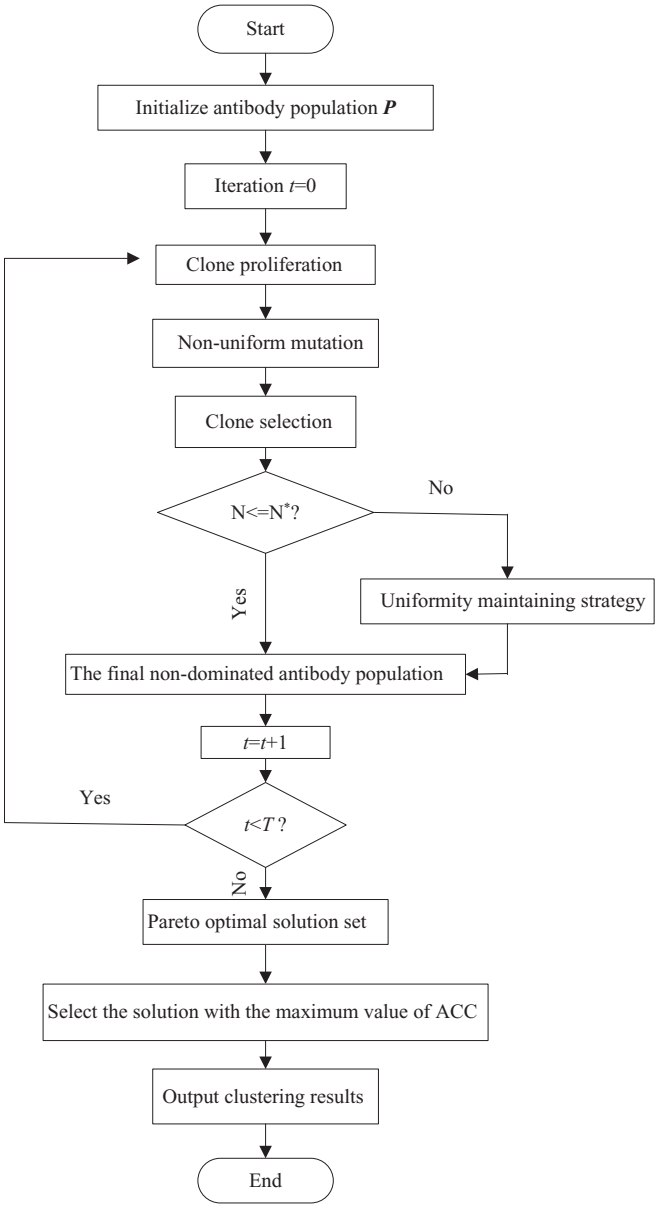


Fig. 3. Flowchart for MAFC algorithm.

Given the number of clusters C, a set of data $\mathbf{X}$ containing N l-dimensional vectors and a set of M such mappings that map features of the data to new feature spaces, the time complexity of MKFC algorithm is $o(N^2CM)$ per iteration. Given the total number of data points N, the number of clusters C and the number of dimensions of the feature vector P: since kernel-based clustering algorithms need to evaluate the kernel values between all possible pairs of two data points, the computational complexity of KFCM-F algorithm is $o(N^2CP)$ per iteration. Given the function number M, and a particle swarm size N, the time complexity of the FMOPSO algorithm is $o((MN)^2\log(N))$ per iteration. Given the total number of elements n, the number of cluster centers C, the number of objective functions 2, the number of individuals in a population N, the time complexity of the CGA algorithm is $o(nC + N^2+2N^3)$.

It can be seen that the complexity of MAFC algorithm is only related to antibody group size and clone ratio. The complexity of MAFC algorithm is lower than that of the other algorithms.

4. Experiments and analysis

4.1. Dataset

The datasets utilized in this paper include 20 UCI real datasets and two face datasets (*ORL* and *Yale*). Detailed information is shown in Table 1.

4.2. Evaluation measures

MAFC and the compared algorithms used here are FCM-related, so each algorithm will generate an $N \times C$ membership degree matrix $\boldsymbol{U} = [u_{ik}]_{i=1,\ \ldots,N,\ k=1,\ \ldots,C}$, where u_{ik} indicates the possibility that data sample $\boldsymbol{x}_i$ belongs to the k-th cluster. In this paper, we make data sample

Table 1
Datasets used in the experiments.

Datasets	Instances	Features	Classes	Comment
Iris(D1)	150	4	3	
seeds(D2)	210	7	3	
Glass identification(D3)	214	9	6	
Heart(D4)	270	13	2	
Ecoli(D5)	336	7	8	
Liver(D6)	345	6	2	
Ionosphere(D7)	351	34	2	
vote(D8)	435	16	2	
Breast Cancer Wisconsin(D9)	569	30	2	
Balance Scale(D10)	625	4	3	
Optical Recognition of Handwritten Digits(2, 7)(D11)	356	64	2	Digit(2,7)
Optical Recognition of Handwritten Digits(0, 6, 8, 9)(D12)	713	64	4	Digit(0, 6, 8 and 9)
Optical Recognition of Handwritten Digits(1, 2, 7, 9)(D13)	718	64	4	Digit(1, 2, 7 and 9)
Pima Indians Diabetes(D14)	768	8	2	
Connectionist Bench(Vowel Recognition-Deterding data)(D15)	990	10	11	letter A and B
Yeast(D16)	1484	8	10	letter A, B, C, D
Letter Recognition(A, B)(D17)	1555	16	2	
Letter Recognition(A, B, C, D)(D18)	3096	16	4	
Statlog(Landsat Satellite)(D19)	2236	36	2	
wave(D20)	5000	21	3	
ORL	400	7744	40	
Yale	165	7744	15	

$\boldsymbol{x}_i$ belong to the cluster corresponding to its maximum membership degree. So we can use the hard clustering evaluation indexes to measure the performance of each clustering algorithm [7].

The normalized mutual information (*NMI*) [62], the adjusted Rand index (*ARI*) [63] and clustering accuracy (*ACC*) are employed to evaluate the clustering performance in this paper. The definition of *ACC* has been given in section 3.7. The definitions of the other two indicators are as follows.

The *NMI* is defined as follows [62]:

$$NMI = \frac{MI(\boldsymbol{G}, \boldsymbol{P})}{\max(H(\boldsymbol{G}), H(\boldsymbol{p}))} \tag{26}$$

where $\boldsymbol{G}$ denotes the real label set of a dataset; $\boldsymbol{P}$ represents the label set obtained by a clustering algorithm; MI ($\boldsymbol{G}$, $\boldsymbol{P}$) is the mutual information between $\boldsymbol{G}$ and $\boldsymbol{P}$; H ($\boldsymbol{G}$) and H ($\boldsymbol{P}$) denote the information entropy of $\boldsymbol{G}$ and $\boldsymbol{P}$ respectively.

$$H(\boldsymbol{G}) = -\sum_{i=1}^{C} \frac{n_i}{N} \log \frac{n_i}{N} \tag{27}$$

where C denotes the number of classification in $\boldsymbol{G}$; n_i is the number of samples belonging to class i in $\boldsymbol{G}$; N represents the total number of samples in a dataset.

$$H(\boldsymbol{P}) = -\sum_{j=1}^{C'} \frac{n_j}{N} \log \frac{n_j}{N} \tag{28}$$

where C' is the number of clusters in $\boldsymbol{P}$; n_j denotes the number of samples belonging to cluster j in $\boldsymbol{P}$; N represents the total number of samples in a dataset.

The *NMI* can be obtained approximately by the following equation:

$$NMI = \frac{\sum_{i=1}^{C} \sum_{j=1}^{C'} \frac{n_{ij}}{N} \log\left(\frac{N \cdot n_{ij}}{n_i \cdot n_j}\right)}{\sqrt{H(\boldsymbol{G}) H(\boldsymbol{P})}} \tag{29}$$

where, n_{ij} is the number of data samples belonging to class i in $\boldsymbol{G}$ and cluster j in $\boldsymbol{P}$; $NMI \in [0, 1]$, the larger of *NMI*, the better clustering result is.

The *ARI* is defined as [63]:

$$ARI = \frac{a - TE}{TD - TE} \tag{30}$$

where $TE = \frac{(a+b)(a+c)}{a+b+c+d}$, $TD = \frac{a+b+c+d}{2}$. The definitions of $\boldsymbol{G}$ and $\boldsymbol{P}$ are the same as for *NMI*. The definitions of a, b, c and d are shown as follows:

a is the number of pairs of data objects that belong to the same class in $\boldsymbol{G}$ and to the same cluster in $\boldsymbol{P}$;
b is the number of pairs of data objects that belong to the same class in $\boldsymbol{G}$ and to different clusters in $\boldsymbol{P}$;
c is the number of pairs of data objects that belong to different classes in $\boldsymbol{G}$ and to the same cluster in $\boldsymbol{P}$;
d is the number of pairs of data objects that belong to different classes in $\boldsymbol{G}$ and to different clusters in $\boldsymbol{P}$.
Larger *ARI* values indicate the better clustering quality of the clustering algorithm.

4.3. Parameter selection

This section describes experiments conducted to choose appropriate values for parameters that are employed in the proposed algorithm. Appropriate parameter choices are important for achieving good results. These parameter values are tested on the seeds and Glass identification datasets. There are some parameters in this paper: the cloning number q, maximum evolution generation T and antibody population N.

For the multiple of clone q, the maximum evolution generation

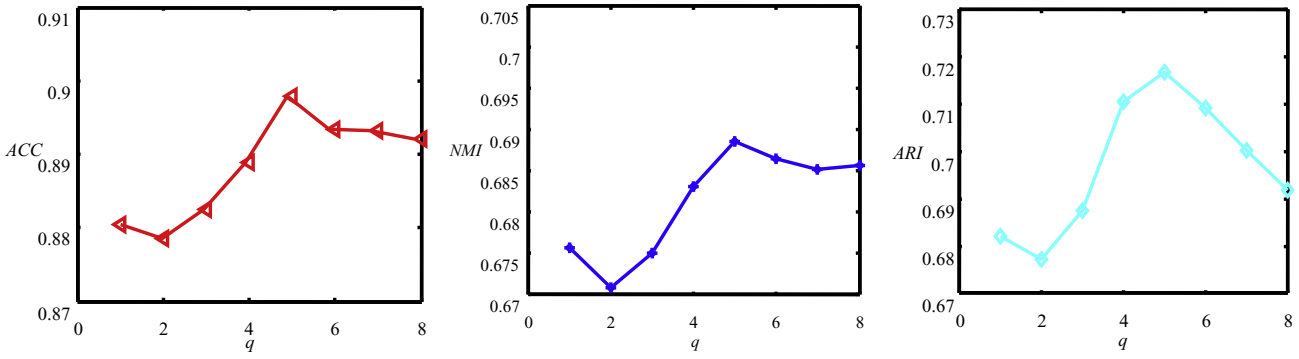


Fig. 4. The results *ACC*, *NMI* and *ARI* on the seeds dataset with different values of parameter q.

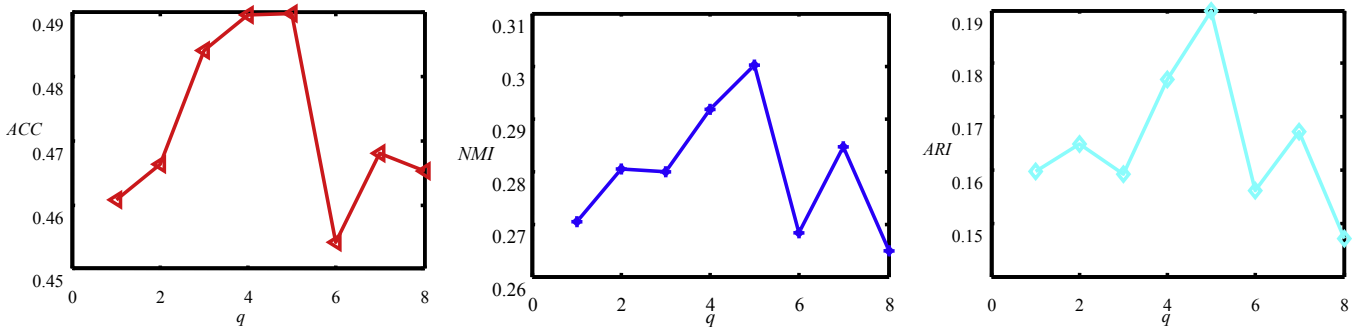


Fig. 5. The results *ACC*, *NMI* and *ARI* on the Glass identification dataset with different values of parameter q.

$T = 50$, *the* antibody population $N = 50$, the result of MAFC algorithm on the seeds dataset is shown in Fig. 4, where q is taken from 1 to 8. The result of MAFC algorithm on the Glass identification dataset is shown in Fig. 5.

It can be seen from Figs. 4 and 5 that when the maximum evolution generation and the antibody population is invariant, better clustering results can be obtained for different datasets when $N = 50$.

For the maximum evolution generation T, the multiple of clone $q = 5$, the antibody population $N = 50$, the result of MAFC algorithm on seeds dataset are shown in Fig. 6, where T ranges from 10 to 80.

The result of MAFC algorithm on Glass identification dataset are shown in Fig. 7, where T ranges from 10 to 80.

It can be seen from Figs. 6 and 7 that, with the increasing of T from 10 to 80, the clustering accuracy of this algorithm is constantly changing, but it can usually be optimized when T is 50. Although the results of *ACC* are not the maximum at T = 50 when the two networks are clustered, this paper sets T to 50 by synthesizing the other two indexes and the rationality of the parameters.

For the antibody population N, the maximum evolution generation $T = 50$, the multiple of clone $q = 5$, the result of MAFC algorithm on seeds dataset is shown in Fig. 8, where N is taken from 10 to 80.

The result of MAFC algorithm on Glass identification dataset is shown in Fig. 9, where N is taken from 10 to 80.

It can be seen from Figs. 8 and 9 that, with the increasing of N from 10 to 80. The clustering accuracy of this algorithm is constantly changing, but it can be optimized at $N = 50$. Synthesizing the situation of three indexes and its influence on the complexity of the algorithm, we set N to 50.

4.4. Data clustering

We test MAFC on 20 real datasets of UCI. Data vectors of each dataset initially undergo min-max normalization. Optimal kernel choice is still an open-research topic. Following the strategy of other multiple kernel-learning approaches, we select a set of reasonable kernels that are frequently used by kernel methods. In this experiment, in order to

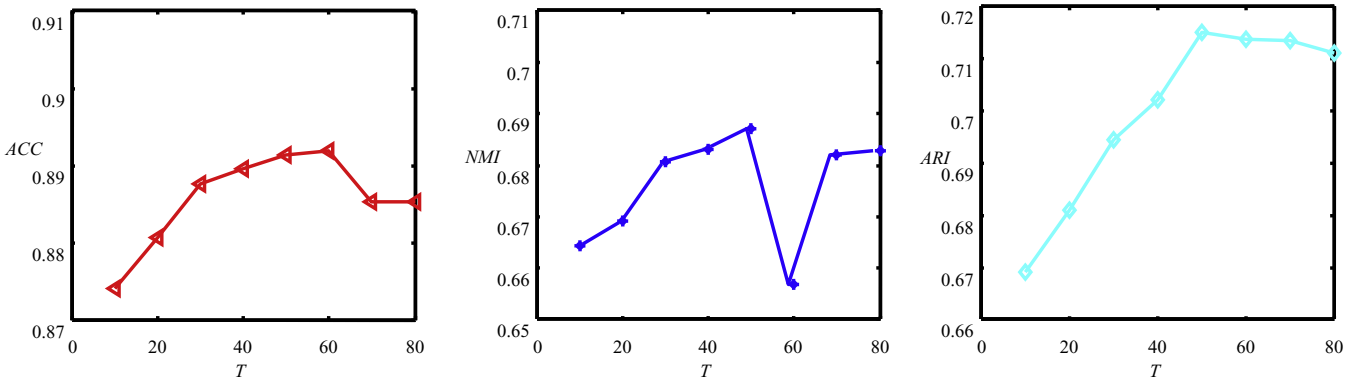


Fig. 6. The results *ACC*, *NMI* and *ARI* on the seeds dataset with different values of parameter T.

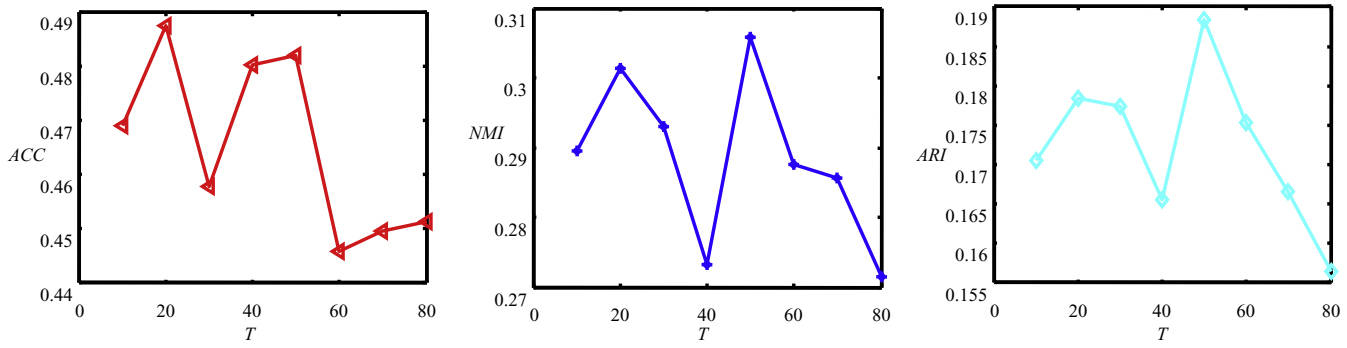


Fig. 7. The results *ACC*, *NMI* and *ARI* on the Glass identification dataset with different values of parameter T.

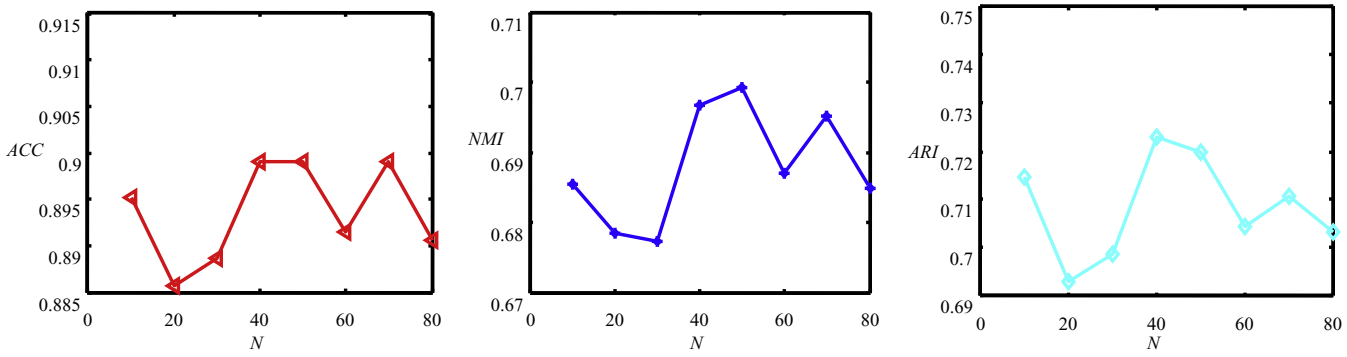


Fig. 8. The results *ACC*, *NMI* and *ARI* on the seeds dataset with different values of parameter N.

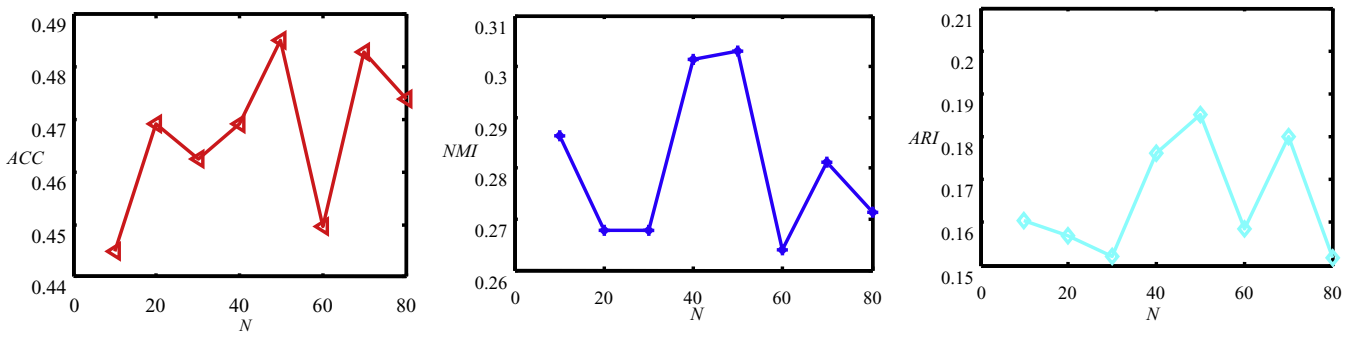


Fig. 9. The results *ACC*, *NMI* and *ARI* on the seeds dataset with different values of parameter *N*.

compare with MKFC, we choose the same kernel functions and set the same parameters. We select a polynomial kernel and seven Gaussian kernels [7]. The polynomial kernel is:

$$\kappa_k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \left(p + \boldsymbol{x}_i^T \boldsymbol{x}_j\right)^q \quad p = 1, q = 2 \tag{31}$$

Gaussian kernel is:

$$\kappa_k(\boldsymbol{y}_i, \boldsymbol{y}_j) = \exp\left(-\left(\boldsymbol{y}_i - \boldsymbol{y}_j\right)^T \left(\boldsymbol{y}_i - \boldsymbol{y}_j\right)/\delta\right) \tag{32}$$

Assume that η is the minimum value of the Gaussian kernel over the dataset. Then we can obtain the corresponding δ as:

$$\delta = \min_{i,j}\left(-\left(\boldsymbol{y}_i - \boldsymbol{y}_j\right)^T \left(\boldsymbol{y}_i - \boldsymbol{y}_j\right)/\log(\eta)\right) \tag{33}$$

We let η take each value of {0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001} to obtain seven Gaussian kernels. And the value of each kernel function is normalized to the range of [0.0001, 1]. For KFC algorithm, we utilize a polynomial kernel and seven Gaussian kernels as the kernel function respectively.

In this experiment, we compare MAFC against FCM [5], KFCM-F [13], CGA [64] and MKFC [7]. *ACC*, *NMI*, and *ARI* are employed to measure the performances of the clustering algorithms. Initial antibody population size n is 50; Number of iterations T is 50; Maximum number of non-dominated antibody population N^* is 50; The multiple of clone $q = 5$; fuzzification degree m is 2. Results in Tables 2, 7 and 9 are the best results over 50 independent runs. And results in Table 6, 8 and 10 are the mean results over 50 runs. The best value of *ACC*, *NMI* or *ARI* of each dataset is marked in bold.

Table 2 shows the best results in terms of *ACC* of different algorithms

Table 3
The best clustering centers of D18.

Dataset	Center 1	Center 2	Center 3	Center 4
*D*18	0.000000	0.000000	0.298746	0.000000
	0.303887	0.354646	0.636364	0.454545
	0.474877	0.458518	0.733333	0.600000
	0.400000	0.500000	0.800000	0.325942
	0.883767	0.666667	0.876198	0.666667
	0.266059	0.410537	0.583333	0.333333
	0.666667	0.416667	0.416667	0.416667
	0.288268	0.636364	0.636364	0.636364
	0.272727	0.636364	0.155270	0.604761
	0.174440	0.500000	0.500000	0.500000
	0.678545	0.611290	0.377636	0.300000
	0.209962	0.459515	0.545455	0.563906
	0.550506	0.230769	0.461538	0.692308
	0.407351	0.300000	0.680852	0.159440
	0.507399	0.636364	0.923832	0.636364
	0.333333	0.250000	0.773292	0.333333

Table 4
The best clustering centers of *D*1and *D*10.

Datasets	Center 1	Center 2	Center 3
*D*1	0.186096	0.443120	0.722222
	0.624904	0.276863	0.458618
	0.077379	0.591871	0.848789
	0.041456	0.500000	0.916473
*D*10	0.991104	0.751412	0.024426
	1.000000	0.603004	0.076027
	1.000000	0.000337	0.734377
	1.000000	0.063147	0.885828

Table 2
Comparisons about the best results of different algorithms on UCI datasets in terms of *ACC*.

Datasets	FCM [5]	KFC_1	KFC_2	KFC_3	KFC_4	KFC_5	KFC_6	KFC_7	KFCM-F [13]	MKFC [7]	CGA [64]	MAFC
*D*1	0.893	0.900	0.893	0.895	0.900	0.900	0.900	0.900	0.900	0.900	0.447	**0.907**
*D*2	0.900	0.861	0.900	0.900	0.900	0.900	0.900	0.900	0.900	0.900	0.612	**0.929**
*D*3	0.467	0.491	0.355	0.477	0.472	0.472	0.467	0.467	0.477	0.500	0.234	**0.547**
*D*4	0.800	0.819	0.804	0.804	0.804	0.811	0.793	0.807	0.804	0.815	0.704	**0.837**
*D*5	0.569	0.723	0.626	0.664	0.726	0.687	0.690	0.667	0.687	0.735	0.542	**0.839**
*D*6	0.554	0.559	0.519	0.512	0.528	0.528	0.516	0.528	0.525	0.522	0.261	**0.586**
*D*7	0.709	0.724	0.712	0.709	**0.732**	0.692	0.729	0.718	0.675	**0.732**	0.587	0.724
*D*8	0.874	0.871	0.864	0.867	0.880	0.871	0.876	0.878	0.875	0.878	0.632	**0.887**
*D*9	0.917	0.931	0.930	0.917	0.921	0.909	0.893	0.879	0.849	0.927	0.634	**0.956**
*D*10	0.698	0.548	0.555	0.523	0.734	0.558	0.530	0.534	0.558	0.704	0.533	**0.794**
*D*11	0.964	0.975	0.975	0.975	0.961	0.963	0.958	0.958	0.944	0.975	0.287	**0.983**
*D*12	0.912	0.964	0.964	**0.974**	0.924	0.917	0.937	0.811	0.802	0.964	0.508	0.889
*D*13	0.868	0.878	**0.880**	0.863	0.855	0.845	0.697	0.686	0.612	0.878	0.214	0.855
*D*14	0.659	0.679	0.651	0.667	0.667	0.659	0.667	0.667	0.663	0.667	0.537	**0.706**
*D*15	0.312	0.312	0.313	0.312	0.327	0.326	0.326	0.333	0.278	0.333	**0.358**	0.342
*D*16	0.398	0.431	0.377	0.398	0.416	0.427	0.409	0.435	0.389	0.457	0.133	**0.484**
*D*17	0.927	0.942	**0.943**	0.942	0.939	0.939	0.933	0.929	0.911	0.942	0.403	0.941
*D*18	0.596	0.612	0.607	0.612	0.628	0.629	0.668	0.654	0.633	0.653	**0.759**	0.690
*D*19	0.942	0.822	0.852	0.936	0.863	0.894	0.801	0.767	0.767	0.945	0.735	**0.969**
*D*20	0.645	0.638	0.638	0.609	0.597	0.576	0.480	0.512	0.455	0.635	0.661	**0.739**

Table 5
The best clustering centers of *D*5.

Dataset	Center 1	Center 2	Center 3	Center 4	Center 5	Center 6	Center 7	Center 8
D5	0.356214	0.404494	0.867599	0.831529	0.606742	0.719101	0.109296	0.743395
	0.421377	0.203141	0.312465	0.687584	0.447409	0.714286	0.289725	0.392857
	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.973563
	0.000000	0.000000	0.000000	0.168954	0.000000	0.000000	0.000000	0.000000
	0.477273	0.544885	0.522727	0.814964	0.684851	0.505928	0.655378	0.666320
	0.402062	0.195876	0.814433	0.515464	0.790744	0.329897	0.783505	0.546392
	0.433354	0.349690	0.753530	0.290135	0.838384	0.383838	0.828283	0.261402

Table 6
Comparisons about mean results of different algorithms on UCI datasets in terms of *ACC*.

Datasets	FCM [5]	KFC_1	KFC_2	KFC_3	KFC_4	KFC_5	KFC_6	KFC_7	KFCM-F [13]	MKFC [7]	CGA [64]	MAFC
*D*1	0.892	0.889	0.889	**0.895**	0.893	0.893	0.894	**0.895**	0.893	0.894	0.361	0.885
*D*2	0.893	0.859	**0.895**	0.892	0.894	0.894	0.893	**0.895**	0.886	0.893	0.548	**0.895**
*D*3	0.433	0.419	0.355	0.400	0.403	0.446	0.395	0.407	0.404	0.425	0.183	**0.459**
*D*4	0.767	0.781	0.728	0.725	0.731	0.722	0.704	0.675	0.695	0.736	0.586	**0.804**
*D*5	0.510	0.572	0.526	0.543	0.539	0.549	0.539	0.544	0.551	0.585	0.443	**0.696**
*D*6	0.516	0.559	0.519	0.512	0.528	0.528	0.516	0.528	0.525	0.522	0.181	**0.573**
*D*7	0.689	0.712	0.698	0.647	0.692	0.644	0.718	0.698	0.652	**0.729**	0.496	0.703
*D*8	0.857	0.837	0.860	0.861	0.863	0.847	0.845	0.864	0.846	0.858	0.492	**0.870**
*D*9	0.854	0.895	0.903	0.887	0.887	0.882	0.835	0.823	0.802	0.897	0.264	**0.914**
*D*10	0.518	0.502	0.534	0.516	0.524	0.517	0.500	0.513	0.533	0.517	0.342	**0.617**
*D*11	**0.964**	0.961	0.961	0.961	0.934	0.930	0.921	0.900	0.894	**0.964**	0.174	0.933
*D*12	0.806	0.947	0.949	**0.964**	0.908	0.900	0.889	0.734	0.736	**0.964**	0.336	0.806
*D*13	**0.868**	0.850	0.867	0.845	0.839	0.825	0.641	0.643	0.597	0.850	0.158	0.759
*D*14	0.659	**0.679**	0.651	0.667	0.667	0.659	0.667	0.667	0.663	0.667	0.369	0.670
*D*15	0.275	0.278	0.278	0.275	0.281	0.281	0.285	0.285	0.240	0.285	0.275	**0.298**
*D*16	0.338	0.329	0.312	0.324	0.333	0.338	0.338	0.327	0.328	0.339	0.099	**0.405**
*D*17	0.927	**0.939**	**0.939**	**0.939**	0.934	0.932	0.927	0.911	0.898	0.935	0.296	0.905
*D*18	0.557	0.589	0.585	0.586	0.580	0.596	0.655	0.633	0.618	0.635	0.614	**0.670**
*D*19	0.900	0.772	0.801	0.887	0.811	0.842	0.753	0.715	0.715	0.896	0.657	**0.960**
*D*20	0.515	0.570	0.567	0.572	0.530	0.515	0.455	0.423	0.416	0.537	0.573	**0.606**

Table 7
Comparisons about the best results of different algorithms on UCI datasets on *NMI*.

Datasets	FCM [5]	KFC_1	KFC_2	KFC_3	KFC_4	KFC_5	KFC_6	KFC_7	KFCM-F [13]	MKFC [7]	CGA [64]	MAFC
*D*1	0.737	0.768	0.736	0.737	0.742	0.747	0.752	0.752	0.752	0.749	**0.962**	0.803
*D*2	0.690	0.679	0.690	0.690	0.690	0.690	0.690	0.690	0.690	0.690	**0.896**	0.761
*D*3	0.353	0.345	0.335	0.360	0.353	0.364	0.363	0.364	0.348	0.366	**0.654**	0.370
*D*4	0.278	0.312	0.290	0.290	0.292	0.306	0.282	0.299	0.286	0.312	0.186	**0.400**
*D*5	0.594	0.583	0.582	0.584	0.619	0.573	0.576	0.574	0.568	0.561	0.421	**0.699**
*D*6	0.009	0.000	0.003	0.003	0.000	0.000	0.001	0.000	0.000	0.001	**0.212**	0.003
*D*7	0.126	0.123	0.121	0.155	0.223	0.215	0.253	0.244	0.223	0.223	**0.457**	0.150
*D*8	0.475	0.458	0.465	0.464	0.479	0.469	0.472	0.474	0.489	0.490	0.319	**0.503**
*D*9	0.598	0.653	0.678	0.641	0.586	0.583	0.545	0.526	0.468	0.632	0.516	**0.731**
*D*10	0.301	0.173	0.139	0.141	0.382	0.146	0.153	0.132	0.153	0.351	0.405	**0.439**
*D*11	0.806	0.830	0.830	0.830	0.815	0.807	0.765	0.759	0.687	0.830	0.126	**0.878**
*D*12	0.807	0.807	0.807	**0.830**	0.812	0.774	0.774	0.598	0.629	**0.830**	0.172	0.728
*D*13	**0.707**	0.688	0.688	0.688	0.673	0.639	0.465	0.448	0.467	0.688	0.175	0.558
*D*14	0.114	0.130	0.130	0.127	0.140	0.144	**0.146**	0.145	0.143	0.140	0.106	0.125
*D*15	0.384	0.391	0.391	0.385	0.392	0.398	**0.403**	**0.403**	0.298	**0.403**	0.362	0.357
*D*16	0.260	0.255	0.258	0.260	0.261	0.258	0.254	0.252	0.250	0.265	**0.781**	0.318
*D*17	0.707	0.734	**0.738**	0.734	0.720	0.715	0.693	0.682	0.685	0.734	0.271	0.720
*D*18	0.389	0.395	0.418	0.441	0.464	0.465	0.552	0.523	0.523	0.545	0.567	**0.605**
*D*19	0.461	0.424	0.438	0.512	0.452	0.487	0.451	0.396	0.387	0.493	0.401	**0.810**
*D*20	0.397	0.381	0.381	0.373	0.368	0.367	0.335	0.324	0.271	0.381	0.237	**0.435**

on UCI datasets, analyzing data in the table we can see: over 20 UCI datasets, MAFC produces better clustering results on 14 datasets, compared with other algorithms. And the improvement is obvious, for dataset *D*5, *D*10 and *D*20, all improvement rate of *ACC* is about 10%. In MKFC, there is only one dataset, that is, *D*7 whose *ACC* is better than that in MAFC. In eight compared algorithms using one kernel function, only four datasets whose *ACC* is higher than that in MAFC. In CGA, there is only 2 datasets, whose *ACC* is better than that in MAFC. The above fully illustrates that combining multi-objective optimization with multiple kernel is effective for data clustering and the optimization ability of MAFC for data clustering is better than that of compared algorithms.

Tables 3–5 show the best clustering centers (normalized into [0, 1]) of part of UCI datasets achieved by MAFC.

Table 6 shows the mean results on *ACC* of different algorithms on UCI datasets. We can see that in the mean results of 50 independent runs, MAFC produces better clustering results on 12 datasets compared with other algorithms. And *ACC* of *D*2 in MAFC equals to that in KFC_2 and KFC_7. All of them get the best result. In MKFC, only three datasets, that is, *D*7, *D*11, *D*12 whose *ACC* are not worse than other algorithms. The above explains that the clustering effect of MAFC has been significantly

Table 8
Comparisons about mean results of different algorithms on UCI datasets on *NMI*.

Datasets	FCM [5]	KFC_1	KFC_2	KFC_3	KFC_4	KFC_5	KFC_6	KFC_7	KFCM-F [13]	MKFC [7]	CGA [64]	MAFC
*D*1	0.735	0.775	0.731	0.737	0.735	0.741	0.746	0.747	0.749	0.737	**0.804**	0.777
*D*2	0.678	0.675	0.682	0.678	0.684	0.681	0.687	0.682	0.680	0.677	**0.768**	0.690
*D*3	0.333	0.333	0.335	0.342	0.349	0.356	0.356	0.354	0.335	0.355	**0.421**	0.303
*D*4	0.228	0.256	0.185	0.181	0.189	0.182	0.164	0.132	0.156	0.196	0.129	**0.293**
*D*5	0.574	0.574	0.573	0.574	0.556	0.563	0.564	0.561	0.556	0.543	0.317	**0.580**
*D*6	0.002	0.000	0.003	0.003	0.000	0.000	0.001	0.000	0.000	0.001	**0.153**	0.003
*D*7	0.120	0.120	0.115	0.122	0.198	0.197	0.244	0.231	0.211	0.202	**0.313**	0.104
*D*8	0.437	0.416	0.455	0.454	0.455	0.431	0.425	0.450	0.419	0.445	0.119	**0.457**
*D*9	0.571	0.584	0.584	0.578	0.546	0.527	0.501	0.474	0.409	0.578	0.343	**0.593**
*D*10	0.118	0.119	0.119	0.118	0.121	0.114	0.116	0.116	0.116	0.120	0.166	**0.241**
*D*11	0.806	0.806	0.806	0.806	0.759	0.784	0.743	0.700	0.607	0.806	0.102	0.675
*D*12	0.800	0.801	0.804	**0.816**	0.788	0.755	0.728	0.555	0.573	0.812	0.066	0.637
*D*13	**0.707**	0.686	0.683	0.683	0.653	0.616	0.425	0.417	0.403	0.686	0.063	0.528
*D*14	0.114	0.130	0.130	0.127	0.140	0.144	**0.146**	0.145	0.143	0.140	0.093	0.089
*D*15	0.369	0.371	0.371	0.370	0.373	0.372	0.377	**0.378**	0.234	**0.378**	0.322	0.306
*D*16	0.250	0.253	0.254	0.252	0.253	0.250	0.253	0.246	0.238	0.253	**0.602**	0.261
*D*17	0.707	**0.718**	0.717	0.717	0.712	0.702	0.677	0.657	0.635	0.716	0.203	0.576
*D*18	0.372	0.389	0.401	0.411	0.421	0.441	0.521	0.516	0.509	0.493	0.123	**0.562**
*D*19	0.411	0.359	0.361	0.374	0.350	0.347	0.320	0.322	0.302	0.373	0.218	**0.768**
*D*20	0.374	0.372	0.371	0.369	0.352	0.338	0.271	0.259	0.243	0.369	0.168	**0.380**

Table 9
Comparisons about the best results of different algorithms on UCI datasets on *ARI*.

Datasets	FCM [5]	KFC_1	KFC_2	KFC_3	KFC_4	KFC_5	KFC_6	KFC_7	KFCM-F [13]	MKFC [7]	CGA [64]	MAFC
*D*1	0.728	0.747	0.731	0.735	0.745	0.745	0.750	0.748	0.752	0.745	0.396	**0.759**
*D*2	0.727	0.652	0.729	0.729	0.729	0.729	0.729	0.729	0.729	0.729	0.562	**0.798**
*D*3	0.194	0.204	0.176	0.175	0.184	0.187	0.182	0.174	0.169	0.185	0.031	**0.264**
*D*4	0.357	0.405	0.369	0.369	0.369	0.387	0.342	0.308	0.369	0.396	0.166	**0.452**
*D*5	0.393	0.401	0.395	0.393	0.495	0.375	0.376	0.365	0.367	0.423	0.403	**0.768**
*D*6	0.009	0.001	−0.006	−0.006	−0.001	−0.001	−0.002	0.000	0.000	−0.001	**0.061**	0.010
*D*7	0.173	0.168	0.178	0.175	0.111	0.102	0.089	0.102	0.098	0.151	**0.193**	0.187
*D*8	0.557	0.551	0.531	0.538	0.578	0.551	0.565	0.572	0.565	0.572	0.203	**0.599**
*D*9	0.712	0.742	0.736	0.724	0.707	0.707	0.645	0.589	0.476	0.738	0.034	**0.831**
*D*10	0.351	0.179	0.180	0.155	0.455	0.175	0.169	0.162	0.175	0.384	0.341	**0.525**
*D*11	0.869	0.901	0.901	0.901	0.873	0.863	0.857	0.848	0.794	0.901	0.021	**0.934**
*D*12	0.783	0.858	0.858	**0.878**	0.824	0.838	0.757	0.703	0.668	**0.878**	0.026	0.740
*D*13	0.663	0.681	**0.708**	0.644	0.627	0.644	0.521	0.514	0.419	0.681	0.009	0.660
*D*14	0.081	0.135	0.116	0.109	0.099	0.096	0.086	0.081	0.083	0.116	0.102	**0.168**
*D*15	0.185	0.201	0.201	0.212	0.208	0.201	0.199	0.207	0.195	0.210	**0.215**	0.186
*D*16	0.150	0.151	0.135	0.156	0.169	0.164	0.145	0.151	0.142	0.169	0.033	**0.218**
*D*17	0.759	0.782	**0.786**	0.782	0.770	0.772	0.750	0.737	0.674	0.782	0.206	0.777
*D*18	0.338	0.353	0.353	0.345	0.396	0.396	0.365	0.365	0.354	0.372	0.081	**0.472**
*D*19	0.467	0.403	0.388	0.365	0.400	0.392	0.354	0.308	0.274	0.415	0.435	**0.875**
*D*20	0.334	0.304	0.304	0.284	0.281	0.266	0.294	0.235	0.189	0.304	**0.442**	0.416

Table 10
Comparisons about mean results of different algorithms on UCI datasets on *ARI*.

Datasets	FCM [5]	KFC1	KFC2	KFC3	KFC4	KFC5	KFC6	KFC7	KFCM-F [13]	MKFC [7]	CGA [64]	MAFC
*D*1	0.725	0.740	0.725	0.735	0.735	0.742	0.748	0.741	**0.749**	0.734	0.268	0.717
*D*2	0.710	0.646	0.719	0.713	0.722	0.717	**0.726**	0.719	0.713	0.713	0.503	0.705
*D*3	0.181	0.182	0.176	0.175	0.179	0.177	0.172	0.161	0.136	0.179	0.015	**0.183**
*D*4	0.294	0.335	0.238	0.230	0.237	0.229	0.200	0.156	0.186	0.250	0.137	**0.368**
*D*5	0.387	0.379	0.380	0.380	0.375	0.361	0.349	0.342	0.329	0.377	0.304	**0.550**
*D*6	−0.002	0.001	−0.006	−0.006	−0.001	−0.001	−0.002	0.000	0.000	−0.001	**0.033**	0.002
*D*7	0.141	0.133	0.124	0.108	0.098	0.074	0.056	0.033	0.010	0.123	**0.166**	0.156
*D*8	0.513	0.478	0.524	0.522	0.524	0.498	0.494	0.520	0.496	0.515	0.173	**0.531**
*D*9	0.690	**0.701**	**0.701**	**0.701**	0.695	0.672	0.621	0.573	0.379	0.695	0.004	0.686
*D*10	0.138	0.136	0.136	0.132	0.142	0.133	0.130	0.127	0.131	0.135	0.108	**0.272**
*D*11	0.869	0.869	0.869	0.869	0.859	0.859	0.848	0.838	0.787	0.869	0.012	**0.885**
*D*12	0.779	0.785	0.802	0.804	0.794	0.773	0.700	0.674	0.644	**0.824**	0.012	0.600
*D*13	**0.663**	0.652	0.662	0.639	0.619	0.583	0.507	0.480	0.262	0.651	0.008	0.522
*D*14	0.081	**0.135**	0.116	0.109	0.099	0.096	0.086	0.081	0.083	0.116	0.081	0.118
*D*15	0.166	0.173	0.174	0.175	0.174	0.174	0.175	0.175	0.171	**0.175**	0.152	0.139
*D*16	0.130	0.130	0.130	0.130	0.129	0.128	0.126	0.122	0.111	0.130	0.018	**0.143**
*D*17	0.759	**0.761**	**0.761**	**0.761**	0.755	0.752	0.726	0.693	0.643	**0.761**	0.114	0.660
*D*18	0.324	0.334	0.338	0.339	0.345	0.350	0.346	0.356	0.335	0.339	0.051	**0.452**
*D*19	0.409	0.333	0.308	0.307	0.320	0.305	0.278	0.245	0.208	0.332	0.314	**0.844**
*D*20	0.258	0.258	0.257	0.256	0.250	0.244	0.190	0.169	0.142	0.256	0.249	**0.283**

improved with respect to MKFC.

Table 7 shows the best results on *NMI* of different algorithms on UCI datasets.

From Table 7 we can see that: over 20 UCI datasets, MAFC produces better *NMI* on 9 datasets compared with other algorithms. MKFC produces better *NMI* only on two datasets, that is, *D*12 and *D*15 compared with MAFC. More importantly, the improvement of *NMI* produced by MAFC is clear. For example, the improvement rate of *NMI* on *D*4, *D*5 and *D*10 is about 10%. This fully illustrates that the introduction of multi-objective optimization based on MKFC improves optimization ability of the algorithm. Table 8 shows the mean results on *NMI* of different algorithms on UCI datasets.

From Table 8, we can obtain the following conclusions. In terms of specific datasets, for *D*4 dataset, *NMI* of FCM algorithm is 0.228, and *NMI* of KFC_1 is 0.256. This suggests that it is effective to introduce the kernel learning method for improving clustering performance. The worst *NMI* of KFC algorithms is 0.132. *NMI* of MKFC is 0.196. Although MKFC is not the highest scoring technique, it is still competitive. MKFC is robust against improper kernel function selection. Therefore, on the whole, the introduction of multiple kernels can achieve more effective and more stable clustering results than that of single kernel. *NMI* of MAFC is 0.293, which is a significant improvement compared with MKFC and CGA. This suggests that the multi-objective optimization based on MKFC is very effective for improving the clustering quality. This situation also exists in *D*9 dataset. Additionally, for dataset *D*10, *D*18 and *D*19, *NMI* has a much larger improvement, which also demonstrates the superiority of the MAFC algorithm for clustering problems. On the whole, over 20 UCI datasets, MAFC produces better clustering results on 8 datasets, compared with other algorithms. The improvements on some datasets are very significant.

In Table 9, we present the best results of different algorithms on *ARI*. From data in Table 9 we can see that in the best results of 50 independent runs, MAFC produces large *ARI* on 13 datasets over 20 UCI datasets, compared with other algorithms. MKFC produces large *ARI* only on 2 datasets compared with other algorithms. All the rest of the algorithms produce better clustering results on one dataset. More importantly, the improvement of *ARI* is significant. For example, for dataset *D*5, the *ARI* of MKFC is 0.423, the ARI of CGA is 0.403 and *ARI* of MAFC is 0.768. The improvement rate is more than 30%. For datasets *D*9 and *D*10, the improvement of *ARI* is both obvious. These results suggest that MAFC can produce better clustering results and it is more suitable for data clustering. Table 10 gives the mean results of different algorithms on *ARI*.

The results of Table 10 suggest that, although the introduction of multi-kernel learning methods alone can make up for the deficiencies in FCM to some extent, the improvement of clustering quality is not significant. As for *D*10 dataset, *ARI* of FCM is 0.138, *ARI* of MKFC is 0.135 and ARI of CGA is 0.108. The clustering quality has no significant improvement, and even a slight decline. In contrast, the *ARI* of our proposed MAFC algorithm is 0.272, which is more than twice that of the other compared algorithms. Similar results are observed for datasets *D*4 and *D*5. Especially notable is the *D*19 dataset, where the *ARI* of MKFC is only 0.332, while the *ARI* of MAFC is 0.844, which is more than 2 times larger. The above results suggest that the proposed MAFC algorithm has an obvious and significant effect on improving the clustering quality. Overall, MAFC generates better clustering results on 10 datasets compared to all other tested algorithms in terms of *ARI*.

To provide overall evaluation and summarization of these results, we now provide a statistical analysis of our experiment to compare the MAFC and FCM, KFC and MKFC algorithms. Table 11 presents the Wilcoxon signed rank test between the MAFC and the FCM algorithms. p represents the probability of the median of two samples being equal, and the null hypothesis should be questioned when p is close to 0. h is the test result where $h=0$ indicates that the difference between the median of the two samples is not significant whereas $h=1$ means the difference between the median of the two samples is significant. We can see from the Wilcoxon signed rank test between the MAFC and the FCM algorithms, there are 14 "$h=1$" in 20 test instances on *ACC*, 12 "$h=1$" in 20 test instances on *NMI* and 10 "$h=1$" in 20 test instances on *ARI*.

Table 11
The Wilcoxon signed rank test between the MAFC and the FCM algorithms.

MAFC/FCM	*ACC*		*NMI*		*ARI*	
	p	*h*	*p*	*h*	*p*	*h*
D1	0.000067	1	0	1	0.113724	0
D2	0.008336	1	0.000062	1	0.237465	0
D3	0.000123	1	0.000001	1	0.285216	0
D4	0	1	0	1	0	1
D5	0	1	0.442450	0	0	1
D6	0	1	0	1	0	1
D7	0	1	0.959893	0	0	1
D8	0.003984	1	0.087814	0	0.000923	1
D9	0	1	0.759597	0	0.008994	1
D10	0.000017	1	0	1	0.000005	1
D11	0	1	0.000002	1	0.065473	0
D12	0.029214	1	0	1	0.364756	0
D13	0.000201	1	0	1	0.529778	0
D14	0.058448	0	0.009503	1	0.042124	1
D15	0.114806	0	0.003514	1	0.886001	0
D16	0.191023	0	0.185207	0	0.178356	0
D17	0.000001	1	0.000003	1	0.728246	0
D18	0.364792	0	0.626592	0	0.042904	1
D19	0.150714	0	0.782025	0	0.001018	1
D20	0.146805	0	0.086706	0	0.283197	0

Table 12 is the Wilcoxon signed rank test between the MAFC and the KFC3 algorithms. From the Wilcoxon signed rank test between the MAFC and the KFC3 algorithms, there are 14 "$h=1$" in 20 test instances on *ACC*, 12 "$h=1$" in 20 test instances on *NMI* and 10 "$h=1$" in 20 test instances on *ARI*.

Table 13 presents the Wilcoxon signed rank test between the MAFC and the MKFC algorithms. There are 13 "$h=1$" in 20 test instances on *ACC*, 14 "$h=1$" in 20 test instances on *NMI* and 11 "$h=1$" in 20 test instances on *ARI*. These results further support the usefulness of our proposed algorithm.

In summary, the proposed MAFC algorithm shows a statistically significant improvement in clustering quality compared with other state of the art algorithms from the literature. It shows better cluster validity on 20 UCI datasets, which supports the ideas of this paper, that combining multi-kernel learning methods and multi-objective optimization methods within a unified clustering framework is a useful and powerful approach.

To further illustrate our method, we plot Pareto fronts of some datasets produced by a single run of MAFC in Figs. 10 and 11, and

Table 12
The Wilcoxon signed rank test between the MAFC and the KFC_3 algorithms.

MAFC/KFC3	*ACC*		*NMI*		*ARI*	
	p	*h*	*p*	*h*	*p*	*h*
D1	0.000010	1	0	1	0	1
D2	0.026997	1	0.000702	1	0.237465	0
D3	0	1	0	1	0.715095	0
D4	0	1	0	1	0	1
D5	0	1	0.668570	0	0	1
D6	0	1	0	1	0	1
D7	0	1	0.598596	0	0	1
D8	0.000666	1	0.386468	0	0.034109	1
D9	0.000456	1	0.524518	0	0.000231	1
D10	0	1	0	1	0	1
D11	0	1	0.000003	1	0.067703	0
D12	0	1	0	1	0.234208	0
D13	0.002030	1	0	1	0.531950	0
D14	0.125453	0	0.001040	1	0.184566	0
D15	0.222383	0	0.002129	1	0.729933	0
D16	0.408081	0	0.187324	0	0.181765	0
D17	0	1	0	1	0.708299	0
D18	0.580130	0	0.628479	0	0.043375	1
D19	0.826649	0	0.794093	0	0.000979	1
D20	0.133873	0	0.119926	0	0.263049	0

Table 13
The Wilcoxon signed rank test between the MAFC and the MKFC algorithms.

MAFC/MKFC	*ACC*		*NMI*		*ARI*	
	p	*h*	*p*	*h*	*p*	*h*
D1	0.000357	1	0	1	0.000211	1
D2	0.008336	1	0.000062	1	0.237465	0
D3	0	1	0.000001	1	0.019652	1
D4	0	1	0	1	0	1
D5	0	1	0	1	0	1
D6	0	1	0	1	0	1
D7	0	1	0	1	0	1
D8	0.004698	1	0.751740	0	0.023085	1
D9	0.000001	1	0.688728	0	0.925677	0
D10	0.000015	1	0.000001	1	0.000124	1
D11	0.003358	1	0.001183	1	0.033416	1
D12	0	1	0	1	0.150778	0
D13	0.057444	0	0.000002	1	0.483229	0
D14	0.692556	0	0.004924	1	0.593410	0
D15	0.798227	0	0.014593	1	0.873772	0
D16	0.892839	0	0.384632	0	0.208796	0
D17	0.000720	1	0.000013	1	0.709152	0
D18	0.917453	0	0.719753	0	0.040230	1
D19	0.072449	0	0.171932	0	0.001228	1
D20	0.890124	0	0.330575	0	0.257378	0

highlight the optimal solution which is selected. The clustering accuracy is also given.

From Figs. 10 and 11, it can be seen that each Pareto front is smooth. Any solution in the Pareto front can be selected as the final solution. For example, if some particular real problem requires a smaller J_m value, then solutions in the upper left part of the Pareto front can be selected. In contrast, solutions in the lower right part of the Pareto front could be selected for problems requiring smaller *MKXB*. In this paper, we focus on obtaining the best clustering accuracy, so we select the final solution by the method described in Section 3.7 and mark it with a square box. The clustering accuracies of two datasets are 95.8% and 92.9% respectively.

In order to analyze statistical characteristics of MAFC and other compared algorithms on data sets, we choose several data sets for carrying out box plot analysis. Because KFC_1, …, KFC_7 are clustering algorithms based on single kernels, they have some similarity. Therefore we select KFC_4, for which the performance is in the middle of the KFC algorithms, as a representative method, and directly denote it as KFC. Fig. 12 shows box plots of partial data sets in terms of *ACC*. Fig. 13 shows box plots of partial data sets in terms of *NMI*. Fig. 14 shows box plots of partial data sets in terms of *ARI*.

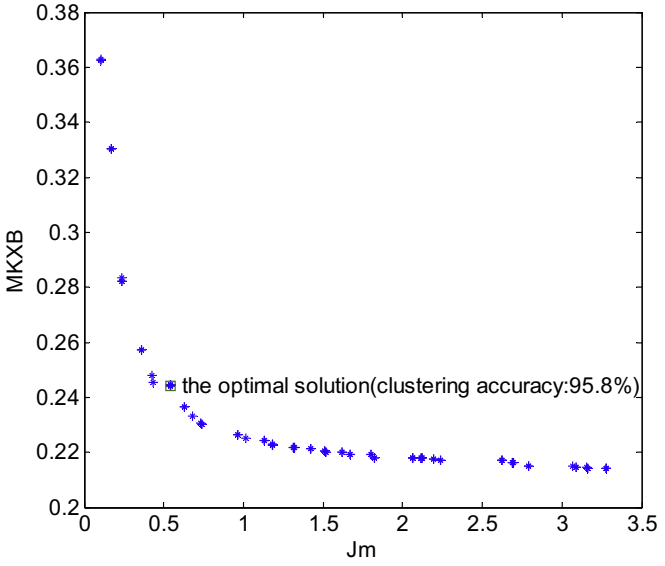


Fig. 10. Pareto front of Optical Recognition of Handwritten Digits (2,7) (D11).

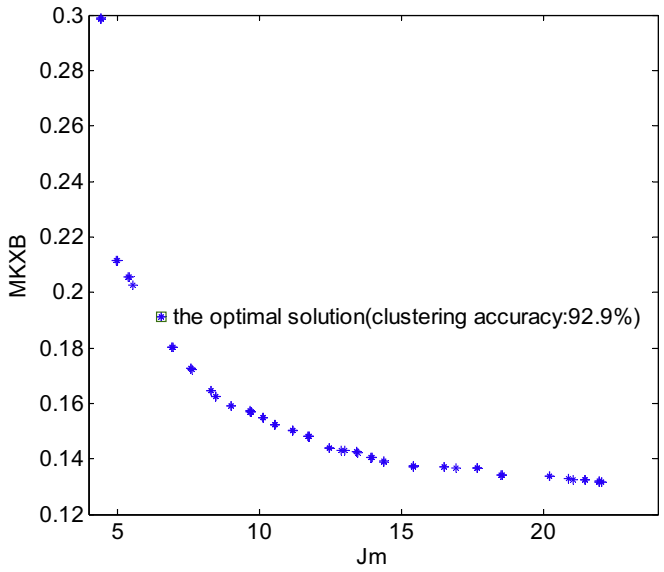


Fig. 11. Pareto front of seeds (D2).

From Fig. 12, we can see that the positions of maximum values and median lines for MAFC are significantly higher than the compared algorithms. This suggests that the clustering accuracy of MAFC is greater than the compared algorithms, and the number of outliers is less than for the compared algorithms.

From Fig. 13, we can observe that MAFC generates superior clustering performance on *Heart*, *Ecoli* and *Balance Scale* from the maximum value and median line of the box plots. On *Heart*, the stability of MAFC is better than that of MKFC. And on *Ecoli* and *Balance Scale*, the number of outliers with MAFC is significantly less than for the compared algorithms.

Fig. 14 suggests that, although MAFC does not demonstrate superior stability over the compared algorithms, its maximum performance is significantly greater than the compared algorithms in terms of *ARI*. The location of the median line is also higher than for the compared algorithms. This suggests that the MAFC algorithm is more suitable for clustering than the compared algorithms.

To provide an intuitive illustration of the premature convergence of other algorithms, and demonstrate how MAFC is more likely to obtain a global optimal solution, we now provide some convergence graphs of the two objectives to show the convergence characteristics of our proposed algorithm. Since each iterative algorithm preserves a portion of the solution, we will take the minimal results of objectives for each generation. Fig. 15 is the convergence graph of J_m objective on the Iris dataset over 50 iterations. Fig. 16 is the convergence graph of *XB* objective on the Iris dataset over 50 iterations.

Fig. 17 is the convergence graph of J_m objective on the seeds dataset over 50 iterations. Fig. 18 is the convergence graph of *XB* objective on the seeds dataset over 50 iterations.

It can be seen from the Figs. 17 and 18 that for two different objective functions, J_m and *XB*, our proposed algorithm remains stable when the number of iterations T is close to 50 generations.

In addition, in order to further prove the effectiveness of the proposed algorithm, we compare the proposed algorithm with a fuzzy multi-objective particle swarm optimization algorithm [65], which optimizes J_m and *XB* simultaneously. Tables 14 and 15 are the clustering results of the proposed algorithm and FMOPSO [65] algorithm on twenty UCI networks.

It can be seen from Table 14, our proposed algorithm can obtain better clustering results than the FMOPSO algorithm on most datasets according to the three index values. It can be seen from Table 15, although FMOPSO has achieved better results on a few datasets, MAFC still obtains better clustering results than the FMOPSO algorithm on most

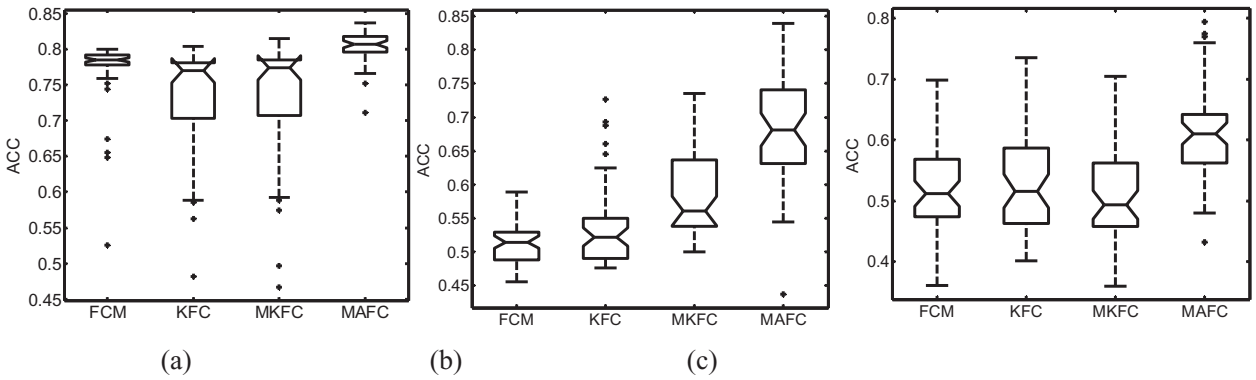


Fig. 12. Box plots of partial data sets in terms of *ACC*. (a) *Heart*. (b) *Ecoli*. (c) *Balance Scale*.

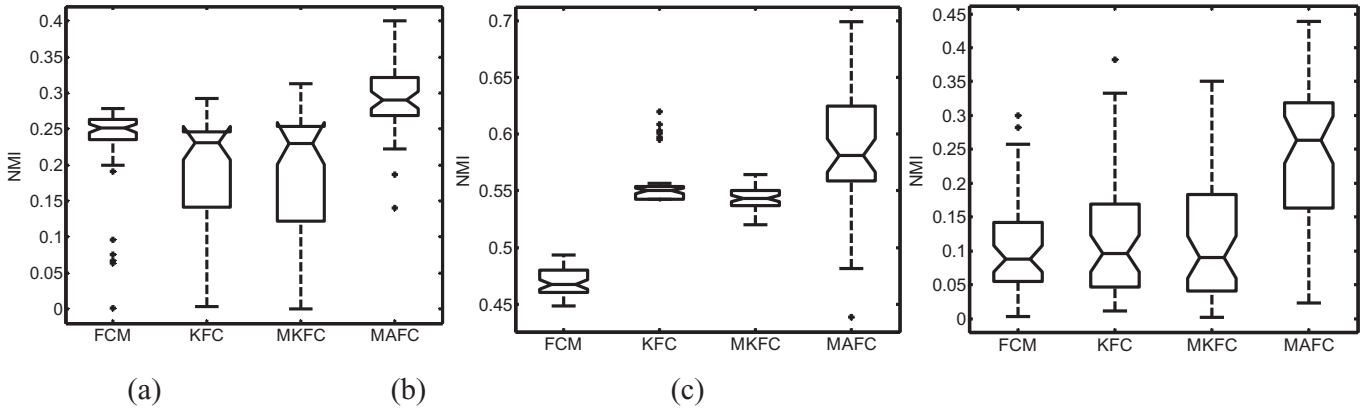


Fig. 13. Box plots of partial data sets in terms of *NMI*. (a) *Heart*. (b) *Ecoli*. (c) *Balance Scale*.

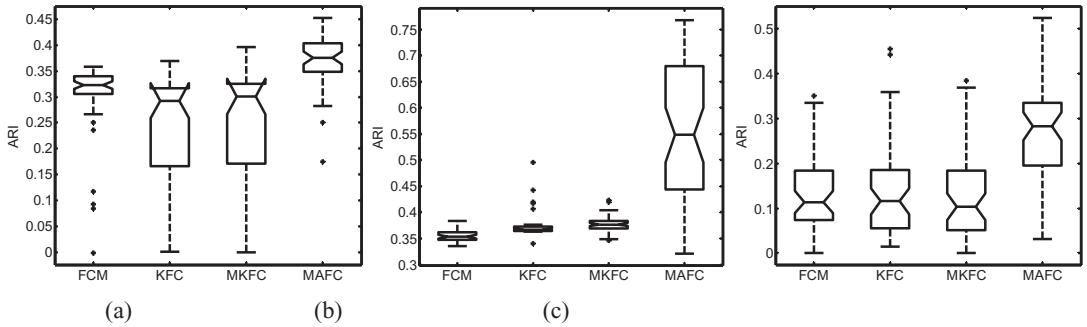


Fig. 14. Box plots of partial data sets in terms of *ARI*. (a) *Heart*. (b) *Ecoli*. (c) *Balance Scale*.

datasets.

4.5. Face clustering

We have also evaluated the proposed MAFC algorithm on face clustering data. Two face datasets employed in this paper are *ORL* and *Yale*. *ORL* includes 400 face images of 40 individuals with 10 different expressions. *Yale* includes 165 face images with different expressions, captured in different lighting conditions. The images are captured from 15 individuals and each individual has 11 different images. Figs. 19 and 20 show some face images of a person from *ORL* and *Yale* respectively. In contrast to the other UCI datasets, these two face datasets cannot be directly utilized for face clustering. Before clustering, we must extract features from the image data.

We first normalize the face images and crop them to 88×88 pixels. Three features are extracted: eigenface, gabor texture and local binary pattern (LBP). The extraction methods that we have used in our experiments are as follows.

Eigenface [66]: principal component analysis (PCA) is utilized to extract eigenface feature. After obtaining the eigenspace Φ, eigenvectors are arranged in descending order according to their corresponding eigenvalues. Because the vast majority of face information is included in the first 5%–10% of the eigenvectors, we only select the top 10% of eigenvectors to make up the eigenspace Φ'. Then each face image is projected into the eigenspace Φ' to obtain a new image vector. The new face image achieves the effect of dimensionality reduction while retaining most of the original image information.

Gabor texture [67]: each face image is filtered with 40 filters, which are generated with 5 different scales and 8 different orientations. All the filtered images undergo globally uniform sampling to get the preliminary dimensionality reduction images. Then we utilize regularized orthogonal fuzzy linear discriminant analysis (ROFLDA) [68] for further

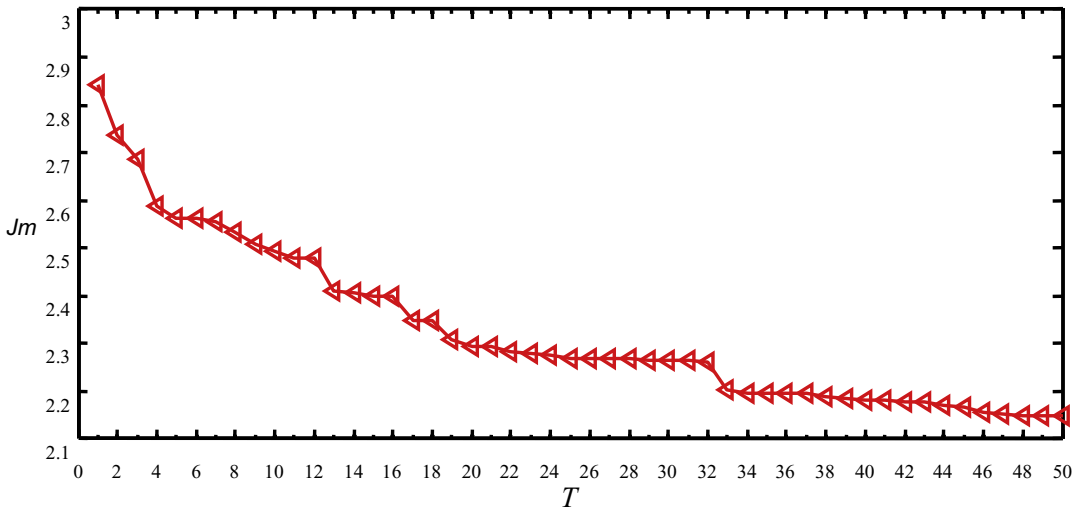


Fig. 15. Convergence graph of J_m objective on the Iris dataset over 50 iterations.

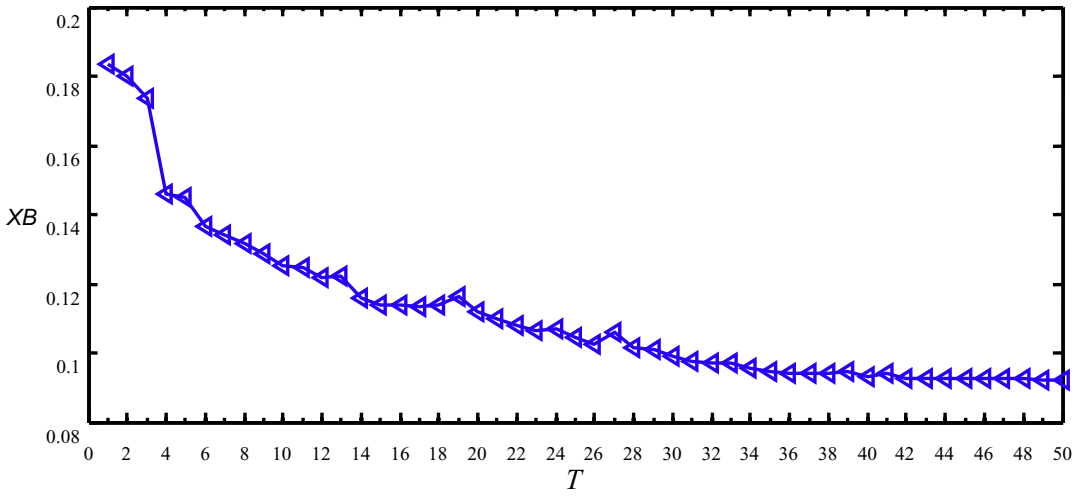


Fig. 16. Convergence graph of *XB* objective on the Iris dataset over 50 iterations.

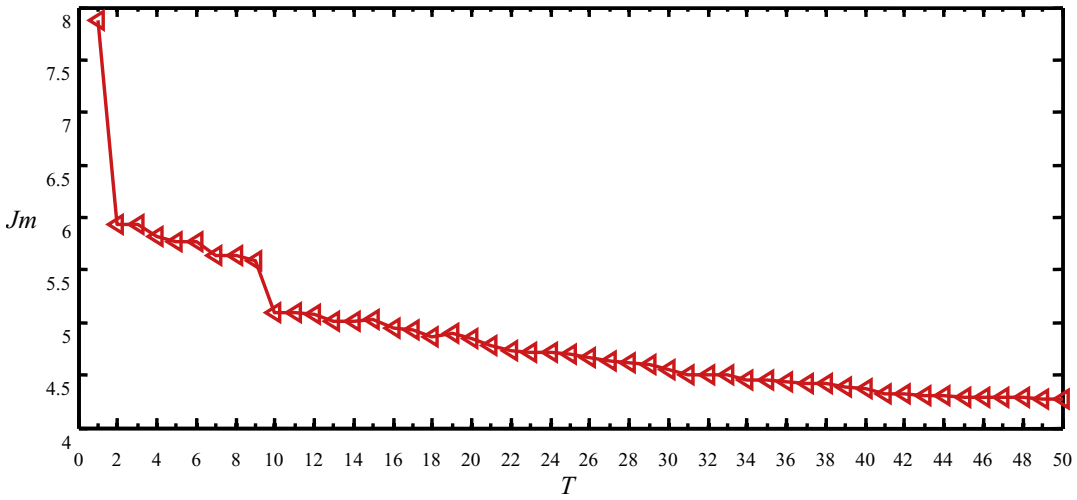


Fig. 17. Convergence graph of J_m objective on the seeds dataset over 50 iterations.

dimensionality reduction to obtain the final image vectors for clustering.

LBP [69]: in order to take into account the features of the location information, each face image is divided into several small regions. We generate a histogram of statistics in each small region, and then connect

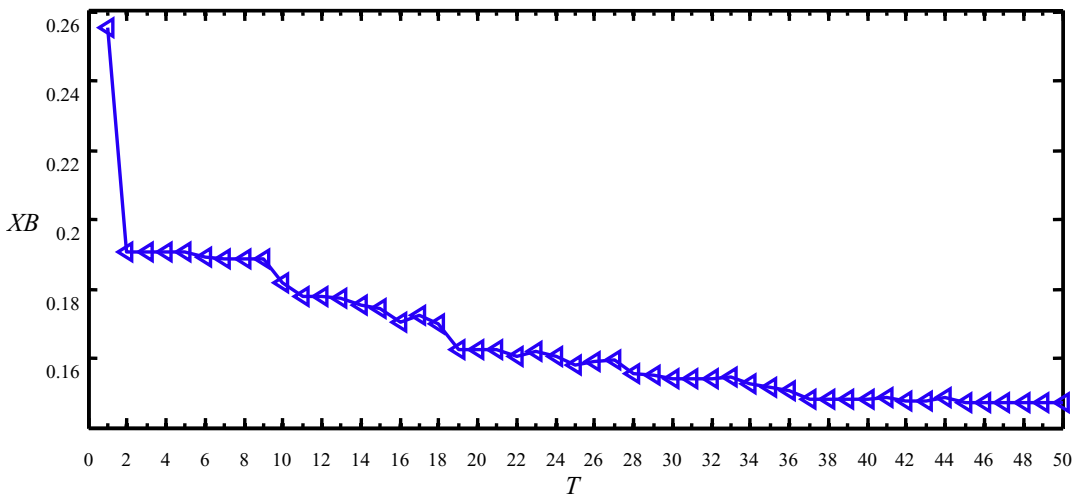


Fig. 18. Convergence graph of *XB* objective on the seeds dataset over 50 iterations.

Table 14
Comparisons about the best results of MAFC and FMOPSO algorithms on UCI datasets.

best	ACC		NMI		ARI	
	FMOPSO	MAFC	FMOPSO	MAFC	FMOPSO	MAFC
D1	0.852	**0.907**	0.762	**0.803**	0.746	**0.759**
D2	0.916	**0.929**	0.719	**0.761**	0.752	**0.798**
D3	**0.612**	0.547	0.281	**0.370**	**0.282**	0.264
D4	0.763	**0.837**	0.221	**0.400**	0.436	**0.452**
D5	0.729	**0.839**	0.581	**0.699**	0.574	**0.768**
D6	0.516	**0.586**	0.002	**0.003**	0.003	**0.010**
D7	0.698	**0.724**	0.121	**0.150**	0.103	**0.187**
D8	0.829	**0.887**	0.492	**0.503**	0.432	**0.599**
D9	0.862	**0.956**	0.520	**0.731**	0.729	**0.831**
D10	0.735	**0.794**	0.312	**0.439**	0.497	**0.525**
D11	0.716	**0.983**	0.719	**0.878**	0.628	**0.934**
D12	0.524	**0.889**	0.456	**0.728**	0.482	**0.740**
D13	**0.878**	0.855	0.514	**0.558**	0.591	**0.660**
D14	0.593	**0.706**	0.118	**0.125**	0.124	**0.168**
D15	0.218	**0.342**	**0.423**	0.357	**0.197**	0.186
D16	0.427	**0.484**	0.292	**0.318**	0.203	**0.218**
D17	0.835	**0.941**	0.523	**0.720**	0.623	**0.777**
D18	**0.706**	0.690	0.592	**0.605**	0.433	**0.472**
D19	0.721	**0.969**	0.591	**0.810**	0.783	**0.875**
D20	0.462	**0.739**	0.421	**0.435**	0.118	**0.416**

Table 15
Comparisons about the mean results of MAFC and FMOPSO algorithms on UCI datasets.

mean	ACC		NMI		ARI	
	FMOPSO	MAFC	FMOPSO	MAFC	FMOPSO	MAFC
D1	0.804	**0.885**	0.749	**0.777**	**0.724**	0.717
D2	**0.903**	0.895	**0.703**	0.690	**0.723**	0.705
D3	0.415	**0.459**	0.229	**0.303**	**0.190**	0.183
D4	0.701	**0.804**	0.137	**0.293**	0.297	**0.368**
D5	0.682	**0.696**	0.547	**0.580**	0.527	**0.550**
D6	0.485	**0.573**	0.001	**0.003**	0.001	**0.002**
D7	0.621	**0.703**	**0.112**	0.104	0.082	**0.156**
D8	0.713	**0.870**	0.381	**0.457**	0.408	**0.531**
D9	0.791	**0.914**	0.512	**0.593**	0.641	**0.686**
D10	**0.683**	0.617	**0.254**	0.241	0.227	**0.272**
D11	0.512	**0.933**	0.604	**0.675**	0.503	**0.885**
D12	0.498	**0.806**	0.342	**0.637**	0.364	**0.600**
D13	**0.800**	0.759	0.429	**0.528**	0.472	**0.522**
D14	0.525	**0.670**	0.068	**0.089**	0.092	**0.118**
D15	0.190	**0.298**	**0.365**	0.306	0.102	**0.139**
D16	**0.416**	0.405	0.239	**0.261**	**0.149**	0.143
D17	0.701	**0.905**	0.487	**0.576**	0.520	**0.660**
D18	**0.683**	0.670	0.504	**0.562**	0.398	**0.452**
D19	0.624	**0.960**	0.482	**0.768**	0.683	**0.844**
D20	0.327	**0.606**	0.307	**0.380**	0.094	**0.283**

the histograms of all the small regions together to form an LBP feature for the image. A uniform LBP with eight neighbors and radius 1 is employed in these experiments, and the obtained LBP feature undergoes dimensionality reduction by ROFLDA to get the final LBP feature.

After feature extraction, each feature is represented as a vector. Then it is used for face clustering via a Gaussian kernel mapping. Here η is set to 0.005, and clustering algorithms with three different features (Eigenface, Gabor texture, LBP) are denoted as KFC_e, KFC_g, and KFC_l respectively. And three different feature vectors are combined together for multi-kernel clustering.

In the following experiments, MAFC is compared with MKFC, KFC_e, KFC_g, KFC_l [7]. The parameter settings in these experiments are the same as for the previously described data clustering experiments. In Tables 16 and 17, *mean* denotes the mean results of 50 independent runs. *best* denotes the best results in 50 independent runs. The bold data represents the best result under the evaluation index.

From Table 16, we can conclude that for the *ORL* face dataset, clustering with three features (eigenface, gabor texture, LBP) obtains a better clustering performance than that of a single feature, suggesting that multi-kernel clustering is more effective than single-kernel clustering.

Fig. 19. Some face images of a person in *ORL*.

Fig. 20. Some face images of a person in *Yale* dataset.

Although *ARI* produced by MAFC in mean results equals to that produced by MKFC, the best result on *ARI* produced by MAFC is better. The improvement of *ACC* in mean result is not significant compared with

Table 16
Comparisons of different algorithms on face dataset *ORL*.

Evaluation index	KFC_e		KFC_g		KFC_l		MKFC		MAFC	
	best	*mean*	*best*	*mean*	*best*	*mean*	*best*	*mean*	*best*	*mean*
ACC	0.345	0.308	0.368	0.340	0.530	0.500	0.550	0.528	**0.623**	**0.549**
NMI	0.553	0.546	0.618	0.602	0.705	0.670	0.711	0.681	**0.743**	**0.712**
ARI	0.190	0.172	0.208	0.183	0.374	0.322	0.411	**0.374**	**0.452**	**0.374**

Table 17
Comparisons of different algorithms on face dataset *Yale*.

Evaluation index	KFC_e		KFC_g		KFC_l		MKFC		MAFC	
	best	*mean*	*best*	*mean*	*best*	*mean*	*best*	*mean*	*best*	*mean*
ACC	0.503	0.425	0.576	0.514	0.593	0.486	0.527	0.428	**0.612**	**0.573**
NMI	0.558	0.498	0.637	0.581	0.616	0.554	0.573	0.533	**0.712**	**0.610**
ARI	0.363	0.291	0.471	0.320	0.473	0.352	0.351	0.308	**0.488**	**0.388**

MKFC. However, improvement of the best result is obvious. The above suggests that the optimization ability of MAFC is better than MKFC.

Table 17 shows that, for the *Yale* face dataset, the clustering performance has been significantly improved after the introduction of multi-objective optimization. Whether it is in the best results or in the mean results, MAFC has achieved better clustering results. Especially in mean results, the improvements on *ACC* produced by MAFC increase by nearly 15% with respect to MKFC, which shows that MAFC algorithm has advantages over single-objective clustering algorithms.

In summary, MAFC has greater applicability and better clustering discrimination ability than other compared algorithms on a variety of difficult clustering problems.

5. Conclusions

For clustering optimization problems, datasets collected from nature are of wide variety with significantly different distribution and dimension. Perfect clusters are also not well defined for many problems. Therefore, the problem of optimal clustering is challenging. For example, determining optimal parameters for the desired clusters is difficult, and ideally we would like to obtain generic clustering algorithms which require minimum domain-specific a-priori knowledge. The time complexity of clustering algorithms should be low enough to run high dimensional data on large databases easily. Datasets may contain noise/outliers. Desired clusters may have varied degree of overlap. These are problems that need to be considered and solved in clustering optimization problems.

In this paper, we proposed a multi-objective artificial immune algorithm for fuzzy clustering based on multiple kernels. It extends single-objective clustering to multi-objective clustering and can effectively overcome the limitations of other state-of-the-art clustering algorithms which result in poor clustering quality. The MAFC algorithm improves the scope of application of FCM and makes the algorithm suitable for general distribution data. Since single-objective clustering algorithms ignore the geometric distribution information of a dataset, they are more prone to local optima convergence. In contrast, the MAFC algorithm can search for a solution more globally and avoid local optima. For some datasets, the non-linear relationships between data can be especially difficult to discover, making it difficult to cluster them accurately. The MAFC algorithm makes these data linearly separable in the new feature space via kernel function mapping, which improves the quality of clustering. We utilize an artificial immune algorithm to address this multi-objective optimization problem. By antibody population initialization, clone proliferation, non-uniform mutation and a uniformity maintenance strategy, the artificial immune algorithm can maintain the competitiveness and diversity of the population simultaneously. This helps avoid the degradation and prematurity problems of conventional genetic algorithms, and makes the proposed algorithm more likely to converge on globally optimal solutions. Although MAFC has no advantage over other single-objective algorithms in terms of time complexity, the experimental results on real datasets show that MAFC is effective and practical. In future work, we will make further efforts in terms of reducing the time complexity of the proposed algorithm.

Acknowledgements

We would like to express our sincere appreciation to the editors and the anonymous reviewers for their insightful comments, which have greatly helped us in improving the quality of the paper. This work was partially supported by the National Natural Science Foundation of China, under Grants 61773304, 61671350, 61371201 and 1772399, the Program for Cheung Kong Scholars and Innovative Research Team in University under Grant IRT1170, the Fundamental Research Funds for the Central Universities, and the Innovation Fund of Xidian University. Rustam Stolkin was supported by a Royal Society Industry Fellowship.

References

[1] Z. Yan, W. Luo, C. Bu, et al., Clustering spatial data by the neighbors intersection and the density difference, Big Data Computing Applications and Technologies (BDCAT), in: 2016 IEEE/ACM 3rd International Conference on, IEEE, 2016, pp. 217–226.
[2] X. Peng, Y. Wu, Large-scale cooperative co-evolution using niching-based multi-modal optimization and adaptive fast clustering, Swarm Evolut. Comput. 35 (2017) 65–77.
[3] R.O. Duda, P.H. Hart, D.G. Stock, Pattern Classification, Wiley, New York, 2001.
[4] R.H. Shang, P.P. Tian, L.C. Jiao, et al., A spatial fuzzy clustering algorithm with kernel metric based on immune clone for SAR image segmentation, IEEE J. Select. Top. Appl. Earth Observ. Remote Sens. 9 (4) (2016) 1640–1652.
[5] D.M. Tsai, C.C. Lin, Fuzzy C-means based on clustering for linearly and nonlinearly separable data, Pattern Recogn. 44 (8) (2011) 1750–1760.
[6] R. Liu, W. Luo, L. Yue, Classifying and clustering in negative databases, Front. Comput. Sci. 7 (6) (2013) 864–874.
[7] H.C. Huang, Y.Y. Chuang, C.S. Chen, Multiple kernel fuzzy clustering, IEEE Trans. Fuzzy Syst. 20 (1) (2012) 120–134.
[8] F. Camastra, A. Verri, A novel kernel method for clustering, IEEE Trans. Pattern Anal. Mach. Intell. 27 (5) (2005) 801–804.
[9] I.S. Dhillon, Y. Guan, B. Kulis, Kernel k-means spectral clustering and normalized cuts, in: Proc. 10th ACM Int. Conf. Knowl. Discovery Data Mining, 2004, pp. 551–556.
[10] D.W. Kim, K.Y. Lee, D. Lee, K.H. Lee, Evaluation of the performance of clustering algorithms kernel-induced feature space, Pattern Recogn. 38 (4) (2005) 607–611.
[11] D. Graves, W. Pedrycz, Kernel-based fuzzy clustering and fuzzy clustering: a comparative experimental study, Fuzzy Sets Syst. 161 (4) (2010) 522–543.
[12] X.W. Yang, G.Q. Zhang, J. Lu, J. Ma, A kernel fuzzy c-means clustering-based fuzzy support vector machine algorithm for classification problems with outliers or noises, IEEE Trans. Fuzzy Syst. 19 (1) (2011) 105–115.
[13] D.Q. Zhang, S.C. Chen, Clustering incomplete data using Kernel-based Fuzzy C-Means algorithm, Neural Process. Lett. 18 (3) (2003) 155–162.
[14] S. Zhou, J. Gan, Mercer kernel fuzzy c-means algorithm and prototypes of clusters, in: Proc. Int. Conf. Data Engineering and Automated Learning, 2004, pp. 613–618.

[15] B. Zhao, J. Kwok, C. Zhang, Multiple kernel clustering, in: Proc.9th SIAM Int. Conf. Data Mining, 2009, pp. 638–649.
[16] J.C. Bezdek, Pattern Recognition with Fuzzy Objective Function Algorithm, Academic, New York, 1981.
[17] N.R. Pal, J.C. Bezdek, On cluster validity for the fuzzy c-means model, IEEE Trans. Fuzzy Syst. 3 (3) (1995) 370–379.
[18] D.W. Kim, K.H. Lee, D. Lee, On cluster validity index for estimation of the optimal number of fuzzy clusters, Pattern Recogn. 37 (2004) 2009–2025.
[19] A. Mukhopadhyay, U. Maulik, A multiobjective approach to MR brain image segmentation, Appl. Soft Comput. 11 (1) (2011) 872–880.
[20] K.S.N. Ripon, M.N.H. Siddique, Evolution multiobjective clustering for overlapping cluster detection, in: Proc. IEEE Cogr. Evol. Comput., 2009, pp. 976–982.
[21] J. Handl, J.D. Knowles, Evolutionary multiobjective clustering, in: Proc. 8th Int. Conf. PPSN, 2004, pp. 1081–1091.
[22] J. Handl, J. Knowles, An evolutionary approach to multi-objective clustering, IEEE Trans. Evol. Comput. 11 (1) (2007) 56–76.
[23] G.N. Demir, A.S. Uyar, S.G. Oguducu, Multi-objective evolutionary clustering of web user sessions: a case study in web page recommendation, Soft Comput. 14 (6) (2010) 579–597.
[24] A. Mukhopadhyay, U. Maulik, Unsupervised pixel classification in satellite imagery using multiobjective fuzzy clustering combined with SVM classifier, IEEE Trans. Geosci. Rem. Sens. 47 (4) (2009) 1132–1138.
[25] Y. Liu, T. Özyer, R. Alhajj, K. Barker, Integrating multiobjective genetic algorithm and validity analysis for locating and ranking alternative clustering, Informatica 29 (2005) 33–40.
[26] J. Du, E.E. Korkmaz, R. Alhajj, K. Barker, Alternative clustering by utilizing multiobjective genetic algorithm with linked-list based chromosome encoding, in: Proc. 4th Int. Conf. MLDM, 2005, pp. 346–355.
[27] A. Mukhopadhyay, U. Maulik, S. Bandyopadhyay, Unsupervised cancer classification through SVM-boosted multiobjective fuzzy clustering with majority voting ensemble, in: Proc. IEEE Congr. Evol. Comput, 2009, pp. 255–261.
[28] U. Maulik, S. Bandyopadhyay, Performance evaluation of some clustering algorithms and validity indices, IEEE Trans. Pattern Anal. Mach. Intell. 24 (12) (2002) 1650–1654.
[29] L. Zhu, L.B. Cao, J. Yang, Multiobjective evolutionary algorithm-based soft subspace clustering, in: Proc. IEEE Congr. Evol. Comput, 2012, pp. 1–8.
[30] S. Bandyopadhyay, U. Maulik, A. Mukhopadhyay, Multi-objective genetic clustering for pixel classification in remote sensing imagery, IEEE Trans. Geosci. Rem. Sens. 45 (5) (2007) 1506–1511.
[31] B.Y. Qu, Y.S. Zhu, Y.C. Jiao, et al., A survey on multi-objective evolutionary algorithms for the solution of the environmental/economic dispatch problems, Swarm Evolut. Comput. 38 (2018) 1–11.
[32] A. García-Nájera, A. López-Jaimes, An investigation into many-objective optimization on combinatorial problems: analyzing the pickup and delivery problem, Swarm Evolut. Comput. 38 (2018) 218–230.
[33] J. Luo, Y. Yang, X. Li, et al., A decomposition-based multi-objective evolutionary algorithm with quality indicator, Swarm Evolut. Comput. (2017).
[34] X.N. Shen, L.L. Minku, N. Marturi, et al., A Q-learning-based memetic algorithm for multi-objective dynamic software project scheduling, Inf. Sci. 428 (2018) 1–29.
[35] X.N. Shen, Y. Han, J.Z. Fu, Robustness measures and robust scheduling for multi-objective stochastic flexible job shop scheduling problems, Soft Comput. 21 (21) (2017) 6531–6554.
[36] X. Peng, S. Zhang, X. Lei, Multi-target trapping in constrained environments using gene regulatory network-based pattern formation, Int. J. Adv. Rob. Syst. 13 (5) (2016), 1729881416670152.
[37] R.H. Shang, Y.Y. Wang, J. Wang, L.C. Jiao, S. Wang, L.P. Qi, A multi-population cooperative coevolutionary algorithm for multi-objective capacitated arc routing problem, Inf. Sci. 277 (2014) 609–642.
[38] K. Deb, A. Pratap, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evol. Comput. 6 (2) (2002) 182–197.
[39] R.H. Shang, L.P. Qi, L.C. Jiao, R. Stolkin, Y.Y. Li, Change detection in SAR images by artificial immune multi-objective clustering, Eng. Appl. Artif. Intell. 31 (2014) 53–67.
[40] J. Yoo, P. Hajela, Immune network simulations in multicriterion design, Struct. Optim. 18 (2–3) (1999) 85–94.
[41] C.A.C. Coello, N.C. Cortés, An approach to solve multi-objective optimization problems based on an artificial immune system, in: J. Timmis, P.J. Bentley (Eds.), First International Conference on Artificial Immune Systems (ICARIS'2002), University of Kent at Canterbury, UK, 2002, pp. 212–221. September.
[42] F. Campelo, F.G. Guimaraes, R.R. Saldanha, A novel multiobjective immune algorithm using nondominated sorting, in: 11th International IGTE Symposium on Numerical Field Calculation in Electrical Engineering, 2004. Seggauberg, Austria.
[43] R.H. Shang, L.C. Jiao, F. Liu, W.P. Ma, A novel Immune clonal MO problems, IEEE Trans. Evol. Comput. 16 (1) (2012) 35–50.
[44] D.D. Yang, L.C. Jiao, M.G. Gong, F. Liu, Artificial immune multi-objective SAR image segmentation with fused complementary features, Inf. Sci. 181 (13) (2011) 2797–2812.
[45] O. Linda, M. Manic, General type-2 fuzzy C-means algorithm for uncertain fuzzy clustering, IEEE Trans. Fuzzy Syst. 20 (4) (2012) 883–897.
[46] S.Y. Li, C.F. Hu, An interactive satisfying method based on alternative tolerance for multiple objective optimization with fuzzy parameters, IEEE Trans. Fuzzy Syst. 16 (5) (2008) 1151–1160.
[47] C.A.C. Coello, G.T. Pulido, M.S. Lechuga, Handling multiple objectives with particle swarm optimization, IEEE Trans. Evol. Comput. 8 (3) (2004) 256–279.
[48] M.A. Abido, Two-level of nondominated solutions approach to multiobjective particle swarm optimization. Proceedings of the 9th annual conference on Genetic and evolutionary computation, ACM (2007) 726–733.
[49] P. Koduru, S. Das, S.M. Welch, Multi-objective hybrid PSO using μ-fuzzy dominance. Proceedings of the 9th annual conference on Genetic and evolutionary computation, ACM (2007) 853–860.
[50] X. Li, A non-dominated sorting particle swarm optimizer for multiobjective optimization, in: Genetic and Evolutionary Computation Conference, Springer, Berlin, Heidelberg, 2003, pp. 37–48.
[51] C.A.C. Coello, N.C. Cortés, Solving multiobjective optimization problems using an artificial immune system, Genet. Program. Evolvable Mach. 6 (2) (2005) 163–190.
[52] A. Lanaridis, A. Stafylopatis, An artificial immune network for multi-objective optimization, in: International Conference on Artificial Neural Networks, Springer, Berlin, Heidelberg, 2010, pp. 531–536.
[53] M. Gong, L. Jiao, H. Du, L. Bo, Multiobjective immune algorithm with nondominated neighbor-based selection, Evol. Comput. 16 (2) (2008) 225–255.
[54] N. Khan, D.E. Goldberg, M. Pelikan, Multi-objective Bayesian optimization algorithm, in: Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation, Morgan Kaufmann Publishers Inc., 2002, 684-684.
[55] M. Laumanns, J. Ocenasek, Bayesian Optimization Algorithms for Multi-objective Optimization. International Conference on Parallel Problem Solving from Nature, Springer, Berlin, Heidelberg, 2002, pp. 298–307.
[56] H. Li, Q. Zhang, Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II, IEEE Trans. Evol. Comput. 13 (2) (2009) 284–302.
[57] Q. Zhang, H. Li, MOEA/D: a multiobjective evolutionary algorithm based on decomposition, IEEE Trans. Evol. Comput. 11 (6) (2007) 712–731.
[58] D. Graves, W. Pedrycz, Kernel-based fuzzy clustering and fuzzy clustering: a comparative experimental study, Fuzzy Sets Syst. 161 (4) (2010) 522–543.
[59] X. Zhao, X.S. Gao, Z.C. Hu, Evolutionary programming based on non-uniform mutation, Appl. Math. Comput. 192 (1) (2007) 1–11.
[60] Z. Michalewicz, C.Z. Janikow, Handling constraints in genetic algorithms, ICGA (1991) 151–157.
[61] M.G. Wu, B. Schölkopf, A local learning approach for clustering, in: Proc. Adv. Neural Inf. Process. Syst., 2007, pp. 1529–1536.
[62] H. Zeng, Y.M. Cheung, Semi-supervised maximum margin clustering with pairwise constraints, IEEE Trans. Knowl. Data Eng. 24 (5) (2012) 926–939.
[63] L. Hubert, P. Arabie, Comparing partitions, J. Classif. 2 (1) (1985) 193–218.
[64] N.J. Martarelli, M.S. Nagano, A constructive evolutionary approach for feature selection in unsupervised learning, Swarm Evolut. Comput. (2018).
[65] B.A. Attea, A fuzzy multi-objective particle swarm optimization for effective data clustering, Memetic Comput. 2 (4) (2010) 305–312.
[66] P.N. Belhumeur, J.P. Hespanha, D.J. Kriegman, Eigenfaces versus fisherfaces: recognition using class specific linear projection, IEEE Trans. Pattern Anal. Mach. Intell. 19 (7) (1997) 711–720.
[67] J.F. Ye, Y.Z. Zhan, Facial expression feature extraction based on gabor wavelet transformation, in: Proc. IEEE Int. Conf. Syst., Man, Cybern., 2004, pp. 2215–2219.
[68] J.P. Ye, T. Xiong, Computational and theoretical analysis of null space and orthogonal linear discriminant analysis, J. Mach. Learn. Res. 7 (2006) 1183–1204.
[69] T. Ojala, M. Pietikainen, T. Maenpaa, Multiresolution gray-scale and rotation invariant texture classification with local binary patterns, IEEE Trans. Pattern Anal. Mach. Intell. 24 (7) (2002) 971–987.