Y

A CLUSTERING ENSEMBLE LEARNING METHOD BASED ON THE ANT COLONY CLUSTERING ALGORITHM

H. PARVIN[1], A. BEIGI[2], AND N. MOZAYANI[3]

Abstract. A very promising approach to reach a robust partitioning is to use ensemble-based learning. In this way, the classification/clustering task is more reliable, because the classifiers/clusterers in the ensemble cover the faults of each other. The common policy in clustering ensemble based learning is to generate a set of primary partitionings that are different from each other. These primary partitionings could be generated by a clustering algorithm with different initializations. It is popular to filter some of these primary partitionings, i.e. a subset of the produced partitionings are selected for the final ensemble. The selection phase is done to reach a diverse ensemble. A consensus function finally aggregates the ensemble into a final partitioning called also the consensus partitioning. Another alternative policy in the clustering ensemble based learning is to use the fusion of some primary partitionings that come from naturally different sources. On the other hand, swarm intelligence is also a new topic where the simple agents work in such a way that a complex behavior can be emerged. The necessary diversity for the ensemble can be achieved by the inherent randomness of swarm intelligence algorithms. In this paper we introduce a new clustering ensemble learning method based on the ant colony clustering algorithm. Indeed ensemble needs diversity vitally and swarm intelligence algorithms are inherently involved in randomness. Ant colony algorithms are powerful metaheuristics that use the concept of swarm intelligence. Different runnings of ant colony clustering on a dataset result in a number of diverse partitionings. Considering these results totally as a new space of the dataset we employ a final clustering by a simple partitioning algorithm to aggregate them into a consensus partitioning. From another perspective, ant colony clustering algorithms have many parameters. Effectiveness of the ant colony clustering methods are questionable because they depend on many parameters. On a test dataset, these parameters should be tuned to obtain a desirable result. But how to define them in a real task does not clear. The proposed clustering framework lets the parameters be free to be changed, and compensates non-optimality of the parameters by the ensemble power. Experimental results on some real-world datasets are presented to demonstrate the effectiveness of the proposed method in generating the final partitioning.

Keywords: Ant Colony, Data Fusion, Clustering.

AMS Subject Classification: 68T20

1. Introduction

Data clustering as an important unsupervised learning task is a very challenging problem. The objective function of clustering is to partition a set of unlabeled objects into homogeneous groups or clusters [1]. There are many applications that may use clustering techniques to discover hidden structures in data, such as data mining [1], information retrieval, image segmentation, and machine learning [2]. In real-world problems, the clusters may have different shapes, sizes, degrees of data sparseness, and degrees of data separation. To use any clustering technique,

[1]Nourabad Mamasani Branch, Islamic Azad University, Nourabad Mamasani, Iran,
e-mail: hamidparvin@mamasaniiau.ac.ir .
[2]School of Computer Engineering, Iran University of Science and Technology, Tehran, Iran,
e-mail: beigi@iust.ac.ir .
[3]School of Computer Engineering, Iran University of Science and Technology, Tehran, Iran,
e-mail: mozayani@iust.ac.ir
Manuscript received xx.

we need to define a similarity measure. The similarity measure should receive two patterns as input, and then give a value (often between [0, 1]) reflecting similarity of the two input patterns. When there is no prior knowledge about cluster shapes, choosing a specialized clustering method is not an easy task [3]. For extracting a proper partitioning out of a given dataset, one requires both clustering expertise and insight about dataset to choose a single appropriate clustering algorithm and also to set parameters of the selected clustering algorithm to appropriate values. Instead of running the risk of picking an inappropriate clustering algorithm, we can benefit from a different alternative solution. The alternative solution is based on the ensemble learning. It can be achieved by applying all possible clustering algorithms (and possibly with all of their initializations) to the dataset and then combining their output partitionings into a consensus partitioning. This is the basic idea behind cluster ensembles [4].

Studies in the field of data mining in the last few years have tended to ensemble methods (combinational methods). Clustering ensemble methods as a subfield of data mining also attempt to find a better, more robust, novel and accurate clustering solution by fusing information from several primary data partitionings [5]. Ensemble learning originates from wisdom of crowd in the society and humanity sciences. In the ensemble-based learning, it is believed that the clustering/classification task is more reliable, because the clusterers/classifiers in the ensemble can cover the faults of each other.

The common policy in clustering ensemble based learning is to generate a set of primary partitionings that are different from each other. These primary partitionings could be generated by a clustering algorithm with different initializations. Some of the produced partitionings may be selected for the final ensemble. The selection phase is done to reach a diverse ensemble. A consensus function finally aggregates the ensemble into a final partitioning named also a consensus partitioning. Another alternative policy in the clustering ensemble learning is to use the fusion of some primary partitionings that come from naturally different sources to ensure they are diverse. In any ensemble diversity between elements of the ensemble has a vital role. It means if an ensemble lacks diversity, it will be surly ineffective. It has been widely used as an inevitable element in a classifier ensemble. Expectedly diversity concept is also vital for a pool of partitionings to be considered as a successful ensemble. To ensure that a number of partitionings are diverse one can use the same strategies used to build a diverse classifier ensemble that include [7, 15, 16]:

(1) **Different subsets of features**: Each partitioning available in the ensemble is obtained by employing a clustering algorithm on a different projection of dataset.
(2) **Different clustering algorithms**: Each partitioning available in the ensemble is obtained using a different clustering algorithm.
(3) **Randomizing**: Each partitioning available in the ensemble is obtained by employing a clustering algorithm with a different initialization. Note that some clustering algorithms like *k*-means are inherently sensitive to their initializations.
(4) **Different datasets**: Each partitioning available in the ensemble is obtained employing a clustering algorithm on a different resampled sub-dataset of original dataset.

There are many types of consensus functions that solve the problem of clustering ensemble heuristically. The most of them require the number of clusters to be specified as a priori, but in practice the number of clusters is usually unknown. An ensemble framework based on swarm intelligence [8] is proposed for addressing the problem of clustering. In particular, given a set of partitionings, Kennedy and Russell apply ant colony clustering (ACC) algorithm to the co-association matrix computed from the ensemble to produce the final partitioning, and automatically determine the number of clusters.

The first ACC model has introduced by Deneubourg et al. [9]. His model possesses the swarm intelligence of real ants, and has inserted into a robot for the object collecting task. Lumer and Faieta [10] have improved the Deneubourg's model by adding the Euclidean distance formula

to the similarity density function and giving ants three kinds of abilities: speed, short-term memory, and behavior exchange.

ACC algorithm is inspired by how ants organize their food in their nests. It typically involves two key operations: picking up an object from a cluster and dropping it off into another cluster [11]. At each step, some ants perform picking-up and dropping-off actions based on some notions of similarity between an object and the clusters. Azimi et al. define a similarity measure based on the co-association matrix [12]. Their clustering process is completely decentralized and self-organized, allowing the clustering structure to emerge automatically from the data. As a result, They can accurately determine the number of clusters in the data. The experimental results show that their proposed consensus function is very effective in predicting the number of clusters and also achieves reliable clustering performance. In addition, by introducing some simple heuristics, they detect the marginal and outlier samples in the data to improve their final clustering.

Liu et al. propose a method for incrementally constructing a knowledge model for a dynamically changing database, using an ACC algorithm. They use information-theoretic metrics to overcome some inherent problems of ant-based clustering. Entropy governs the picking-up and dropping-off behaviors, while movement is guided by pheromones. They show that dynamic clustering can provide significant benefits over static clustering for a realistic problem scenario [13].

In this paper extending our recently proposed framework [6], it is tried to bring two successful concepts in the field of clustering: (a) ensemble concept and (b) swarm concept. We introduce a new clustering ensemble learning method based on the ACC algorithm. Indeed ensemble needs diversity vitally and swarm intelligence is inherently involved in randomness. Different runnings of ACC algorithm with different initializations on a dataset result in a number of diverse partitionings. Considering these results totally as a new space of the dataset we employ a simple partitioning algorithm to aggregate them into a consensus partitioning. Our experimental results on some real-world datasets have been presented previously to demonstrate the effectiveness of the proposed method in generalization of the final partitioning [6]. However this paper explores the effectiveness of the proposed method by some new metrics to generalize the conclusion. To show how much the swarm intelligence has contribution in reaching better results is the question to be answered in the paper.

The rest of the paper is organized as follows. Section 2 explains the original ACC. The proposed new space is presented in Section 3. In Section 4, the experimental results of the some clustering algorithms over original feature space and mapped feature space are compared. The paper is concluded in Section 5.

2. Ant Colony Clustering

General form of original ACC algorithm is presented in this section. The ACC algorithm includes a population of ants. Each ant operates as an autonomous agent that reorganizes data patterns during exploration to reach an optimal clustering. Pseudo code of the original ACC algorithm is depicted in Figure 1.

At the first step each object represented by a multi-dimensional vector (feature vector) in the original feature space is randomly scattered in a two-dimensional space. In each step each ant randomly searches the space. It uses its short-term memory to jump into a location that is potentially near to an object if it is unloaded. It can picks up or drops off an object using a probability density obtained by equation (1):

$$f(o_i) = max\{0, \frac{1}{s^2} \sum_{o_j \in Neigh_{s\times s}(r)} [1 - \frac{d(o_j, o_i)}{\alpha \times (1 + \frac{v-1}{v_{max}})}]\}, \quad (1)$$

where $Neigh_{s\times s}(r)$ is the observable local area (or the set of observable rooms) for an ant located at room r. $Neigh_{s\times s}(r)$ should be adjacent to the location r. It is worthy to mention that each

```
inilialli.ing p;u·amele•;
for each ant a
        plałe random a in a position nul occupied by olher anls;
end;
fo1· al:h objed o
        place random o in a position not occupied by other objects;
end ;
for t=l:t""·'
        for· ach an l a
                fiFSelect a random number uniformly from range [0,1];
                r=position(a)
                if(loaded(a) and (is_empty(r)))
                        if(.,"<Jidrop) o=llrop(a);
                                put(r,o);
                                save(o,r,q);
                        end;
                end ;
                elseif(not (loaded(a) or (is_empty(r))))
                        if(J,K]Jpi<)
                                o=rcmovc(r);
                                pick_up(a,o);
                                scarch_and_jump(a,o);
                        end;
                end;
                else
                        wandcr(a,J,,.IVJi);
                end;
        end;
end;
```

FIGURE 1. Pseudo code of original ant colony clustering algorithm.

room including *Neighs xs(r)* and *r* is a two-dimensional vector (standing for *x* and *y* axes). Threshold o: is a parameter that scales the distance between each pair of objects and speed parameter *v* control the volume of feature space that an ant explores in each epoch of algorithm. The function *d(Oj,*oi) is the distance between two objects *0i* and *Oj* in the original feature space. It is calculated by equation (2):

$$d(oj,oi) = \sqrt{\sum_{k=1}^{m} l,)oik\text{-}Ojk)^2}, \tag{2}$$

where m is the number of original features and *Oik* is kth feature of object *Oi*. Probability that an unloaded ant takes an object that is in the room occupied by the ant, obtained from the equation (3):

$$Ppick(\text{Oi}) = \left(\text{kl} + \boldsymbol{j}(0\text{i})\right)^2, \tag{3}$$

where k1 is a threshold to control the probability of picking an object. It is worthy to mention that threshold k1 is kept fixed during progress of ACC. The probability that a loaded ant lays down its object is obtained by equation (4):

$$p_{drop}(o_i) = \begin{matrix} 2 \times f(o_i) \text{ if } f(o_i) > k_2 \\ 1 \text{ if } f(o_i) \leq k_2 \end{matrix}, \quad (4)$$

where k_2 is a threshold to control the probability of dropping an object. It is worthy to mention that threshold k_2 is also kept fixed during progress of ACC. Similarity measure, speed parameter, local density and short-term memory are described in the following.

2.1. **Perception Area.** Number of rooms observed by an ant in the two-dimensional area is a square of size $s \times s$ around the ant's position. Parameter s is considered as one of the effective factors controlling the overall similarity measure, and consequently the performance and the computational cost of the algorithm. If parameter s is set to a large value, it will cause the rapid formation of clusters and therefore generally fewer developed clusters. If parameter s is set to a small value, it will cause the slower formation of clusters and therefore the number of developed clusters will be larger. It is worthy to mention that proper selection of parameter s is a very important factor in the whole ACC algorithm. While selecting a large value can cause premature convergence of the algorithm, selecting a small value will also cause slow convergence of the algorithm.

2.2. **Similarity Scaling Factor.** Scaling parameter value α is defined in the interval (0, 1]. If α is large, then the similarities between objects will be increased, so it is easier for the loaded ants to lay down their objects and more difficult for the unloaded ants to lift the objects. So if α is large, fewer clusters are formed and it will be highly likely that really well-ordered clusters will not have the chance to be formed. If α is small, the similarities between objects will be reduced, so it is easier for the unloaded ants to pick up objects and more difficult for the loaded ants to remove their objects. So if α is small, many really well-shaped clusters will be created. On this basis, the appropriate setting of parameter α is very important and should not be data independent.

2.3. **Speed Parameter.** Speed parameter v can uniformly be selected form range $[1, v_{max}]$. Rate of picking-up or droping-off actions can be affected by the speed parameter. If v is large, few rough clusters can irregularly be formed on a large scale view. If v is small, then many dense clusters can precisely be formed on a small scale view. The speed parameter is a critical factor for the speed of convergence. An appropriate setting of speed parameter v can cause faster convergence of ACC algorithm.

2.4. **Short-Term Memory.** Each ant can remember the original real features and the virtual defined two-dimensional features of the last q objects it has dropped off. Whenever an ant takes an object it will search its short-term memory to find out which object(s) in the short-term memory is similar to the current object. If an object in its memory is similar enough to satisfy a similarity threshold, it will jump to the position of the similar object, hoping the current object will be dropped off near the location of the similar object. If there is no object in its memory satisfying the similarity threshold, it will not jump to any position; it will only hold the object and wander. Using short-term memory helps the ACC algorithm to prevent the objects that originally belong to the same cluster to be spitted in different clusters.

2.5. **Drawbacks of Original Ant Colony Clustering Algorithm.** The original ACC algorithm presented in this section suffers from two major drawbacks. First many clusters are produced in the virtual two-dimensional space and it is hard and very time-consuming to merge them and this work is inappropriate. The second drawback arises where the density detector is the sole measure based on that the clusters are formed in the local similar objects. But it fails to detect their dissimilarity properly. So a cluster without a significant between-object variance may not break into some smaller clusters. It may result in forming the wrong big clusters including some real smaller clusters provided the boundary objects of the smaller clusters are similar. It is because the probabilities of dropping-off or picking-up actions are dependent only

on density. So provided that the boundary objects of the smaller clusters are similar, they will be placed near to each other and the other objects will be also placed near to them gradually. Finally those small clusters form a big cluster, and there is no further mechanism to break the big cluster into smaller clusters. So there are some changes on the original algorithm to handle the mention drawbacks.

2.6. **Entropy Measure of Local Area.** We combine the information entropy and the mean similarity and consider their combination as a new metric. Incorporating the new metric into the existing models is employed in order to detect rough areas of spatial clusters, dense clusters and troubled borders of the clusters that are wrongly merged.

Shannon information entropy has been widely used in many areas to measure the uncertainty of a specified event or the impurity of an arbitrary collection of samples. Consider a discrete random variable X, with N possible values x_1, x_2, ..., x_N with probabilities $p(x_1)$, $p(x_2)$, ..., $p(x_N)$. Entropy of the discrete random variable X is obtained using equation (5):

$$H(X) = -\sum_{i=1}^{N} (p(x_i) \lg(p(x_i))). \tag{5}$$

Similarity degree between each pair of objects can be expressed as a probability that the two belong to the same cluster. Based on *Shannon* information entropy, each ant can compute the impurity of the objects observed in a local area L to determine if the object o_i in the center of the local area L has a high entropy value with group of object o_j in the local area L. Each ant can compute the local area entropy using equation (6):

$$E(L/o_i) = -\sum_{o_j \in Neigh_{s\times s(r)}} p_{ij} \times \frac{\lg p_{ij}}{\lg /Neigh_{s\times s(r)}/}, \tag{6}$$

where the probability p_{ij} indicates that we have a decisive opinion about central object o_i considering a local area object o_j in its local area L. The probability p_{ij} is obtained according to equation (7):

$$p_{ij} = \frac{2 \times /D(o_i, o_j)/}{n}, \tag{7}$$

where n ($n = /Neigh_{s\times s}(r)/$) is the number of neighbors. Distance function $D(o_i, o_j)$ between each pair of objects is measured according to equation (8):

$$D(o_i, o_j) = \frac{2 \times d(o_i, o_j)}{norm(o_i)} - 0.5, \tag{8}$$

where $d(o_i, o_j)$ is Euclidian distance defined by equation (2), and $norm(o_i)$ is defined as the maximum distance of object o_i with its neighbors. It is calculated according to equation (9):

$$norm(o_i) = \max_{o_j \in Neigh_{s\times s}(r)} d(o_i, o_j). \tag{9}$$

Now the function $H(L/o_i)$ is defined as equation (10):

$$H(L/o_i) = 1 - E(L/o_i). \tag{10}$$

Three examples of local area objects on a $3 \times 3 (= 9)$ neighborhood are depicted in Figure 2. Different classes with different colors are displayed.

When the data objects in the local area L and central object of the local area L exactly belong to a same cluster, i.e. their distances are almost uniform and low values, such as the shape or the form depicted by the left rectangle of Figure 2, uncertainty is low and $H(L/o_i)$ is far from one and consequently near to 0. When the data objects in the local area L and central object of the local area L belong to some completely different separate clusters, i.e. their distances are

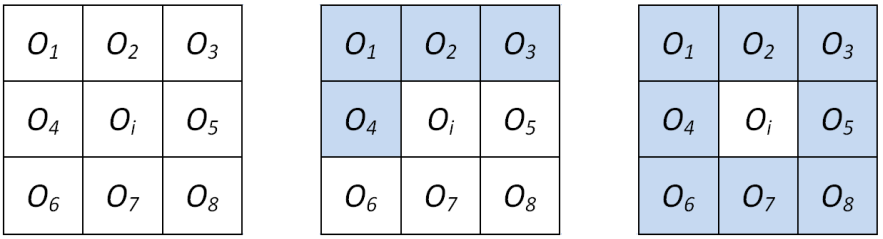

Figure 2. Three examples of local area objects.

almost uniform and high values, such as the shape or the form depicted by the right rectangle of Figure 2, uncertainty is again low and $H(L/o_i)$ is far from one and consequently near to 0. But in the cases of the form depicted by the middle rectangle of Figure 2 where some data objects in the local area L and central object of the local area L exactly belong to a same cluster and some others does not, i.e. the distances are not uniform, the uncertainty is high and $H(L/o_i)$ is far from 0 and consequently close to 1. So the function $H(L/o_i)$ can provide ants with a metric that its high value indicates the current position is a boundary area and its low value indicates the current position is not a boundary area.

In ant-based clustering, two types of pheromones are employed: (a) cluster pheromone and (b) object pheromone. Cluster pheromone guides the loaded ants to valid clusters for a possible successful dropping-off action. Object pheromone guides the unloaded ants to free object for a possible successful picking-up action.

Each loaded ant deposits some cluster pheromone on the current position and positions of its neighbors after a successful dropping-off of an object to guide other ants for a place to unload their objects. The cluster pheromone intensity deposited in location $\boldsymbol{j}$, by m ants in the colony at time t is calculated by the equation (11):

$$rc_j(t) = \sum_{a=1}^{m} [\mu^{(t-t_a^1)} \times C \times E(L/o_j)], \tag{11}$$

where C is cluster pheromone constant, t_a^1 is the time step at that ath cluster pheromone is deposited at position $\boldsymbol{j}$, and μ is evaporation coefficient. On the other hand, an unloaded ant deposits some object pheromone after a successful picking-up of an object to guide other agents for a place to take the objects. The object pheromone intensity deposited in location $\boldsymbol{j}$, by m ants in the colony at time t is calculated by the equation (12):

$$ro_j(t) = \sum_{a=1}^{m} [\mu^{(t-t_a^2)} \times O \times H(L/o_j)], \tag{12}$$

where O is object pheromone constant, and t_a^2 is the time step at that ath object pheromone is deposited at position $\boldsymbol{j}$. Transmission probabilities of an unloaded ant based on that the ant moves from the current location $\boldsymbol{i}$ to next location $\boldsymbol{j}$ from its neighborhood can be calculated according to equation (13):

$$P_j^u(t) = \begin{cases} \frac{1}{w} & \text{if } \bigwedge_{j=1}^{w} ro_j(t) = 0 \ \forall \boldsymbol{j} \in N_{dir} \\ \frac{ro_j(t)}{\sum_{j=1}^{n} ro_j(t)} & \text{otherwise} \end{cases}, \tag{13}$$

where N_{dir} is the set of possible w actions (possible w directions along with ant can move) from current position $\boldsymbol{i}$. Transmission probabilities of a loaded ant based on that the ant moves from the current location $\boldsymbol{i}$ to next location $\boldsymbol{j}$ from its neighborhood can be calculated according to equation (14):

```
Input:
  QD, it1; q, AntNum, Data, 0, C, k 1, k2, v'''''' pe1·iod, ttu·, st, distributions ofv, a and 11
initializing pa1·amete·using dish·ibutions ofv, a and 11;
fo1·each ant a
        place 1·andom a in a position not occupied by othe1·ants in a plane QD*QD;
end;
fm·each object a
      place random o in a position not occupied by other objects in the plane QU*QU;
end;
success(1:ant)=O;
faiim·e(1:ant)=O;
fo1·1=1:itJ·
        for each ant a
                g=select a 1·andom numbe1·unifonnly f1·om 1·ange [0,1];
                r=position(a)
                if(loaded(a) and (is_empty(r)))
                        if(,o<pd'op) o=dmp(o);
                                pul(r,u);
                                save(o,r,q);
                        end;
                elseif(not (loaded(a) or (is_empty(••))))
                        if(g<ppc)
                                o=•·emove(r);
                                pick_up(a,o);
                                sem·ch_and_jump(a,o);
                                success(a)=success(a)+1;
                        else
                                faiim·e(a)=faiJme(a)+1;
                        end;
                end;
                else
                        wandei·(a,,,Nd1, ); // conside1 ing the defined phHomone
                end;
        end;
        if( t mod pel'iod==O)
                for each ant a
                        if(success(a)/(failu1·e(a)+success(a))>thr)
                                a.(a)=a.(a)+st;
                        else
                                a(a)=a(a)-st;
                        end;
                end;
        end;
end;
```

FIGURE 3. Pseudo code of modified ant colony clustering algorithm.

$$P_j^l(t) = \begin{cases} \frac{1}{w} & \text{if } \sum_{j=1}^{n} rc_j(t) = 0 \ \forall j \in Ndir \\ \frac{rc_j(t)}{\sum_{j=1}^{n} rc_j(t)} & \text{otherwise} \end{cases} \tag{14}$$

2.7. **Modified Ant Colony Clustering.** By incorporating the information entropy and the mean similarity into a new metric and using it in the original ACC algorithm, we want to reinforce ACC. After all the above mentioned modification, the pseudo code of ACC algorithm is presented in Figure 3.

Figure 4 shows an exemplary running of the modified ant colony algorithm. In Figure 4 the final result of modified ant colony clustering algorithm (MACCA) over Iris dataset is presented.

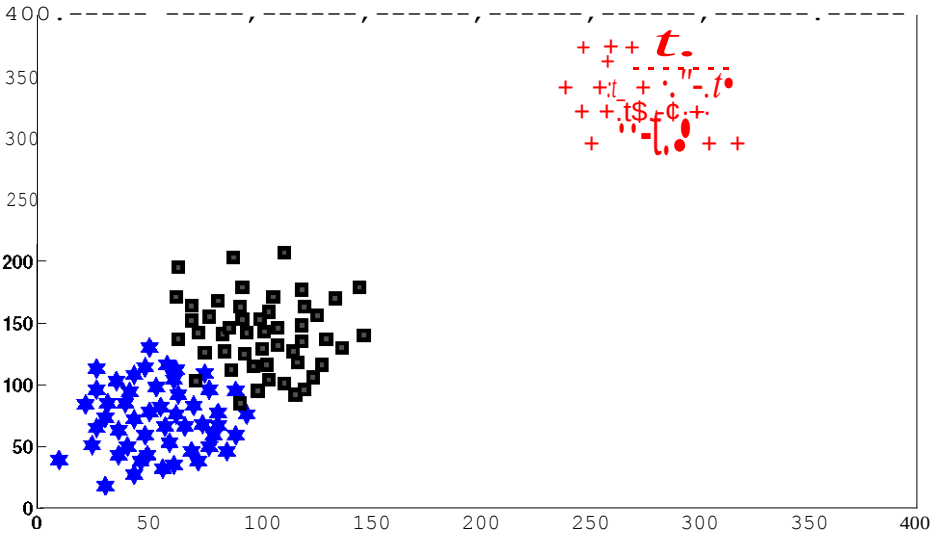

FIGURE 4. Final result of modified ant colony clustering algorithm over Iris dataset.

TABLE 1. The details of used parameters for the modified ant colony clustering algorithm

Parameter Name	Value
QD	400
$tmax$	5000000
q	20
$AntNum$	240
O	1
C	1
$k1$	0.1
$k2$	0.3
$Vmax$	150
period parameter	2000
thr	0.9
μ	0.95
st	0.01
a is uniformly extracted from the interval	[0.1,1]
v is uniformly extracted from the interval	[1,Vmax]

It is worthy to mention that the quantization degree parameter (QD), the algorithm maximum iteration $tmax$, queue size parameter (q), ant number parameter ($AntNum$), object pheromone parameter (O), cluster pheromone parameter (C), k_1 parameter, k_2 parameter, maximum speed parameter ($vmax$), period parameter, update parameter (thr) evaporation parameter μ and step of update for a parameter (st) are respectively set to 400, 5000000, 20, 240, 1, 1, 0.1, 0.3, 150, 2000, 0.9, 0.95 and 0.01 for reaching the result presented in Figure 4. Parameter o: for each ant is extracted uniformly from the range [0.1, 1]. Parameter v for each ant is also extracted uniformly from the range [1, $Vmax$]. The details of used parameters for the MACCA are presented in Table 1.

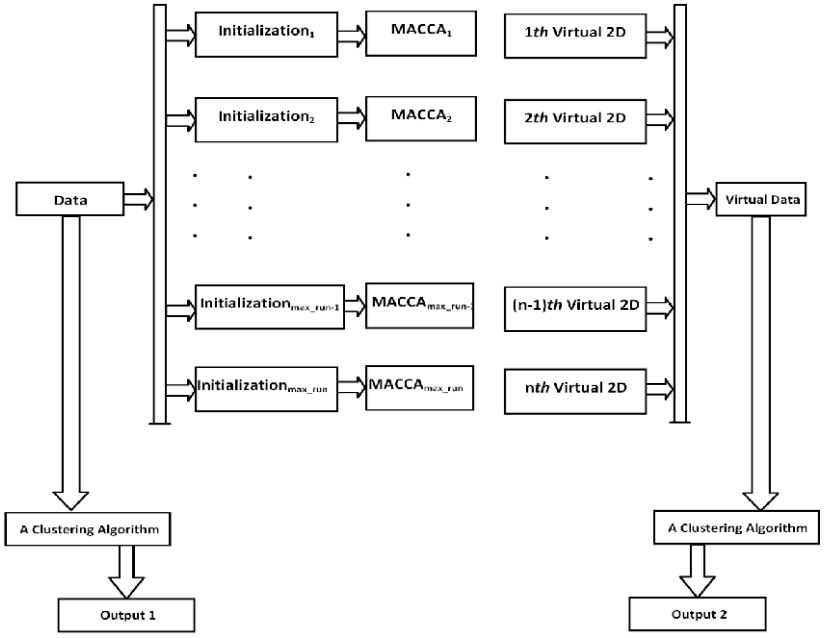

FIGURE 5. Proposed framework to cluster a dataset using ant colony clustering algorithm.

It is important to note that the result shown in Figure 4 is a well separated running of algorithm. So it is considered as a successful running of algorithm. The algorithm may also converge to a set of overlapping clusters in an unsuccessful running.

3. PROPOSED NEW SPACE DEFINED BY ANT COLONY ALGORITHM

Both ACC algorithm and MACCA have many parameters. Their effectiveness is questionable because they depends on many parameters. On a test data, these parameters should be well-tuned to obtain a desirable result. But how to define them in a real task does not clear. The proposed method lets the parameters be free to be changed, and compensates non-optimality of their parameters by the ensemble power.

Indeed the main idea behind the proposed clustering framework is the usage of ensemble learning concept in the field of ACC. The MACCA is very sensitive to the proper initialization of its parameters. If one employs a proper initialization for parameters of the MACCA, the final discovered clusters can be desirable. On the other hand, an ensemble needs diversity to be successful. It can be inferred that by several runs of MACCA with different initializations for parameters, we can reach several partitions that are very diverse. So we use an ensemble approach to overcome the problem of well-tuning of its parameters. The main contribution of the paper is illustrated in Figure 5.

As it is depicted in Figure 5, a dataset is fed to as many as max_run different MACCAs with the different initializations. By running each MACCA, an output in a 2-dimension plane, denoted by virtual 2-dimension, is produced. Therefore, we obtain max_run virtual 2-dimensions, one per each run of MACCA. Then by considering all these virtual 2-dimensions as a new space with $2 \times max\_run$ dimensions, we reach a new data space which is named *mapped* dataset. It means

that ith feature of *mapped* dataset will be the dimension x of the output of rth MACCA where $r = \lceil \frac{i}{2} \rfloor$, if i is odd; otherwise, it will be the the dimension y of the output of rth MACCA. We can finally employ a clustering algorithm on *mapped* dataset to extract the final partitioning.

So the proposed method have two main contributions: (a) proposing a method that bypasses well-tuning of parameters of the original ACC algorithm and MACCA, (b) proposing a framework to construct a diverse and suitable clustering ensemble.

4. Experimental Study

This section evaluates the result of applying the proposed clustering framework on some real datasets available at UCI repository [14]. The main metrics based on which a partitioning is evaluated are discussed in the first subsection of this section. The details of the used datasets are given in the subsequent section. Then the settings of experimentations are given. Finally the experimental results are presented.

4.1. **Evaluation Metric.** After producing the final partitioning, the most important issue is the method of its evaluation. The evaluation of a partitioning is a very important and challenging task due to the lack of supervisor. Here the normalized mutual information (NMI) between the output partitioning and real labels of the dataset is considered as the main evaluation metric of a partitioning [4, 15]. The output partitioning is the one that is obtained as the final partitioning of applying any clustering method on the dataset. It means that after applying a clustering algorithm on the dataset, the partitioning obtained by applying a clustering algorithm is output partitioning of that clustering algorithm. The true labels of the dataset can be used after termination of a clustering algorithm to evaluate how good clustering algorithm has done the clustering task [4, 15]. The NMI between two partitionings, P^a and P^b, is calculated based on equation (15):

$$NMI(P^a, P^b) = \frac{-2 \sum_{i=1}^{k_a} \sum_{j=1}^{k_b} n_{ij}^{ab} \log\left(\frac{n_{ij}^{ab} \times n}{n_i^a \times n_j^b}\right)}{\sum_{i=1}^{k_a} n_i^a \log\left(\frac{n_i^a}{n}\right) + \sum_{j=1}^{k_b} n_j^b \log\left(\frac{n_j^b}{n}\right)}, \tag{15}$$

where n is the total number of samples, and n_{ij}^{ab} denotes the number of shared patterns between clusters $C_i^a \in P^a$ and $C_j^b \in P^b$, and n_i^a is the number of patterns in the cluster i of partitioning P^a, and also n_j^b is the number of patterns in the cluster j of partition P^b.

Another alternative to evaluate a partition is the accuracy metric, provided that the number of clusters and their true assignments are known. To compute the final performance of a clustering algorithm in terms of accuracy, one can first re-label its output partitioning in such a way that has maximal matching with the ground true labels (the true labels of dataset) and then counting the percentage of the true classified samples. So the error rate can be determined after solving the correspondence problem between the labels of derived and known clusters [16]. The Hungarian algorithm is employed to solve the minimal weight bipartite matching problem. It has been shown that it can efficiently solve the label correspondence problem [17].

F-measure (F-score) is the last measure to evaluate a clustering algorithm. To compute the final performance of a clustering algorithm in terms of F-measure, after termination of the clustering algorithm, its output partitioning and true labels of dataset are fed to F-measure computation formula. Both the precision and recall measures are used to compute the F-measure. The F-measure can be interpreted as a weighted average of the precision and recall, where an it hits its best value at 1 and its worst score at 0 [18]. The F-measure formula used in the paper is based on equation (16):

Table 2. Details of used dataset

Dataset Name	number of records	number of features	number of classes
Image-Segmentation	210	19	7
Zoo	101	17	7
Thyroid	215	5	3
Soybean	683	35	4
Iris	150	4	3
Wine	178	13	3
HalfRing	300	2	2

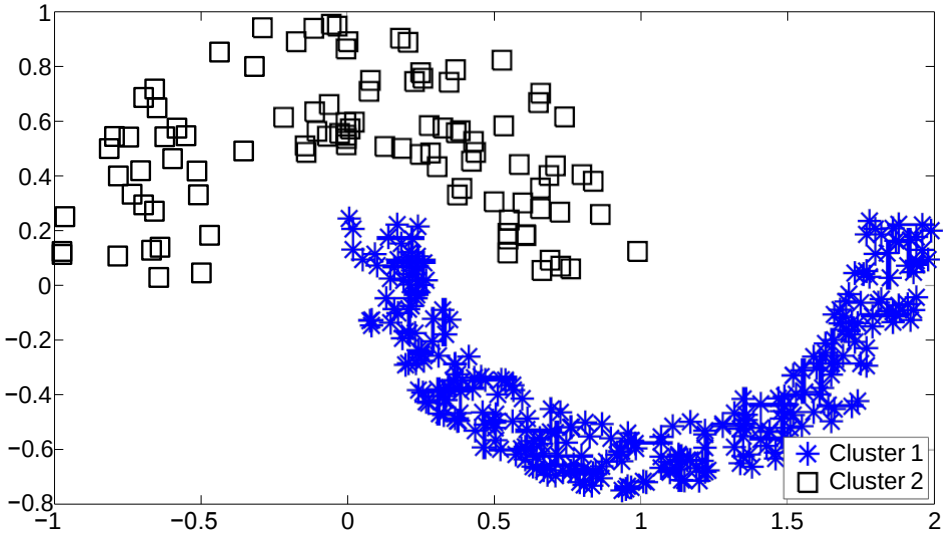

Figure 6. HalfRing dataset.

$$Fmeasure(P^a, P^b) = \max_{\Pi} \sum_{i=1}^{k_a} \frac{2 \times (\frac{n^{ab}_{i\Pi(i)}}{n^a_i} \times \frac{n^{ab}_{i\Pi(i)}}{n^a_{\Pi(i)}})}{n^a_i \times (\frac{n^{ab}_{i\Pi(i)}}{n^a_i} + \frac{n^{ab}_{i\Pi(i)}}{n^a_{\Pi(i)}})}, \tag{16}$$

where n is the total number of samples, and n^{ab}_{ij} denotes the number of the shared patterns between clusters $C^a_i \in P^a$ and $C^b_j \in P^b$, and n^a_i is the number of patterns in the cluster i of partitioning P^a, and n^b_j is the number of patterns in the cluster j of partition P^b, and also Π is a permutation of 1,..., k_b for relabeling of P_b.

It is worthy to mention that in all experimentations, NMI, F-measure and accuracy are scaled between [0-100] for convention, so they are always reported in range [0-100].

4.2. **Datasets.** The proposed method is examined over 6 different standard and one hand-made datasets. Brief information about the used datasets is available in Table 2. More information about them is available in [14].

The artificial HalfRing dataset is depicted in Figure 6. The HalfRing dataset is considered as one of the most challenging datasets for clustering algorithms.

4.3. **Experimental Settings.** The quantization degree parameter (QD) is a very important parameter in qualification of the final clustering. The high value QD directly increases the

Table 3. Experimental results in terms of accuracy, O1 stands for output 1 and O2 stands for output 2

Dataset Name	FCM O1	FCM O1	FCM O1	FCM O2	FCM O2	FCM O2
-	Accuracy	NMI	F-Measure	Accuracy	NMI	F-Measure
Image-Segmentation	52.27	38.83	59.42	54.39	40.28	62.29
Zoo	80.08	79.09	84.68	81.12	81.24	89.10
Thyroid	83.73	50.23	87.91	87.94	59.76	91.09
Soybean	90.10	69.50	93.38	94.34	80.30	95.97
Iris	90.11	65.67	96.11	93.13	75.22	96.94
Wine	74.71	33.12	72.58	76.47	35.96	76.44
HalfRing	72.20	33.04	80.01	93.71	49.78	94.76

time burden. Time order of the proposed algorithm is related to QD quadratically. So if it is set very high, the algorithm may fail to find a solution. On the other hand, selecting a low value for QD results in limiting space for the cluster points to be appropriately shaped. The queue size parameter (q) as it has been mentioned before should be a value that covers some of the main dense locations in the space. Experimentally we found it that choosing a value 20 is always more than necessity. As it relates to the time order of the proposed algorithm linearly so increasing it may cause some problems. Therefore we use a value no more than 20 for parameter q. The object pheromone parameter (O), the cluster pheromone parameter (C), k_1 parameter, k_2 parameter, maximum speed parameter (v_{max}), period parameter, update parameter (thr) evaporation parameter μ and step of update for α parameter (st) are set to the values from previous papers, because their concepts have not changed during their usage in this paper.

The quantization degree parameter (QD), the algorithm maximum iteration t_{max}, queue size parameter (q), ant number parameter ($AntNum$), object pheromone parameter (O), cluster pheromone parameter (C), k_1 parameter, k_2 parameter, maximum speed parameter (v_{max}), period parameter, update parameter (thr) evaporation parameter μ and step of update for α parameter (st) are respectively set to 400, 5000000, 20, 240, 1, 1, 0.1, 0.3, 150, 2000, 0.9, 0.95 and 0.01 in all experimentations as before. Parameter α for each ant is extracted uniformly from the range [0.1, 1]. Parameter v for each ant is extracted uniformly from the range [1, v_{max}]. Fuzzy k-means (c-means) is employed as the base clustering algorithm to perform final clustering over the original dataset and the *mapped* dataset. Parameter max_run is set to 30 in all experimentations. So the *mapped* dataset has 60 (virtual) features. Number of real cluster in each dataset is given to fuzzy k-means clustering algorithm in all experimentations. The details of used parameters for the MACCA are still the same in Table 1. Although MACCA have many parameters that should be well-tuned to obtain a desirable result, as we have mentioned before their values have no side effect in the proposed clustering ensemble framework based on MACCA. The proposed clustering ensemble framework based on MACCA compensates the non-optimality of the parameters used in MACCA by its ensemble power.

As it is inferred from the Table 3, the new defined feature space (the *mapped* dataset) is better clustered by a base clustering algorithm rather than the original space.

4.4. **Experimental Results.** Table 3 shows the performance of the fuzzy clustering in both original and the *mapped* spaces in terms of accuracy, NMI and F-measure. All experiments are reported by averaging over 10 independent runs of algorithms. It means to reach F-measure of the proposed clustering framework on dataset *Iris*, 10 different independent runs of the proposed clustering framework over *Iris* are done. Then 10 different output clusterings are produced. Feeding any of them to computation formula of F-measure, we reach 10 F-measure values. Averaged 10 F-measures has been reported in the Table 3 as F-measure of the proposed clustering framework over *Iris* dataset. The same scenario is done for NMI and accuracy.

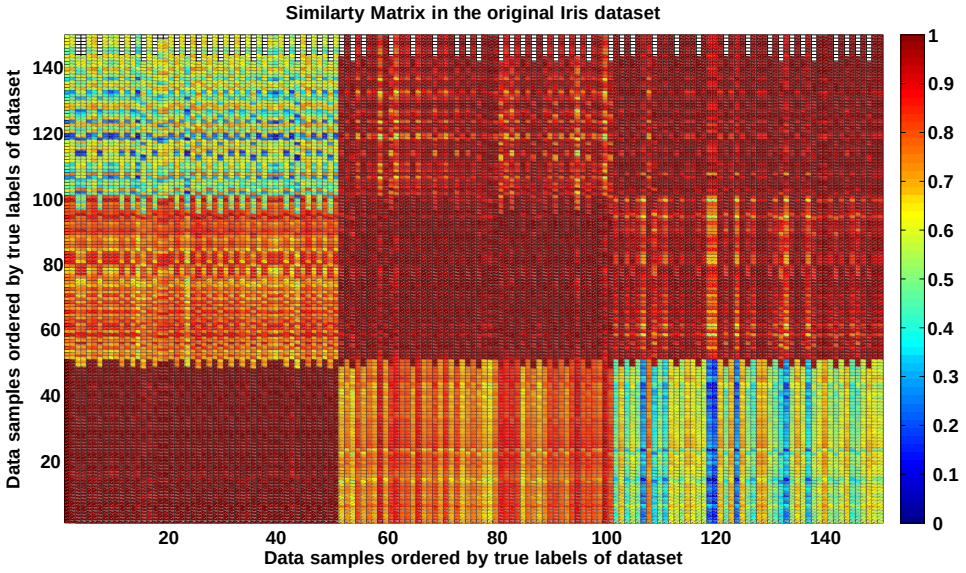

Figure 7. Similarity matrix of data points in original *Iris* dataset.

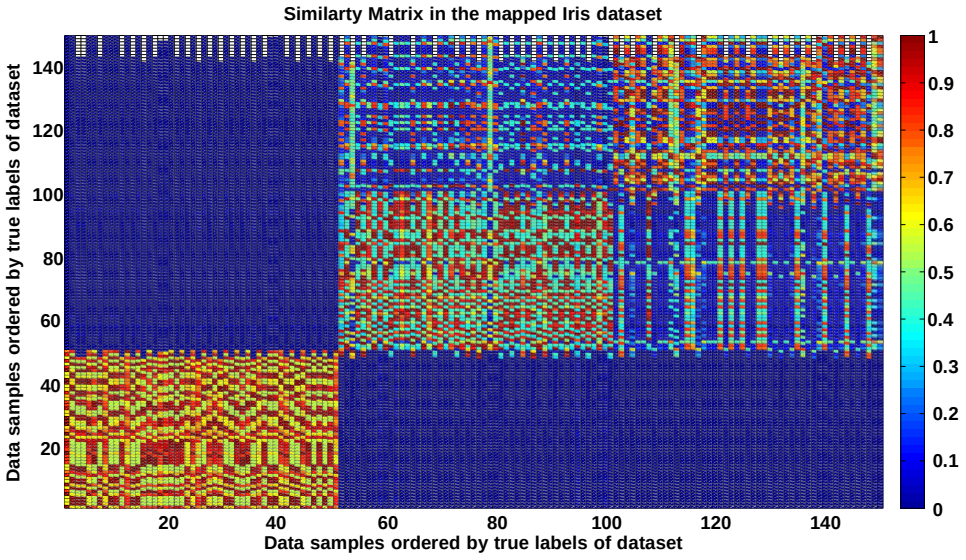

Figure 8. Similarity Matrix of data points in mapped *Iris* dataset.

To demonstrate the effectiveness of the proposed clustering framework, consider Figure 7 and Figure 8. In Figure 7 and Figure 8, a more comprehensive example of similarity matrices of data points in the original and *mapped* (by running of a number of MACCAs) *Iris* dataset are presented. Figure 7 shows the similarity matrix of data points in original *Iris* dataset. It is worthy to mention that the order of data points in each presented similarity matrix is based on real labels of the dataset for better understanding.

Figure 8 shows the similarity matrix of data points in the *mapped Iris* dataset. As it is inferred from Figure 7 and Figure 8 comparatively, the similarity matrix in the *mapped Iris* dataset is more discriminative than the similarity matrix in the original *Iris* dataset.

To generalize our conclusion, the same experimentations have been repeated and presented over *Wine* dataset. Figure 9 shows the similarity matrix of data points in original *Wine* dataset.

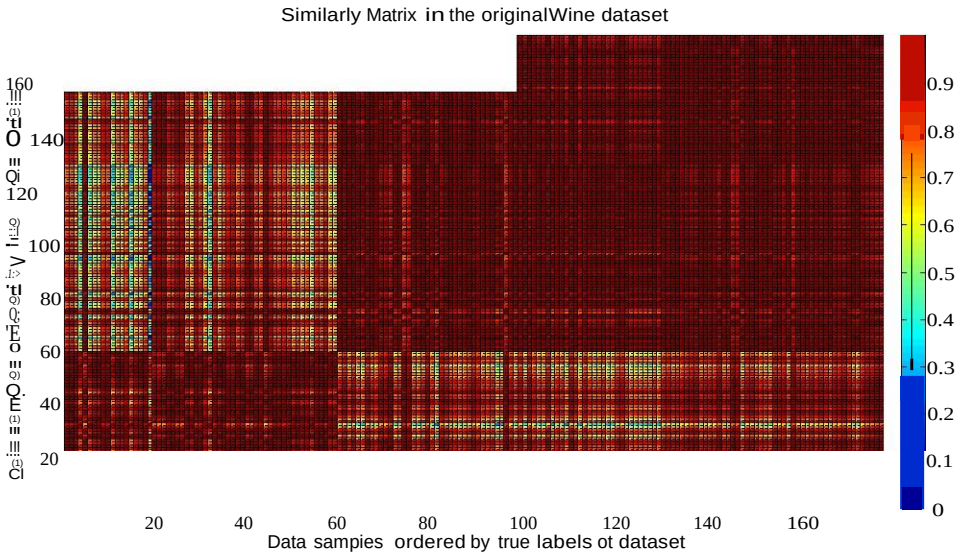

FIGURE 9. Similarity matrix of data points in original *Wine* dataset.

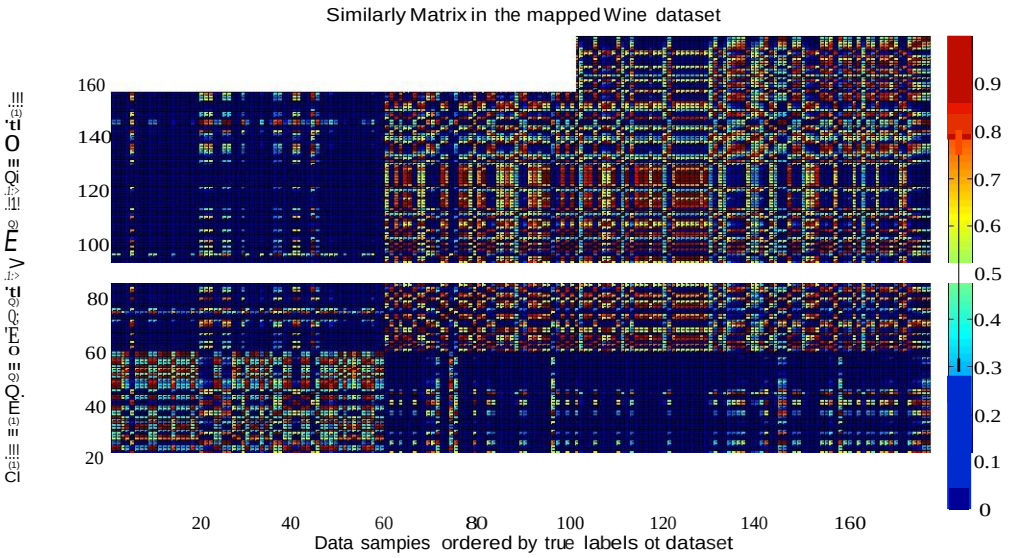

FIGURE 10. Similarity Matrix of data points in mapped *Wine* dat aset.

Figure 10 also shows the similarity matrix of data points in the *mapped Wine* dataset. As it can be again inferred from Figure 9 and Figure 10 comparatively, the similarity matrix in the *mapped Wine* dat aset is more discriminative than the similarity matrix in the original *Wine* dataset.

5. CONCLUSIO N AND FUTURE WORK

All *ACC* algorithms have many parameters. Their effectiveness is questionable because they depends on many parameters that should be well-tuned to obtain a desirable result. But how to define them in a real task does not clear. The proposed method lets the parameters be free to be changed, and compensates non-optimality of their parameters by the ensem ble power. This

paper has proposed a method that bypasses well-tuning of parameters of ACC algorithms. It has also proposed a framework to construct a diverse and suitable clustering ensemble. The paper have also brought two successful concepts in the field of clustering: (a) ensemble concept and (b) swarm concept. Then based on them a new clustering ensemble framework has been proposed. Indeed different runnings of MACCA with different initializations result in a number of diverse partitionings. In the proposed framework we use a type of MACCA and produce an intermediate space considering their outputs totally as a defined virtual space. After producing the virtual space we employ a base clustering algorithm to obtain the consensus partitioning.

The experiments show that the clustering task in the defined data space outperforms in comparison with the clustering in original data space. It is concluded that the new defined feature space is better clustered rather than the original space. As it is shown, the similarity matrix in the mapped dataset is more discriminative than the similarity matrix in the original dataset.

As a future work, a weighting mechanism to participate the output of each ACC based on its fitness can be examined. The effect of niching and elitism mechanisms can be studied. One can also turn to other types of swarm based clustering algorithms.

References

[1] Faceli, K., Marcilio, C.P., Souto, D. Multi-objective Clustering Ensemble, International Conference on Hybrid
Intelligent Systems, V. 4, N. 3, pp. 145-156, 2006.

[2] Maheshkumar S., Gursel S. Application of machine learning algorithms to KDD intrusion detection dataset within misuse detection context, Innternational Conference on Machine Learning, Models, Technologies and Applications, CSREA Press, pp. 209-215, 2003.

[3] Roth, V., Lange, T., Braun, M., Buhmann, J. A Resampling Approach to Cluster Validation, Innternational
Conference on Computational Statistics, pp. 123-128, 2002.

[4] Strehl A., Ghosh J. Cluster ensembles - a knowledge reuse framework for combining multiple partitions,
Journal of Machine Learning Research, V. 3, N. 1, pp.583-617, 2002.

[5] Ayad, H.G., Kamel, M.S., Cumulative Voting Consensus Method for Partitions with a Variable Number of
Clusters, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, V. 30, N. 1, pp. 160-173, 2008.

[6] Parvin, H., Beigi, A. Clustering Ensemble Framework via Ant Colony, *Lecture Note on Computer Science*, V. 7095, N. 1, pp. 153-164, 2011.

[7] Kuncheva L.I. *Combining Pattern Classifiers, Methods and Algorithms*, New York: Wiley,
2005. [8] Kennedy, J., Russell, S., *Swarm Intelligence*, Morgan Kaufmann, San Francisco,
2001.

[9] Deneubourg, J.L., Goss, S., Franks, N., Sendova-Franks, A., Detrain, C., Chretien, L. The dynamics of collective sorting robot-like ants and ant-like robots, International conference on simulation of adaptive behavior: from animals to animates. MIT Press, pp. 356-363, 1991.

[10] Lumer, E.D. and Faieta, B.: Diversity and adaptation in populations of clustering ants, International confer- ence on simulation of adaptive behavior: from animals to animates, MIT Press, Cambridge, MA, pp. 501-508,
1994.

[11] Tsang, C.H., Kwong, S. Ant Colony Clustering and Feature Extraction for Anomaly Intrusion Detection,
Studies in Computational Intelligence, V. 34, N. 1, pp. 101-123, 2006.

[12] Azimi, J., Cull, P., Fern, X. Clustering Ensembles Using Ants Algorithm, *Lecture Note on Computer Science*, V. 5601, N. 1, pp. 295-304, 2009.

[13] Liu, B., Pan, J., McKay, R.I. Incremental Clustering Based on Swarm Intelligence, *Lecture Note on Computer*
Science, V. 4247, N. 1, pp. 189-196, 2006.

[14] Newman, C.B.D.J., Hettich, S., Merz, C. UCI repository of machine learning databases, 1998, http://www.ics.uci.edu/mlearn/MLSummary.html.

[15] Alizadeh, H., Minaei-Bidgoli, B., Parvin, H. A New Criterion for Clusters Validation. *IFIP Advances in*
Information and Communication Technology, V. 364, N. 1, pp. 240-246, 2011.

[16] Minaei-Bidgoli, B., Parvin, H., Alinejad-Rokny, H., Alizadeh, H., Punch, W.F. Effects of resampling method and adaptation on clustering ensemble efficacy, *Artificial Intelligence Review*, 2011, doi: 10.1007/s10462-011-
9295-x.

[17] Munkres, J. Algorithms for the Assignment and Transportation Problems, *Journal of the Society for Indus- trial and Applied Mathematics*, V. 5, N. 1, pp. 32-38, 1957.

[18] Marxer, R., Holonowicz, P., Purwins, H., Hazan, A. Dynamical hierarchical self-organization of harmonic, motivic, and pitch categories. Music, Brain and Cognition, Part 2: Models of Sound and Cognition, NIPS, Vancouver, Canada, 2007.