Received: December 16, 2020. Revised: January 25, 2021. 421

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

A New Hybrid Clustering Method of Binary Differential Evolution and Marine Predators Algorithm for Multi-omics Datasets

Mohamed Ghoneimy[1]* Hesham A. Hassan[2] Emad Nabil[2,3]

[1]*Faculty of Information Technology, MUST University, 6th of October City, Giza, Egypt*
[2]*Faculty of Computers and Artificial Intelligence, Cairo University, Giza, Egypt*
[3]*Faculty of Computer and Information Systems, Islamic University of Madinah, Madinah, Saudi Arabia*
* Corresponding author's Email: Mohamed.ghoneimy@must.edu.eg

Abstract: Clustering of biological datasets proved to reveal a lot of significant insights into the medical and biological research. It is an important step towards drug design, vaccine discovery, disease diagnosis, and more. The current trend in biological and medical research is to combine more than one dataset, referred to as multi-omics, related to a specific problem, then perform the clustering or the analysis. The insights we gain for a particular biological problem or disease are double using multi-omics rather than using one dataset. It is like investigating a problem from many dimensions rather than using one dimension. On the other hand, the difficulty of clustering is increased. Another tricky problem in data clustering is determining the best number of clusters used by a clustering algorithm. Due to the big success of metaheuristics in solving the automatic clustering problems, we propose in this paper a new hybrid method that utilizes two powerful metaheuristics algorithms, the Binary Differential Evolution and Marine Predators Algorithm, to perform automatic clustering on multi-omics datasets. Our proposal's performance is investigated upon eight multi-omics datasets from TCGA, and it is compared with four recent and powerful metaheuristics. The used performance metrics are clustering quality and execution time. The experimental results show that the proposed algorithm not only outperformed its competitors in terms of clustering quality, it also only needed a third of the execution time that of its fastest competitor. Moreover, the statistical analysis shows that the obtained results are statistically significant. Accordingly, the proposed method can be considered as an efficient clustering method for multi-omics datasets.

Keywords: Clustering, Automatic clustering, Differential evolution, MPA, Metaheuristics, Nature-inspired metaheuristics, Molecular-level interaction, Multi-omics.

1. Introduction

The rapid development in high throughput methods produced huge data types such as DNA methylation, DNA genome sequence, and RNA expression, each of them is called omic. The analysis of multi-omics datasets is beneficial for the following reasons. First, it reduces the effect of noise on results. Seconds, using omics from different molecular aspect levels such as genomic and epigenomic can show different aspects of patients. Third, even using omics from the same level, such as mutation and copy number, can reveal the omics' different aspects [1].

It is necessary to develop new computational methods to analyze these datasets. Clustering is the process of discovering the natural grouping of records according to their similarities. Clustering is a fundamental process in analysis, and it is often used as the first process of data analysis. Clustering is essential for medical research as it is used to discover the co-regulated genes and new grouping of patients based on genetic similarity. Many clustering methods need to determine the number of clusters before starting, which can be challenging to obtain in a multi-omics problem [2, 3]. This problem can be solved using automatic clustering, which needs to determine the minimum and the maximum number of clusters.

The two main automatic clustering tasks are determining the optimal number of clusters and determining the appropriate cluster for each object.

Finding the optimum solution for the clustering problem is an NP-hard problem [4]. Many publications tackled the problem of automatic clustering using metaheuristic algorithms [5–7]. A survey about solving the automatic clustering problem using nature-inspired metaheuristics is presented in [8].

To the best of our knowledge, the problem of automatic clustering for multi-omics datasets is not solved yet. In this paper, we propose a new hybrid method combining modified Binary Differential Evolution (BDE) [9] and Marine Predators Algorithm (MPA) [10] called DEMP to solve the automatic clustering problem for multi-omics datasets. DEMP uses at the first stage BDE, and it uses MPA at the second stage. In addition, we propose a modified BDE algorithm to fit our method. In the first stage, the DEMP method tries to reach the right number of clusters and reasonable clustering solutions. In contrast, in the second stage, it tries to reach the optimal solution to the automatic clustering problem. The DEMP method is benchmarked by eight multi-omics datasets that are available on The Cancer Genome Atlas (TCGA).

The rest of this paper is organized as follows: in the second section we introduce the related works for metaheuristics methods which solve an automatic clustering problem. The third section presents our new hybrid clustering method DEMP. The fourth section displays the experimental results. Next, the fifth section displays the result's discussions. Finally, the sixth section presents the conclusion of this paper.

2. Related work

In this section, we will show and discuss the related works to automatic clustering methods using metaheuristics. Any metaheuristic needs to determine an encoding to solve the automatic clustering problem. The encoding schema is vital for determining the metaheuristic search space, affecting both the result's quality and the execution time. Automatic clustering methods that use metaheuristics can be divided in terms of encoding into three categories, which are binary, integer, and real encoding. Because the number of clusters is not known in advance, the encoding scheme must be designed to adapt to the clusters number change in the specified pre-determined range [k_{min}, k_{max}].

In the binary encoding scheme, a binary string of length equal to *N* (where *N* equals the number of objects/patients in the data set) is used to represent each clustering solution. Furthermore, each object/patient in the data set is represented as a position in that binary string. In the binary string, the object/patient is considered a cluster centroid only if the corresponding position equals one, as shown in Fig. 1. A genetic algorithm for automatic clustering that uses binary encoding is proposed in [11]. [12] proposed a hierarchical evolutionary algorithm that solves the automatic clustering problem for medical images using binary encoding. The clustering solution's binary string's length is frequently short because the number of patients in multi-omics datasets is relatively small, leading to a smaller search space with faster convergence. On the other hand, the metaheuristic that uses the binary encoding chooses the potential centroids from the data set's objects/patients, so obtaining a good clustering solution using the binary encoding scheme depends on the dataset. Therefore, metaheuristics do not often use the binary encoding on clustering.

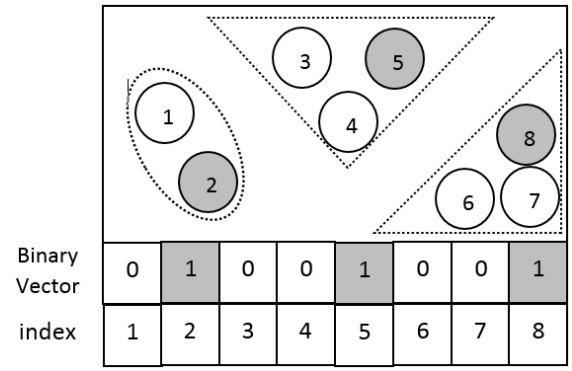

Figure. 1 Clustering solution representation in binary encoding

In the integer encoding scheme, an integer array of length equal to *N* is used to represent each clustering solution. Each object/patient in the data set is represented as an integer array position, which equals an integer value in the range [1:k_{max}]. All objects/patients that have the same integer value belong to the same cluster as shown in Fig. 2. One of the drawbacks of this encoding is repetition, as the same solution can have more than one representation, which leads to doubling the clustering search space. For example, the following representations {1,1,2,2,2}, {2,2,1,1,1} represent the same clustering solution. [13] proposed a genetic algorithm for automatic clustering using integer encoding. Proposed improvements that enhance the efficiency of genetic algorithm in solving the automatic clustering problem is published in [14]. A Bacterial Evolutionary Algorithm is a metaheuristic that uses integer encoding to solve the automatic clustering problem [15]. [16, 17] are two automatic clustering methods that use metaheuristics with integer encoding for the microarray datasets (single omic).

In the real encoding scheme, each clustering solution is represented as a set of centroids whose

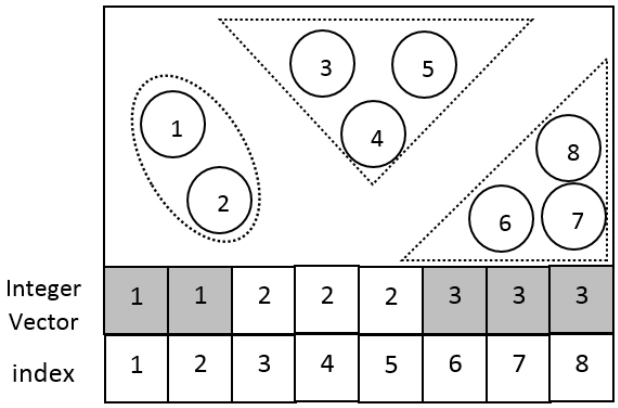

Figure. 2 Clustering solution representation in integer encoding

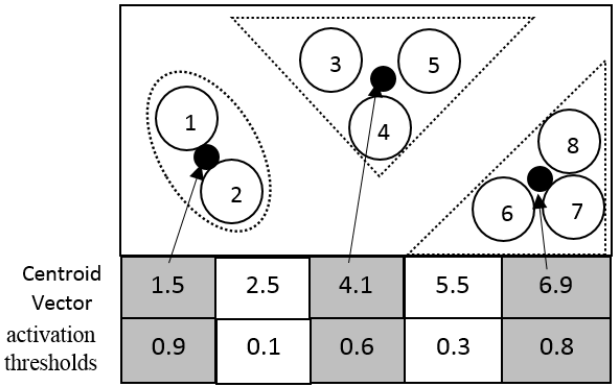

Figure. 3 Clustering solution representation in centroid-base real encoding

number is K_{max}. Due to representing clusters number varying, activation thresholds are commonly used to determine which centroid is active in a particular clustering solution, as shown in Fig. 3. The metaheuristics widely use real encoding in automatic clustering. Many Genetic algorithm versions used real encoding in solving the automatic clustering problem such as [18, 19]. Two particle swarm optimization (PSO) metaheuristics also used the real encoding [20, 21]. Invasive weed optimization (IWO) metaheuristics chose real encoding in [22]. An evolutionary programming-based clustering approach based on real encoding is presented in [23]. [24] proposed an automatic clustering method based on a clone selection algorithm that uses real encoding. [25] presented a comparative performance experiment of five metaheuristics in solving the automatic clustering problem. These metaheuristics are firefly algorithm (FA), differential evolution (DE), genetic algorithm (GA), IWO, and PSO. All metaheuristics that are chosen in the mentioned experiment are based on real encoding. Recently, a modified DE metaheuristic used the real encoding to solve the clustering problem [26]. Some powerful modern metaheuristics have not been used to solve the automatic clustering before due to their recency. These metaheuristics can compete with the current metaheuristics in this field. Such as Equilibrium Optimizer (EO) [27], MPA, and Slime mould algorithm (SMA) [28]. SMA is nature-inspired metaheuristic that mainly inspired by slime mould behaviour. It uses weights to mimic the negative and positive oscillator's feedback through searching for the food. It achieved superior results in terms of average fitness value because it has a good balance between explorations and exploitations. EO is a physics-based metaheuristic that mainly inspired by the mass balance equation for a control volume. It has high exploitative and exploratory search techniques to solutions' random modifications. This metaheuristic is known for its speed in reaching optimum or near-optimum solutions. Despite the metaheuristics, which uses real encoding can find a much better solution to the clustering problem for multi-omics datasets, they need higher execution time because of the large number of decision variables. The increase in the number of decision variables because each centroid is represented as a real value array of length equal to the number of features in the multi-omics dataset, which are a huge number.

3. The proposed method

This section presents our proposed hybrid clustering method DEMP, which contains two metaheuristics with two different encodings. We first show the BDE algorithm and explain our modification to it. Second, we present the MPA algorithm and explain our adjustments to it. Finally, we offer the DEMP method and how does it merge both BDE and MPA.

3.1 Modified binary differential evolution

The DE is an evolutionary metaheuristic proposed in [29]. In the beginning, it was designed to solve the global optimization problem. A new binary DE version is developed in [9] to solve the problem of feature selection. We modified the BDE algorithm to fit the proposed method (DEMP). In this section, we present the modified version of BDE.

The BDE uses the binary encoding scheme in which each solution is represented as a binary vector with a length equals to the number of patients. The i^{th} element of a vector corresponds to the i^{th} patient, as shown in Fig. 1. In a specific solution, a patient of index i is considered a centroid of a cluster if it's corresponding element equals one. We modified the initial population generation phase as follows. Let N be the population's size; the algorithm chooses N

random numbers in the range [K_{min}, K_{max}]. Each random number corresponds to a specific vector/solution, where the random number specifies the number of centroids of that vector/solution. Then the algorithm chooses randomly which elements/patients are centroids in each vector/solution. The selected elements are set to 1, while the remaining elements are set to 0.

Many clustering validity methods are used in measuring the clustering algorithms results in terms of the clustering cohesion and separation [25, 30]. We chose the DB index [31] as the fitness function because of its capability of finding feasible high-quality results and its computational cost-efficiency [25]. The DB index method, which is used as the fitness function, includes calculating the distance between every dataset object and its cluster centroid every time it runs. Calculating the distance between every object and its cluster centroid is extremely time-consuming because of the massive number of features in the multi-omics datasets. Since we use the binary encoding, the BDE chooses centroids of the clusters from the dataset's existing objects. Accordingly, we made a modification, which is to calculate the distances between every pair of the dataset's objects once and save them in a table at the beginning of the BDF algorithm. So, instead of calculating these distances with each calling of the fitness function, it can be obtained directly from the previously mentioned saved table, which leads to a vast decrease in the execution time of the proposed method.

BDE has only three operators, which are mutation, crossover, and selection. In this study, BDE uses the same mutation operator used in [32]. At the beginning of the mutation process for each vector X_i, BDE selects three random vectors from the population X_{r1}, X_{r2}, and X_{r3} while $r1 \neq r2 \neq r3$. Then BDE calculates the difference vector described in Eq. (1) where *d* is element order in the vector, and *i* is the vector order in the population. After that, the mutant vector is obtained, as shown in Eq. (2). Since BDE is frequently used in solving feature selection problems, it always uses the traditional crossover operator [9, 32]. The traditional crossover operator works on each vector's element alone, which may lead to the number of ones/centroids in the trial vector outside the specified range [K_{min}, K_{max}]. If this occurs, the obtained trial vector will represent an unfeasible solution and needs to be repaired. We propose a modified version of the crossover operator in this article to avoid unfeasible solutions. Instead of working on each vector's element alone, our proposed crossover operator works only on elements that equal one (centroids) in both target and mutant vectors as shown Eq. (3).

$$diff.vector_i^d = \begin{cases} 0 & if\, x_{r1}^d = x_{r2}^d \\ x_{r1}^d & Otherwise \end{cases} \quad (1)$$

$$mutant\ vector_i^d = \begin{cases} 1 & if\ diff.vector_i^d = 1 \\ x_{r1}^d & Otherwise \end{cases} \quad (2)$$

$$trialCentorids_i^d = \begin{cases} mutantCentroids_i^{r1} & r \leq CR(t) \\ targetCentroids_i^{r2} & Otherwise \end{cases} \quad (3)$$

The trial-centroids, mutant-centroids and target-centroids are integer vectors that contain the centroids' indexes for the trial, mutant, and target vectors respectively. *r1* and *r2* are unique integer random numbers. *r1* ranges from 1 to the size of mutant-centroids vector, while *r2* is between 1 and the size of target-centroids vector. Then, the binary trial vector's elements that their indexes exist in trial-centroids are set to 1 as shown in Eq. (4).

$$trial_i^d = 1 \qquad d \in \text{trailCentroids}_i \quad (4)$$

Finally, the selection operator is the same as the one used in [32]. In the selection process, the BDE replaces the target vector with the trial vector only if the trial vector has a better fitness value. Otherwise, it keeps the target vector in the next population.

Algorithm 1 BDE pseudocode

Initialize the first population
While termination condition
  -Generate a mutant vector for every target vector using Eq. (1) and Eq. (2)
  -Generate a trail vector for every target vector using Eq. (3) and Eq. (4)
  -Evaluate target and trial vectors using DB index [31].
  -Replace the target vector with the trial vector only if the trial vector has a better fitness value
End while

3.2 Marine predators algorithm

MPA is a recent nature-inspired metaheuristic inspired mainly by the food searching strategies, namely Brownian and Lévy movements in ocean predators [10]. MPA considers both predator and prey as a search agent as the prey itself is a predator when it searches for its food.

Each search agent in the initial population is initialized randomly, as shown in Eq. (5). The a_{max} and a_{min} are the upper and lower bound for variables, while *r* is a random number between 0 and 1.

MPA represents the best predators in a separate matrix *E* for Elite which mathematically represented in Eq. (6). Another matrix to represent the preys called *P* is mathematically represented in Eq. (7)

$$a_i = a_{min} + r\,(a_{max} - a_{min}) \tag{5}$$

$$\mathrm{E} = \begin{bmatrix} a^I_{1,1} & a^I_{1,2} & \dots & a^I_{1,d} \\ a^I_{2,1} & a^I_{2,2} & \dots & a^I_{2,d} \\ \dots & \dots & \dots & \dots \\ a^I_{n,1} & a^I_{n,2} & \dots & a^I_{n,d} \end{bmatrix} \tag{6}$$

$$\mathrm{P} = \begin{bmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,d} \\ a_{2,1} & a_{2,2} & \dots & a_{2,d} \\ \dots & \dots & \dots & \dots \\ a_{n,1} & a_{n,2} & \dots & a_{n,d} \end{bmatrix} \tag{7}$$

MPA balances exploration and exploitation by mixing two types of searching strategies, which are Brownian and Lévy, through dividing the optimization process into three phases.

In the first phase, when the exploration is more important, the search agents (Preys) use the Brownian for exploration as modelled in Eq. (8) and Eq. (9). This phase is performed in the first third of iterations.

$$\mathrm{s}_i = r_B \odot (E_i - r_B \odot P_i) \quad i = 1, \dots . n \tag{8}$$

$$P_i = P_i + d.R \odot \mathrm{s}_i \tag{9}$$

s_i is the step size for the $pray_i$. *R* is a vector of uniformly random numbers between 0 and 1 while *d*=0.5. r_B is a random vector that represents the Brownian movement. $\odot$ refers to the element-wise multiplication.

During the second phase, when both exploration and exploitation have the same importance, half of the search agents (Prey) use the Lévy strategy for exploitation as modelled in Eq. (10) and Eq. (11).

$$\mathrm{s}_i = r_L \odot (E_i - r_L \odot P_i) \quad i = 1, \dots . \frac{n}{2} \tag{10}$$

$$P_i = P_i + d.R \odot \mathrm{s}_i \tag{11}$$

r_L is a random vector that represents the Lévy strategy. In contrast, the other half (Predators) use the Brownian for exploration as modelled in Eq. (12) and Eq. (13).

$$\mathrm{s}_i = r_B \odot (r_B \odot E_i - P_i) \quad i = \frac{n}{2}, \dots , n \tag{12}$$

$$P_i = E_i + d.AP \odot \mathrm{s}_i \tag{13}$$

The *AP* is an adaptive parameter that equals$(1 - \frac{iter}{\max_iter})^{(2*\frac{iter}{\max_iter})}$. In the last phase, when the exploitation is more important, the search agents (predators) use the Lévy for exploitation as modelled in Eq. (14) and Eq. (15). In that way, MPA moves smoothly from the exploration to the exploitation phase.

$$\mathrm{s}_i = r_L \odot (r_L \odot E_i - P_i) \quad i = 1, \dots , n \tag{14}$$

$$P_i = E_i + d.AP \odot \mathrm{s}_i \tag{15}$$

To avoid stuck in a local optimum, MPA mimics the effect of Fish Aggregating Devices (*FADs*). That effect makes search agents take a long jump in different dimensions at a certain probability which modelled in Eq. (16).

$$P_i = \begin{cases} P_i + AP\,[a_{min} + R \odot (a_{max} - a_{min})] \odot \mathrm{B} & \text{if } \mathrm{r} \le \mathrm{FADs} \\ P_i + [FADs(1-r) + r](P_{r1} - P_{r2}) & \text{if } \mathrm{r} > \mathrm{FADs} \end{cases} \tag{16}$$

FADs parameter represents the FADs effect's probability that equals 0.2. *B* is a binary array which is constructed by using another array *R* with the same length. Each element in *R* is a real number between 0 and 1. The element in *B* equals 1 if the corresponding element in *R* less than 0.2, otherwise it equals 0. *r1* and *r2* indicate random indexes of the *P* matrix. The newly generated search agents are only added to the new population if their fitness values are better than their corresponding agents in the current population. The Elite matrix is only updated if better solutions than the Elite solutions appear at the end of every iteration. MPA pseudocode is presented in Algorithm 2. For more details about MPA, please refer to [10].

Algorithm 2 MPA pseudocode

```
Initialize the first population Eq. (5)
While less than the max number of iterations
    Evaluate each search agent
    If iterNum < Max(iter) / 3
        Use Brownian to update search agents
        Eq. (8) and Eq. (9)
    Else If 2×Max(iter)/3> iterNum > Max(iter) / 3
        Use Lévy to update half of search agents
        Eq. (10) and Eq. (11)
```

And use Brownian for the other half
Eq. (12) and Eq. (13)
Else If iterNum >2×Max(iter)/3
Use Lévy to update search agents
Eq. (14) and Eq. (15)
End if
search agents take jumps at a certain probability
Eq. (16)
the new agents are added in the new population
only if it is better than its previous counterparts.
Update Elite matrix.
End While

3.3 The proposed hybrid method (DEMP)

In the beginning, the DEMP method starts by reading omics datasets as presented in Algorithm 3. Each omic dataset is an $n \times m$ matrix where n is the number of patients and m is the number of features for that omic. After that, the DEMP method merges all the omic datasets into one $n \times d$ matrix called M where n is the number of patients and d is the sum of all the features numbers for all omics as described in Eq. (17).

$$M = \begin{bmatrix} c_{1,1} & c_{1,2} & \dots & c_{1,d} \\ c_{2,1} & c_{2,2} & \dots & c_{2,d} \\ \dots & \dots & \dots & \dots \\ c_{n,1} & c_{n,2} & \dots & c_{n,d} \end{bmatrix} \quad (17)$$

In the first stage, DEMP uses the modified version of BDE, presented in Algorithm 1. After finishing the first stage, the DEMP method sets k_{max} equals the clusters' number of the best-obtained clustering solution. Then, it selects the fittest $N/2$ (where N is the MPA population size) solutions from the BDE's last population as long as their clusters' number less than or equal k_{max}. Subsequently, it inserts the selected solutions into MPA's initial population. The other half of the MPA population is initialized using Eq. (5). Before inserting the selected solution into MPA's initial population, the DEMP method must transform the solution encoding from binary to real, as depicted in Fig. 4.

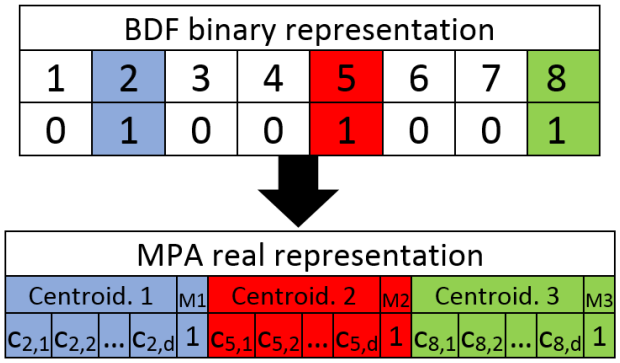

Figure. 4 Transforming binary representation into real representation example where n=8 and k_{max}= 3

The transformation process works as follows, for every element$_i$ that equals 1 (centroid) in the binary representation, the M_i vector Eq. (17) is added to the real representation in addition to a real number between 0 and 1 works as a mask for that centroid. In that way, the binary vector with length n is transformed into a real vector with length equals $(d+1) \times k_{max}$. In the second stage, the DEMP method uses the MPA algorithm, presented in Algorithm 2. Finally, the fittest clustering solution found by MPA is the best clustering solution for the DEMP method. The general schema for the DEMP method is depicted in Fig. 5.

Algorithm 3 DEMP pseudocode

Read more than one omic dataset
Merge all of them into one $n \times d$ matrix Eq. (17)
Run BDE Algorithm 1
k_{max} = fittest clustering solution's clusters number
best_solutions = the best $N/2$ clustering solutions where N is the MPA population's size
transform best_solutions into a real representation as shown in Fig. 4.
Run MPA Algorithm 2
Output the fittest clustering solution

4. Experimental results

In order to measure DEMP's performance, we applied it and four recent and powerful metaheuristics to eight multi-omics datasets available from TCGA. We chose FA [33] based on its superiority in the recent comparative experiment with four other metaheuristics DE, GA, IWO, and PSO in solving the automatic clustering problem [25]. EO is a physics-based metaheuristic while MPA, and SMA are nature-inspired metaheuristics. All of them are recent metaheuristic algorithms proven to be competent in solving global optimization problems. Since MPA is already an essential part of the DEMP method, comparing it with DEMP can reveal whether the DEMP method provides an improvement or not. The comparison experiments have been applied to eight datasets contain cancer tumor multi-omics data, while each dataset belongs to a specific type of cancer. Each dataset includes three omics (gene expression - DNA methylation - miRNA expression). The characteristics of all datasets are shown in Table 1. We pre-processed the datasets as follows: we removed features that have more than 20% missing values. We then removed any patient record has more than 20% values missing. Regarding the methylation data, the highest 5000 features in terms of variance are selected.

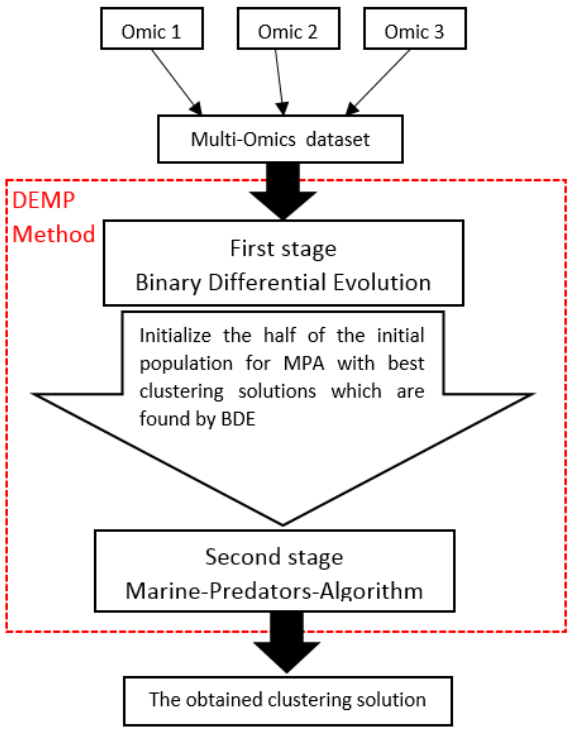

Figure. 5 General schema of DEMP method

We used two criteria to evaluate the proposed method's performance. First, we measured the average fitness value over 20 runs. The fitness function for all algorithms uses the DB index. Second, we used the average execution time in seconds over 20 runs for each algorithm. Tables. 2-9 show the comparative experiment done in this study. All algorithms executed on the windows 8 operating system installed on Intel Core i5 machine with 8 GB of memory using Matlab R2017b.

The number of iterations equals 500, while the number of search agents equals 30 for all algorithms. The K_{max} for all algorithms equals five, while the K_{min} equals two. For any other parameters, we chose what the author recommended. During the first phase of DEMP method (BDE), the number of iterations equals 500, while the number of search agents equals 60 and the K_{max} equals 20. After that, during the second phase MPA, the number of iterations equals 100, while the number of search agents equals 30.

Table 1. Summary of multi-omics datasets

Dataset name	Number of objects	Exp	Methy	miRNA
Aml	159	20531	5000	705
Colon	214	20531	5000	705
Gbm	271	12042	5000	534
Kidney	206	20531	5000	1046
Liver	404	20531	5000	1046
Melanoma	439	20531	5000	1046
Ovarian	290	20531	5000	705
Sarcoma	261	20531	5000	1046

Table 2. Average fitness values and average execution time over 20 runs for the AML dataset. The best result is shown in boldface and underlined, and the second-best is in boldface

algorithm name	Avg fitness	STD	Avg time
FireFly	1.166	0.0187	1.70×10^{4}
EO	1.789	0.3158	**8.39×10^{3}**
MPA	0.813	**0.0084**	9.38×10^{3}
SMA	**0.804**	**0.0033**	8.63×103
DEMP	**0.791**	0.0094	**1234**

It can be seen in Table 2 that DEMP method got the best results in terms of average fitness value and execution time. SMA algorithm got the second-best results in terms of average fitness value, but it took eight times the DEMP's time.

The results of Table 3 show that the best method in terms of average fitness value and average execution time is DEMP. For the second time, SMA algorithm won second place in terms of average fitness value, but it needed five times the DEMP's time.

Table 4 shows that the DEMP method won first place in terms of average fitness value and average execution time. For the first time, MPA algorithm got the second-best results in terms of average fitness value, but it took almost four times the DEMP's time. We can see from Table 5 that, as usual, DEMP method got the best results in terms of average fitness value and execution time. SMA algorithm won second place in terms of average fitness value, but it needed five times the time of DEMP.

We can get from Table 6 that the DEMP method is the best one in terms of the average fitness value and execution time. MPA for the second time comes second in terms of the average fitness value, but it needed nine times the DEMP's time.

Table 3. Average fitness values and average execution time over 20 runs for the colon dataset. The best result is shown in boldface and underlined, and the second-best is in boldface

algorithm name	avg fitness	STD	Avg time
FireFly	1.331	**0.022**	1.99×10^{4}
EO	2.584	0.2711	**7.55×10^{3}**
MPA	1.003	0.0287	8.06×10^{3}
SMA	**0.981**	0.0329	1.81×10^{4}
DEMP	**0.977**	**0.0182**	**1517**

Table 4. Average fitness values and average execution time over 20 runs for the gbm dataset. The best result is shown in boldface and underlined, and the second-best is in boldface

algorithm name	avg fitness	STD	Avg time
FireFly	1.552	**0.0084**	1.11×10^{4}
EO	2.692	0.1147	4.66×10^{3}
MPA	**1.527**	0.0121	**4.58×10^{3}**
SMA	1.590	**0.0071**	5.93×10^{3}
DEMP	**1.235**	0.0438	**1294**

Table 5. Average fitness values and average execution time over 20 runs for the kidney dataset. The best result is shown in boldface and underlined, and the second-best is in boldface

algorithm name	avg fitness	STD	Avg time
FireFly	1.405	**0.025**	**1.46×10^{4}**
EO	2.719	0.300	5.49×10^{3}
MPA	1.074	0.031	5.98×10^{3}
SMA	**1.062**	0.060	8.37×10^{3}
DEMP	**0.964**	**0.021**	**1.53×10^{3}**

Table 7 shows that the best method for getting average fitness value and execution time is DEMP. SMA succeeded to come second even though it took almost three times the time of DEMP.

Table 6. Average fitness values and average execution time over 20 runs for the liver dataset. The best result is shown in boldface and underlined, and the second-best is in boldface

algorithm name	avg fitness	STD	Avg time
FireFly	0.8814	0.0168	2.34×10^{4}
EO	1.3478	0.3454	**9.13×10^{3}**
MPA	**0.642**	**0.0024**	9.23×10^{3}
SMA	0.6461	**0.0034**	1.04×10^{4}
DEMP	**0.6313**	0.0042	**2757**

Table 7. Average fitness values and average execution time over 20 runs for the melanoma dataset. The best result is shown in boldface and underlined, and the second-best is in boldface

algorithm name	avg fitness	STD	Avg time
FireFly	1.434	0.0353	2.23×10^{4}
EO	3.086	0.4717	1.02×10^{4}
MPA	0.950	**0.0134**	**9.88×10^{3}**
SMA	**0.934**	**0.0158**	1.05×10^{4}
DEMP	**0.914**	0.0198	**2952**

Table 8. Average fitness values and average execution time over 20 runs for the ovarian dataset. The best result is shown in boldface and underlined, and the second-best is in boldface

algorithm name	avg fitness	STD	Avg time
FireFly	1.308	**0.0198**	1.74×10^{4}
EO	2.223	0.5269	**6.90×10^{3}**
MPA	1.010	**0.0189**	7.02×10^{3}
SMA	**0.989**	0.0382	9.01×10^{3}
DEMP	**0.960**	0.0383	**1941**

The results of Table 8 show that the method with the best average fitness value is DEMP. It also has the shortest execution time. SMA algorithm got the second-best results in terms of average fitness value, but it took almost five times the DEMP's time.

As is evident from the results of Table 9, DEMP method obtained the clustering solutions that have the best average fitness value. On the other hand, MPA came in second place, while it needed almost five times the time of DEMP.

Table 10 reveals that all experimental results are statistically significant, except for comparing DEMP and SMA on the colon dataset. So, we can conclude that the proposed method DEMP not only has an

Table 9. Average fitness values and average execution time over 20 runs for the sarcoma dataset. The best result is shown in boldface and underlined, and the second-best is in boldface

algorithm name	avg fitness	STD	Avg time
FireFly	1.233	0.0381	1.48×10^{4}
EO	1.523	0.2834	6.88×10^{3}
MPA	**0.727**	**0.0066**	**6.81×10^{3}**
SMA	0.737	**0.0074**	8.87×10^{3}
DEMP	**0.720**	0.0087	**1.87×10^{3}**

Table 10. The p-values of Wilcoxon's test for DEMP method against the other considered algorithms through eight datasets

	FireFly	EO	MPA	SMA
AML	0.00001	0.00001	0.00004	0.00005
Colon	0.00001	0.00001	0.00151	**0.95443**
gbm	0.00001	0.00001	0.00001	0.00001
Kidney	0.00001	0.00001	0.00001	0.00001
Liver	0.00001	0.00001	0.00001	0.00001
melanom	0.00001	0.00001	0.00002	0.0027
Ovarian	0.00001	0.00001	0.00031	0.00325
Sarcoma	0.00001	0.00001	0.01099	0.00003

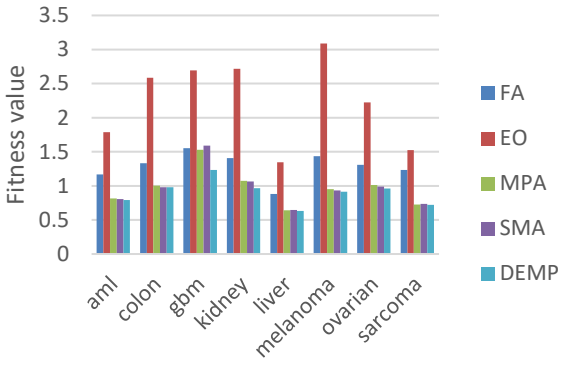

Figure. 6 Average fitness values over 20 runs for all dataset

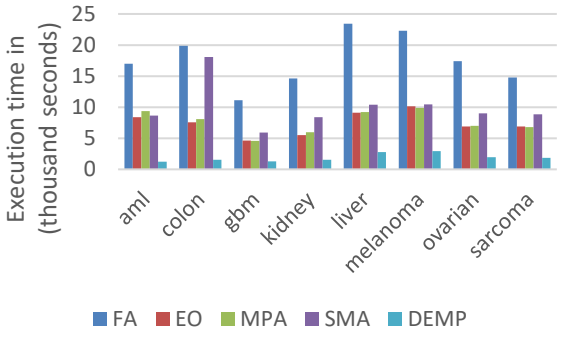

Figure. 7 Average execution time over 20 runs for all dataset

advantage over other algorithms, but almost all the results are statistically significant.

The results in Fig. 6 and Fig. 7 are deduced from Tables. 2-9. These results show us that DEMP's average fitness values are the best on all datasets. Besides, it is far faster than other algorithms under study.

5. Discussions

5.1 DEMP's superiority explanation

Although metaheuristics that use centroids-base real encoding can achieve better results than those who use binary encoding, they take a long execution time because solving the automatic clustering problem for multi-omics datasets is more complicated than other datasets. Fig. 3 shows the real encoding example for a dataset that each centroid can be represented with a single real value. In this case, each solution's length equals $2 \times k_{max}$ since we need extra decision variables for the activation thresholds. In case of the multi-omics dataset, instead of representing each centroid with a single real number, it is represented with d real numbers as depicted in Fig. 4. So, in that case, the number of decision variables equals $k_{max} \times d + k_{max}$. Since k_{max} normally equals $\sqrt{N}$ where N is the number of objects/patients in the dataset, the number of decision variables equals $\sqrt{N} \times \mathrm{d} + \sqrt{N}$. If we consider the data in Table 1, this leads to a huge number of decision variables for each clustering solution, which greatly increases the search space.

From the previous section, we can see the superiority of DEMP over all the real encoding metaheuristics under study. We can explain this for the following reasons. First, during the DEMP's first stage, it uses BDE that uses the binary encoding which needs only a binary vector of length N to represent a clustering solution as shown in Fig. 1. BDE looks for the centroids only in the objects/patients in the dataset, which allows it to converge faster. Second, our modification which is to calculate the distances between every pair of the dataset's objects once and save them in a table at the beginning of the BDF algorithm. This modification greatly increased the speed of BDF and thus DEMP since it only calculates the distance between two real victors (with length equals d) N^2 times. On the contrast, without this modification, DEMP needs to calculate the same distance almost $(I \times S \times \mathrm{k_{max}} / 2)$ times. Where I is BDE's iterations number and S is BDE's search agents number. Third, at the end of the first stage, the $\mathrm{K_{max}}$ for the second stage is determined. Often the determined $\mathrm{K_{max}}$ is smaller than the normal $\mathrm{K_{max}}$ that equals the $\sqrt{N}$. As mentioned earlier in this section, the number of the decision variables equals $\mathrm{k_{max}} \times d + \mathrm{k_{max}}$. Therefore, if the $\mathrm{K_{max}}$ decreases by x, the number of decision variables is reduced by $x \times d$, which dramatically reduces the search space and convergence time.

Fourth, at the beginning of the second stage, half of MPA initial population initialized using the fittest clustering solutions in the BDE final population. So, MPA in the second stage needs only 100 iterations to get better results than other metaheuristics that need 500 iterations.

Since the genomic data amount grows fast and, new omics are expected to appear in the future, a faster clustering method such as DEMP will have a greater chance to cope with that growth. Therefore, it is expected that the DEMP method will become widespread in the future.

6. Conclusions

This paper proposes a new hybrid automatic clustering method based on BDE and MPA called DEMP method. BDE uses a binary encoding to solve the binary optimization problem, while MPA uses real encoding to solve the global optimization problem. DEMP method combines the speed

convergence of BDE and accurate clustering results of MPA to improve both the clustering solution accuracy and execution time. We proposed adjustments for both algorithms to allow them to solve the problem of automatic clustering.

This article evaluated the DEMP method's performance using clustering accuracy and execution time on eight multi-omics datasets. Moreover, we compared the DEMP method to four powerful and recent optimization algorithms FA, EO, MPA, and SMA. The experimental results reveal that the DEMP method could solve the automatic clustering problem for multi-omics datasets. Furthermore, the results show that the clustering solutions accuracy of the DEMP method is much superior to other algorithms under study. It is also evident from the results that the DEMP's execution time is much less than the algorithms under investigation. Finally, we should note that 97% of the comparisons are statistically significant. Based on the above, it can be concluded that DEMP is a powerful automatic clustering tool for multi-omics datasets.

As future research, DEMP can be applied to other problems with a vast number of features. Besides, adding an efficient local search method to DEMP can further improve the clustering solution quality.

Conflicts of Interest

The authors declare no conflict of interest.

Author Contributions

Conceptualization, EN and HAH; methodology, MG and EN; software, MG; validation, MG, and EN; formal analysis, MG; investigation, MG; data curation, MG; writing—original draft preparation, MG; writing—review and editing, EN; visualization, MG; supervision, EN; project administration, HAH.

References

[1] N. Rappoport and R. Shamir, "Multi-Omic and Multi-View Clustering Algorithms: Review and Cancer Benchmark", *Nucleic Acids Research*. Vol. 46, No. 20, pp. 10546–10562, 2018.

[2] A. K. Jain, "Data Clustering: 50 Years beyond K-Means", *Pattern Recognition Letters*. Vol. 31, No. 8, pp. 651–666, 2010.

[3] R. Xu and D. Wunsch, "Survey of Clustering Algorithms", *IEEE Transactions on Neural Networks*. Vol. 16, No. 3, pp. 645–678, 2005.

[4] M. Nicholson, "Genetic Algorithms and Grouping Problems", *Software: Practice and Experience*. Vol. 28, No. 10, pp. 1137–1138, 1998.

[5] C.-W. Bong and M. Rajeswari, "Multi-Objective Nature-Inspired Clustering and Classification Techniques for Image Segmentation", *Applied Soft Computing*. Vol. 11, No. 4, pp. 3271–3282, 2011.

[6] J. Handl and J. Knowles, "An Evolutionary Approach to Multiobjective Clustering", *IEEE Transactions on Evolutionary Computation*. Vol. 11, No. 1, pp. 56–76, 2007.

[7] S. Bandyopadhyay and U. Maulik, "Genetic Clustering for Automatic Evolution of Clusters and Application to Image Classification", *Pattern Recognition*. Vol. 35, No. 6, pp. 1197–1208, 2002.

[8] E. R. Hruschka, R. J. Campello, A. A. Freitas, and others, "A Survey of Evolutionary Algorithms for Clustering", *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*. Vol. 39, No. 2, pp. 133–155, 2009.

[9] E. Zorarpac and S. A. Özel, "A Hybrid Approach of Differential Evolution and Artificial Bee Colony for Feature Selection", *Expert Systems with Applications*. Vol. 62, pp. 91–103, 2016.

[10] A. Faramarzi, M. Heidarinejad, S. Mirjalili, and A. H. Gandomi, "Marine Predators Algorithm: A Nature-Inspired Metaheuristic", *Expert Systems with Applications*. Vol. 152, p. 113377, 2020.

[11] G. Garai and B. Chaudhuri, "A Novel Genetic Algorithm for Automatic Clustering", *Pattern Recognition Letters*. Vol. 25, No. 2, pp. 173–187, 2004.

[12] H.-J. Lin, F.-W. Yang, and Y.-T. Kao, "An Efficient GA-Based Clustering Technique", *Tamkang Journal of Science and Engineering*. Vol. 8, No. 2, pp. 113–122, 2005.

[13] E. R. Hruschka and N. F. Ebecken, "A Genetic Algorithm for Cluster Analysis", *Intelligent Data Analysis*. Vol. 7, No. 1, pp. 15–25, 2003.

[14] E. R. Hruschka, R. J. Campello, and L. N. de Castro, "Improving the Efficiency of a Clustering Genetic Algorithm", In: *Proc. of Ibero-American Conf. on Artificial Intelligence*, Berlin, Heidelberg, pp. 861–870, 2004.

[15] S. Das, A. Chowdhury, and A. Abraham, "A Bacterial Evolutionary Algorithm for Automatic Data Clustering", In: *2009 IEEE Congress on Evolutionary Computation*, Trondheim, Norway, pp. 2403–2410, 2009.

[16] P. C. Ma, K. C. Chan, X. Yao, and D. K. Chiu, "An Evolutionary Clustering Algorithm for Gene Expression Microarray Data Analysis",

IEEE Transactions on Evolutionary Computation. Vol. 10, No. 3, pp. 296–314, 2006.

[17] E. R. Hruschka, R. J. Campello, and L. N. De Castro, "Evolving Clusters in Gene-Expression Data", *Information Sciences*. Vol. 176, No. 13, pp. 1898–1927, 2006.

[18] D. Horta, I. C. De Andrade, and R. J. Campello, "Evolutionary Fuzzy Clustering of Relational Data", *Theoretical Computer Science*. Vol. 412, No. 42, pp. 5854–5870, 2011.

[19] Y. Liu, X. Wu, and Y. Shen, "Automatic Clustering Using Genetic Algorithms", *Applied Mathematics and Computation*. Vol. 218, No. 4, pp. 1267–1279, 2011.

[20] J. Qu, Z. Shao, and X. Liu, "Mixed PSO Clustering Algorithm Using Point Symmetry Distance", *Journal of Computational Information Systems*. Vol. 6, No. 6, pp. 2027–2035, 2010.

[21] Y. Kao and C.-C. Chen, "Automatic Clustering for Generalised Cell Formation Using a Hybrid Particle Swarm Optimisation", *International Journal of Production Research*. Vol. 52, No. 12, pp. 3466–3484, 2014.

[22] A. Chowdhury, S. Bose, and S. Das, "Automatic Clustering Based on Invasive Weed Optimization Algorithm", In: *Proc. of International Conf. on Swarm, Evolutionary, and Memetic Computing*, Berlin, Heidelberg, pp. 105–112, 2011.

[23] M. Sarkar, B. Yegnanarayana, and D. Khemani, "A Clustering Algorithm Using an Evolutionary Programming-Based Approach", *Pattern Recognition Letters*. Vol. 18, No. 10, pp. 975–986, 1997.

[24] R. Liu, L. Jiao, X. Zhang, and Y. Li, "Gene Transposon Based Clone Selection Algorithm for Automatic Clustering", *Information Sciences*. Vol. 204, pp. 1–22, 2012.

[25] A. E. Ezugwu, "Nature-Inspired Metaheuristic Techniques for Automatic Clustering: A Survey and Performance Study", *SN Applied Sciences*. Vol. 2, No. 2, pp. 273–330, 2020.

[26] P. P. W. Cho and T. T. S. Nyunt, "Data Clustering Based on Modified Differential Evolution and Quasi-Opposition-Based Learning", *Intelligent Engineering & Systems*. Vol. 13, No. 6, pp. 168–178, 2020.

[27] A. Faramarzi, M. Heidarinejad, B. Stephens, and S. Mirjalili, "Equilibrium Optimizer: A Novel Optimization Algorithm", *Knowledge-Based Systems*. Vol. 191, p. 105190, 2020.

[28] S. Li, H. Chen, M. Wang, A. A. Heidari, and S. Mirjalili, "Slime Mould Algorithm: A New Method for Stochastic Optimization", *Future Generation Computer Systems*. Vol. 111, pp. 300–323, 2020.

[29] R. Storn and K. Price, "Differential Evolution-a Simple and Efficient Heuristic for Global Optimization over Continuous Spaces", *Journal of Global Optimization*. Vol. 11, No. 4, pp. 341–359, 1997.

[30] A. José-Garcia and W. Gómez-Flores, "Automatic Clustering Using Nature-Inspired Metaheuristics: A Survey", *Applied Soft Computing*. Vol. 41, pp. 192–213, 2016.

[31] D. L. Davies and D. W. Bouldin, "A Cluster Separation Measure", *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Vol. 1, No. 2, pp. 224–227, 1979.

[32] J. Too, A. R. Abdullah, and N. Mohd Saad, "Hybrid Binary Particle Swarm Optimization Differential Evolution-Based Feature Selection for EMG Signals Classification", *Axioms*. Vol. 8, No. 3, pp. 79–95, 2019.

[33] X.-S. Yang, "Firefly Algorithms for Multimodal Optimization", In: *International Symposium on Stochastic Algorithms*, Berlin, Heidelberg, pp. 169–178, 2009.