SN Computer Science (2021) 2:185
https://doi.org/10.1007/s42979-021-00511-0

ORIGINAL RESEARCH

EvoCluster: An Open-Source Nature-Inspired Optimization Clustering Framework

Raneem Qaddoura[1] · Hossam Faris[2] · Ibrahim Aljarah[2] · Pedro A. Castillo[3]

Received: 4 August 2020 / Accepted: 8 February 2021 / Published online: 31 March 2021

Abstract

EvoCluster is an open source and cross-platform framework implemented in Python language, which includes the most well-known and recent nature-inspired metaheuristic optimizers that are customized to perform partitional clustering tasks. This paper is an extension to the existing EvoCluster framework in which it includes different distance measures for the objective function, different techniques of detecting the k value, and a user option to consider either supervised or unsupervised datasets. The current implementation of the framework includes ten metaheuristic optimizers, thirty datasets, five objective functions, twelve evaluation measures, more than twenty distance measures, and ten different ways for detecting the k value. The source code of EvoCluster is publicly available at http://evo-ml.com/evocluster/.

Keywords Clustering · Cluster analysis · Evolutionary computing · Framework · Python

Introduction

Clustering is an unsupervised learning task that is essential in many applications. The main goal of clustering is to find the similarities between every group of data to find common relationships between them. It is widely used in different domains such as medical diagnosis [53], ransomeware detection [54], customer segmentation [49], image processing [33], dental radiography [48], and pattern recognition [34].

Swarm intelligence (SI) and evolutionary algorithms (EA) as nature-inspired metaheuristic algorithms are commonly utilized for performing partitional clustering tasks. They are proven to be efficient for multiple scientific and engineering domains [43]. The main advantage of using these algorithms in clustering is the ability to explore and search for better grouping of data to achieve high quality clustering results [43]. In addition, they have reasonable running time [63], they can avoid falling in local optima [47], and they can work with noisy data [25].

Metaheuristic algorithms use predefined objective function to lead the solution toward the optimal one [50]. The objective function directly affects the quality of the results [55]. Thus, considering the best objective function is very important, and is not an easy task.

Nature-inspired algorithms include well-regarded optimization algorithms such as Genetic algorithm (GA) [22], Evolution strategy (ES) [9], Particle swarm optimization (PSO) [27], and Ant colony optimization (ACO) [31]. While noticeable recent nature-inspired algorithms include Cuckoo Search (CS) [73], Grey Wolf Optimizer (GWO) [7, 42], Multi-verse optimizer (MVO) [8, 41, 63], Moth-flame optimization (MFO) [38], Whale Optimization Algorithm

This article is part of the topical collection "Evolution, the New AI Revolution" guest edited by Anikó Ekárt and Anna Isabel Esparcia-Alcázar.

✉ Pedro A. Castillo
pacv@ugr.es

Raneem Qaddoura
rqaddoura@philadelphia.edu.jo
http://www.evo-ml.com

Hossam Faris
hossam.faris@ju.edu.jo
http://www.evo-ml.com

Ibrahim Aljarah
i.aljarah@ju.edu.jo
http://www.evo-ml.com

[1] Information Technology, Philadelphia University, Amman, Jordan

[2] King Abdullah II School for Information Technology, The University of Jordan, Amman, Jordan

[3] ETSIIT-CITIC, University of Granada, Granada, Spain

(WOA) [40], Bat Algorithm (BAT) [72], Firefly Algorithm (FFA) [71], and many others [2, 4–6].

EvoCluster is an open-source framework for partitional clustering based on using nature-inspired metaheuristic optimizers. It provides a set of nature-inspired optimizers for performing the partitional clustering task to facilitate using these algorithms by researchers and practitioners, It provides a wide set of objective functions that are customized for the partitional clustering task. It also facilitates the evaluation process of clustering by including many well-known evaluation measures for clustering and providing a set of well-known data sets which are widely used for performing experiments in clustering.

In this paper, we extend the current implementation of EvoCluster to include the following:

- Different distance measures for the objective function from the well-known *scipy.spatial.distance* package [66].
- A user option to run the framework on either supervised or unsupervised datasets. Accordingly, different presentations of the results are held.
- Different ways to specify the k value; The user can determine a specific k value for each dataset, or choose the automatic option. The automatic option includes automatically deriving the k value from the true labels, or applying different detection techniques such as the Elbow method to automatically predict the k value for each dataset.

The remainder of this paper is organized as follows: Sect. 2 presents the latest evolutionary and clustering frameworks and libraries in the literature. Section 3 describes in detail the components and functionalities of the framework including the extended features. Sect. 4 displays the design of population and the framework components. Section 5 shows some visual representation of some results and plots generated from the framework. Section 6 concludes the work and gives further other possible extensions to the framework.

Related Work

Many frameworks and libraries can be found in the literature to perform the clustering tasks. Some popular examples are Weka [19], Elki [1] , and scikit-learn [45]. Some other frameworks that are specific to clustering are clusterNOR [37] and ClustEval [70]. clusterNOR is a parallel framework which includes nine clustering algorithms, while ClustEval is a recent framework that includes around twenty well-known algorithms and fourteen evaluation measures. Other frameworks are specific to certain domains: TimeClust [35] is a clustering tool for gene expression time series having four clustering algorithms. A recent framework was developed in [57] named clusterExperiment for clustering single-cell RNA-Seq data. These clustering frameworks and libraries include the basic and traditional clustering algorithms and most of them do not include nature-inspired metaheuristic optimizers.

Since nature-inspired metaheuristic algorithms are commonly used in different applications, some general-purpose frameworks and libraries were developed to facilitate their use. EvoloPy [16] is one of the recent open-source Python frameworks that includes well-known and recent nature-inspired optimizing algorithms. It aims at facilitating the use of the optimization algorithms by researchers and practitioners for different problems. The framework is scalable and can be customized to include additional algorithms and benchmark functions as well as modifications to existing implementation. It also provides numerical and graphical representation of the results. Evolopy-FS [28] is another version of Evolopy for feature selection.

On the other hand, NiaPy [67] is a Python microframework for building nature-inspired algorithms. Other popular frameworks include DEAP [18], ECJ [69], EO [26], HeuristicLab [68], jMetal [15], and ParadisEO [10]. Some frameworks are specific to certain domains: GEATbx [20] is a framework in MATLAB having many variants of the Genetic Algorithms and Genetic Programming. GAlib [36] is a C++ library of genetic algorithm tools and operators for parallel environments. These frameworks are used for optimizing general problems which do not include clustering.

Since the clustering task can be approached by optimizing the centroids for the clusters according to a predefined objective function, which needs special implementation, frameworks and libraries can be implemented for this purpose. To the best of our knowledge, we found only one framework for clustering with evolutionary algorithms which is LEAC [58]. LEAC is implemented using C++ which includes 23 Evolutionary Algorithms for partitional clustering. However, most practitioners use other languages which have more libraries and packages than C++ language. In addition, the algorithms used in LEAC are only different variations of the evolutionary operators for fixed and variable k-clusters and do not include other algorithms. Thus, there is a need for a framework of nature-inspired metaheuristic clustering optimizers which is not specific to the evolutionary algorithms.

In this sense, EvoCluster [51] is a flexible framework that includes several nature-inspired metaheuristic optimizers for performing the clustering task. This framework allows users' customizations of the clustering algorithms, the objective functions, and the evaluation measures. EvoCluster is an extension to the aforementioned EvoloPy framework in which the algorithms are customized for the partitional clustering task. It also considers multiple objective functions for enhancing the performance of the population at

each iteration. Evaluation measures are also implemented for evaluating the clustering results. To the best of our knowledge, EvoCluster framework is the first framework in Python language for clustering data using the selected metaheuristic optimizers. It is implemented efficiently considering the computation time and the quality of results.

In addition, the effect of the selected distance measure and the technique used for specifying the number of clusters in the framework should be investigated. Although the effect of different distance measures was discussed and compared in several works, some were specific to a certain domain or application [17, 23, 44], others were based on a specific algorithm [30, 52], and other were examined on a specific type of datasets such as the ones with high dimensionality [44, 52]. On the other hand, specifying the k value has been discussed in several works [29, 32, 64] which include the Elbow, gap analysis, Silhouette coefficient, Calinski-Harabasz (CH), Davies-Bouldin (DB), and Bayesian Information Criterion score (BIC) techniques. Therefore, due to the importance of these parameters, both are covered in the extended implementation of the framework.

Under this view, extending useful frameworks such as EvoCluster with additional features gives researchers and practitioners more possibilities and opportunities while using the framework. Thus, we are extending the current implementation of EvoCluster to include new different distance measures for the objective function, different ways of specifying the k value, and a user option to consider either supervised or unsupervised datasets.

Framework Overview

EvoCluster includes the most well-regarded nature-inspired metaheuristic optimizers that are adopted for performing the partitional clustering task with a very easy and useful interface. The framework is constructed with six main components which are described in the following sections:

The Optimizer

It serves as the main interface of the framework. Users can select the set of optimizers, data sets, and objective functions according to their preference for running the experiments. They can also specify the main parameters that are common for most optimizers which are the number of iterations and the population size. The number of runs can also be determined through this interface. In addition, users select the evaluation measures to evaluate the predicted labels generated from the framework.

The optimizer is further extended to include different distance measures from the well-known *scipy.spatial.distance* package [66]. It includes braycurtis, canberra, chebyshev, cityblock, correlation, cosine, dice, euclidean, hamming, jaccard, jensenshannon, kulsinski, mahalanobis, matching, minkowski, rogerstanimoto, russellrao, seuclidean, sokalmichener, sokalsneath, sqeuclidean, and yule distance measures [66].

The user is also allowed to run the framework on datasets that are either unsupervised or supervised according to the existence of the true labels in the datasets. In the case of unsupervised datasets, results files do not include the evaluation measures which depend on comparisons between the true and predicted labels.

Another feature is added to the framework; the user can specify the k value by different ways: first, the user can determine specific k values for the datasets. The user passes a list of k integer values corresponding to the list of datasets provided to the framework. Second, the user can choose the automatic option. The automatic option includes automatically deriving the k value from the true labels if the user provided supervised datasets, or applying different detection techniques to automatically predict the k value for each dataset. The detection techniques that are provided at the time of writing this paper are: elbow, gap analysis, Silhouette coefficient, Calinski–Harabasz (CH), Davies–Bouldin (DB), and Bayesian Information Criterion score (BIC) techniques. Other options include the minimum, maximum, median, and majority k values calculated from the values predicted by all these techniques.

Nature-Inspired Metaheuristics

The implementation of each optimizer is visible as a separate file in the framework. The optimizers that are available at the time of writing this paper are as follows:

- Genetic algorithm (GA) [61]: It is inspired by biological evolution. The algorithm evolves toward better solutions based on four main operations: selection, crossover, mutation, and elitism.
- Particle swarm optimization (PSO) [28, 62]: It is inspired by the flocking behavior of birds and the schooling behavior of fish. The algorithm evolves toward better solutions based on a mathematical formula considering the position and velocity of the particles. The movement of the particle is influenced by its local best and the global best positions.
- Salp swarm algorithm (SSA) [39]: It is inspired by the swarming behavior of salps. The algorithm evolves toward better solutions based on two mathematical models to update the position of leading and follower salps.
- Firefly algorithm (FFA) [71]: It is inspired by the flashing behavior of fireflies. The algorithm evolves toward better solutions by the attraction of fireflies based on the bright-

ness of other fireflies calculated by the inverse square law.
- Gray Wolf optimizer (GWO) [42]: It is inspired by grey wolves. The algorithm evolves toward better solutions based on hunting, searching for prey, encircling prey, and attacking prey.
- Whale optimization algorithm (WOA) [40]: It is inspired by social behavior of humpback whales. The algorithm evolves toward better solutions based on three operators to simulate the search for prey, encircling prey, and bubble-net foraging behavior of humpback whales.
- Multi-verse optimizer (MVO) [41]: It is inspired by the theory of multi-verse in physics. The algorithm evolves toward better solutions based on mathematical models of the white hole, black hole, and worm hole which reflect exploration, exploitation, and local search, respectively.
- Moth flame optimizer (MFO) [38]: It is inspired by the death behavior of moths. The algorithm evolves toward better solutions based on logarithmic spiral update mechanism of moths.
- Bat algorithm (BAT) [72]: It is inspired by the echolocation behavior of bats. The algorithm evolves toward better solutions based on the pulse of loudness and pulse rate.
- Cuckoo search algorithm (CS) [73]: It is inspired by the brood parasitism of some cuckoo species. The algorithm evolves toward better solutions based on three idealized rules where the bird decides whether it throws the eggs away or abandons its nest and creates a new one.

Objective Functions

This component consists of the implementation of the objective functions that are used to optimize the individuals at each iteration. The list of objective functions which are used with the data sets having k clusters of N points, available at the time of writing this paper, are as follows:

- Sum of squared error (SSE) [11]:

$$\sum_{n=1}^{N} d_{nc}^2 \tag{1}$$

where d_{nc} is the Euclidean distance between the centroid and the point. By minimizing SSE, we obtain better results.
- Total within cluster variance (TWCV) [46]:

$$\text{TWCV} = \sum_{n=1}^{N}\sum_{f=1}^{F} p_{nf}^2 - \sum_{k=1}^{K}\frac{1}{|p_k|}\sum_{f=1}^{F}\left(\sum p_{kf}\right)^2 \tag{2}$$

where F is the number of features, p_{nf} is feature f of the point n, p_{kf} is feature f of the point k, and $|p_k|$ is the number of points in cluster k. By minimizing TWCV, we obtain better results.
- Silhouette coefficient (SC) [13, 46]:

$$\text{SC} = \frac{\sum_{k=1}^{|K|}((b-a)/max(a,b)))}{N} \tag{3}$$

where a is the average distance between a point and the other points in the same predicted cluster, and b is the average distance between a point and the other points in the next nearest cluster. By maximizing SC, we obtain better results. We normalize the values of SC to the interval [0, 1], and then use the reversed value of the normalized SC (1 − norm(SC)) in the objective function.
- Davies–Bouldin (DB) index [14]:

$$\text{DB} = \frac{1}{|k|}\sum_{k=1}^{|K|}\max_{k\neq j}\left(\frac{s_k + s_j}{d_{kj}}\right) \tag{4}$$

where s_k is the average distance between a point and the cluster center, and d_{kj} is the distance between the centroid of cluster k and the centroid of cluster j. By minimizing DB, we obtain better results.
- Dunn Index (DI) [13, 21]:

$$\text{DI} = \frac{d_{\min}}{d_{\max}}, \tag{5}$$

where $d_{\min}$ is the minimal distance between two points in different clusters, and $d_{\max}$ is the maximal distance between the farthest two points in a cluster. By maximizing DI, we obtain better results. Thus, we use the reversed value of DI (1 − DI) in the objective function.

The extended feature, which considers different distance measures, is only effective for SSE, SC, and Dunn index.

Evaluation Measures

The framework includes a set of evaluation measures to evaluate the results obtained from running the framework. Given T as the true classes of N points and P as the predicted clusters of these points. The evaluation measures that are available at the time of writing this paper are as follows:

- Purity (P) [6]:

$$\text{Purity} = \frac{1}{N}\sum_{j=1}^{k}\max_i(|T_i \cap P_j|) \tag{6}$$

where P_j represents the points assigned to cluster j, k is the number of clusters, and T_i is the true assignments of points in cluster i.

- Entropy (E) [6]:

$$\text{Entropy} = \sum_{j=1}^{k} \frac{(|P_j|)}{n} E(P_j) \tag{7}$$

where $E(P_j)$ is the individual entropy of a cluster. Individual cluster entropy is calculated by Eq. (8):

$$E(P_j) = -\frac{1}{\log k} \sum_{i=1}^{k} \frac{|P_j \cap T_i|}{P_j} \log\left(\frac{|P_j \cap T_i|}{P_j}\right) \tag{8}$$

- Homogeneity score (HS) [60]:

$$\text{HS} = 1 - \frac{H(T|P)}{H(T)} \tag{9}$$

where $H(T)$ is the classes Entropy and $H(T|P)$ is the classes conditional Entropy. $H(T)$ and $H(T|P)$ are calculated as follows [60]:

$$H(T) = -\sum_{t=1}^{|T|} \frac{n_t}{N} \cdot \log\left(\frac{n_t}{N}\right) \tag{10}$$

$$H(T|P) = -\sum_{p=1}^{|P|} \sum_{t=1}^{|T|} \frac{n_{pt}}{N} \cdot \log\left(\frac{n_{pt}}{n_p}\right), \tag{11}$$

where n_t and n_p are the number of points of the true class t and the predicted cluster p, respectively. n_{pt} is the number of points of the true class t which are clustered to the predicted cluster p.

- Completeness Score (CS) [60]:

$$\text{CS} = 1 - \frac{H(P|T)}{H(P)}, \tag{12}$$

where $H(P)$ is the cluster Entropy and $H(P|T)$ is the clusters conditional Entropy. $H(P)$ and $H(P|T)$ are calculated as follows [60]:

$$H(P) = -\sum_{p=1}^{|P|} \frac{n_p}{N} \cdot \log\left(\frac{n_p}{N}\right) \tag{13}$$

$$H(P|T) = -\sum_{t=1}^{|T|} \sum_{p=1}^{|P|} \frac{n_{pt}}{N} \cdot \log\left(\frac{n_{pt}}{n_t}\right). \tag{14}$$

- V-measure (VM) [60]:

$$\text{VM} = 2 \cdot \frac{\text{HS} \cdot \text{CS}}{\text{HS} + \text{CS}} \tag{15}$$

- Adjusted mutual information (AMI) [65]:

$$\text{AMI} = \frac{\text{MI} - E[\text{MI}]}{\max(H(P), H(T)) - E[\text{MI}]} \tag{16}$$

where $H(P)$ and and $H(T)$ are the cluster Entropy (Eq. 13) and the class entropy (Eq. 10). MI is the Mutual Index which is calculated by [65]:

$$\text{MI} = \sum_{p=1}^{|P|} \sum_{t=1}^{|T|} \frac{n_{pt}}{N} \log\left(\frac{\frac{n_{pt}}{N}}{\frac{n_p}{N} \cdot \frac{n_t}{N}}\right) \tag{17}$$

$E[\text{MI}]$ is the Expected Mutual Index which is calculated by [59, 65]:

$$E(\text{MI}) = \sum_{p=1}^{|P|} \sum_{t=1}^{|T|} \sum_{n_{pt}=\max(0, n_p+n_t-N)}^{\min(n_p, n_t)} \frac{n_{pt}}{N} \log\left(\frac{N.n_{pt}}{n_p n_t}\right) \times \left(\frac{n_p! n_t! (N-n_p)! (N-n_t)!}{N! n_{pt}! (n_p - n_{pt})! (n_t - n_{pt})! (N - n_p - n_t + n_{pt})!}\right). \tag{18}$$

- Adjusted Rand Index (ARI) [24]:

$$\text{ARI} = \frac{\text{RI} - E[\text{RI}]}{\max(\text{RI}) - E[\text{RI}]}, \tag{19}$$

where RI is the Rand Index, $E[\text{RI}]$ is the Expected Rand Index, and max[RI] is the Maximum Rand Index. RI, $E[\text{RI}]$, and max[RI] are calculated by Eqs. (20), (21), and (22) [24, 56], respectively:

$$\text{RI} = \frac{a+b}{\binom{N}{2}} = \frac{\sum_{p,t} \binom{n_{pt}}{2}}{\binom{N}{2}} \tag{20}$$

$$E(\text{RI}) = E\left(\sum_{p,t} \binom{n_{pt}}{2}\right) = \frac{\sum_{p=1}^{|P|} \binom{n_p}{2} \sum_{t=1}^{|T|} \binom{n_t}{2}}{\binom{N}{2}} \tag{21}$$

$$\max[\text{RI}] = \frac{1}{2}\left[\sum_{p=1}^{|P|} \binom{n_p}{2} + \sum_{t=1}^{|T|} \binom{n_t}{2}\right] \tag{22}$$

where a is the number of pair of points located in the same true class t and clustered at the same predicted cluster p. b is the number of pair of points located in a different true class t and clustered at a different predicted cluster p. $\binom{N}{2}$ is the number of pair of points.

- Sum of squared error (SSE) [11]
- Total within cluster variance (TWCV) [46]
- Silhouette coefficient (SC) [13, 46]
- Davies–Bouldin (DB) index [14]
- Dunn Index (DI) [13, 21]

Table 1 Data sets properties showing the name, number of clusters, number of points, number of features, data set type, and source

ID	Data set	k	# points	# features	Type	Source
1	Aggregation	7	788	2	Artificial	University of Eastern Finland[3]
2	Aniso	3	1500	2	Artificial	scikit learn[1]
3	Appendicitis	2	106	7	Real	KEEL[5]
4	Balance	3	625	4	Real	UCI[2]
5	Backnote	2	1372	4	Real	UCI[2]
6	Blobs	3	1500	2	Artificial	scikit learn[1]
7	Blood	2	748	4	Real	UCI[2]
8	Circles	2	1500	2	Artificial	scikit learn[1]
9	Diagnosis II	2	120	6	Real	UCI[2]
10	Ecoli	5	327	7	Real	UCI[2]
11	Flame	2	240	2	Artificial	University of Eastern Finland[3]
12	Glass	6	214	9	Real	UCI[2]
13	Heart	2	270	13	Real	UCI[2]
14	Iris	3	150	4	Real	UCI[2]
15	Iris 2D	3	150	2	Real	UCI[2]
16	Ionosphere	2	351	344	Real	UCI[2]
17	Jain	2	373	2	Artificial	University of Eastern Finland[3]
18	Liver	2	345	7	Real	UCI[2]
19	Moons	2	1500	2	Artificial	scikit learn[1]
20	Mouse	3	490	2	Artificial	ELKI[4]
21	Pathbased	3	300	2	Artificial	University of Eastern Finland[3]
22	Seeds	3	210	7	Real	UCI[2]
23	Smiley	4	500	2	Artificial	naftaliharris[6]
24	Sonar	2	208	60	Real	UCI[2]
25	Varied	3	1500	2	Artificial	scikit learn[1]
26	Vary Density	3	150	2	Artificial	ELKI[4]
27	Vertebral2	2	310	6	Real	UCI[2]
28	Vertebral3	3	310	6	Real	UCI[2]
29	WDBC	2	569	30	Real	UCI[2]
30	Wine	3	178	13	Real	UCI[2]

The last five measures are discussed in the previous section and are used as evaluation measures. These five measures are the only ones considered if the user provided unsupervised datasets having no true labels.

Benchmark Data Sets

Most common and well-known data sets used for clustering can be found in the framework in which it can be extended to include other data sets. The list of the data sets that are available at the time of writing this paper are summarized in Table 1. As observed from the table, the data sets have different number of points, features, and clusters. The data sets are either real-world or artificial synthetic data sets. They are gathered from scikit learn,[1] UCI machine learning repository,[2] School of Computing at University of Eastern Finland,[3] ELKI,[4] KEEL,[5] and Naftali Harris Blog.[6] These data sets are extensively found in the literature for solving the clustering problem [12]. The selected data sets have varying number of points allowing different experiments to be performed on different volume of data. Different dimensionality of the data sets are also available, which is a critical issue in machine learning problems.

Datasets are available in the framework in two representations: supervised and unsupervised, where supervised

[1] http://scikit-learn.org/stable/datasets/index.html.

[2] https://archive.ics.uci.edu/ml/.

[3] http://cs.uef.fi/sipu/datasets/.

[4] https://elki-project.github.io/datasets/.

[5] https://sci2s.ugr.es/keel/datasets.php.

[6] https://www.naftaliharris.com/blog/visualizing-K-means-clustering/.

datasets contain the expected labels in the last column of the file, while unsupervised datasets do not contain this column.

Results Management

The results obtained by running the framework are gathered in three types of CSV files and two types of plots:

- Average results file: This file gives average results of the runs for performing each combination of an optimizer, objective function, and data set. Each item in the CSV file includes the average execution time, the average value of each external measure, and the average objective function value for each iteration. This file is important for two reasons: first, it shows how well the optimizers are performing. Second, it shows how the optimizer is enhancing the results by converging towards the optimal solution.
- Detailed results file: This file gives detailed results for each run. Each item in the CSV file includes the execution time, the value of each external measure, and the value for each iteration. Practitioners and researchers find this file useful as it allows them to compare the results of different runs for the same combination of an optimizer, objective function, and data sets.

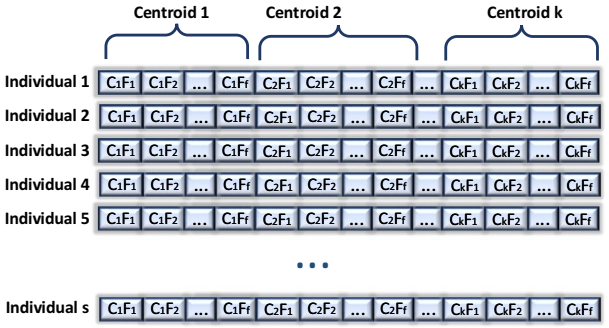


Fig. 1 Population of individuals at a specific iteration

- Best individual labels file: This file includes the values of the labels obtained from the best individual at the last iteration for each run. It is useful for the practitioners and researchers as it provides the final solution to the problem, which is the ultimate goal of using the framework for optimizing the clustering task.
- Convergence curve plot: Plots are generated for the convergence curves for each selected data sets, optimizers, and objective functions. Each plot represents the convergence curves for multiple optimizers having the values of the objective function at each iteration. This plot

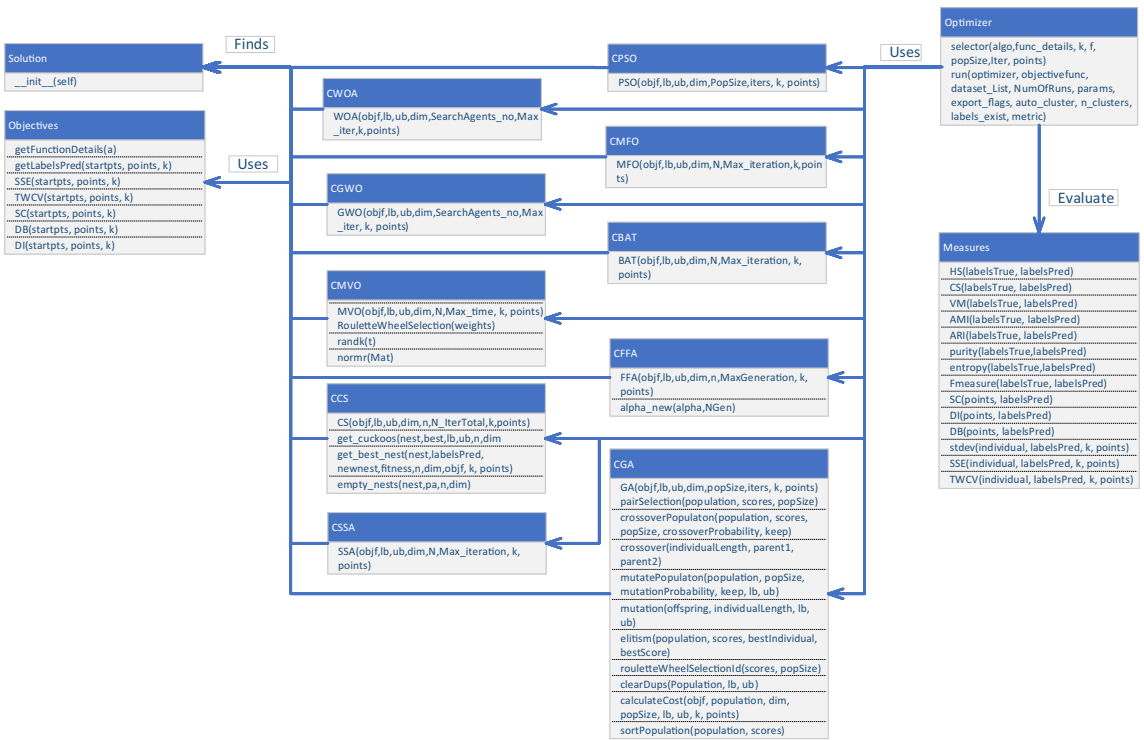


Fig. 2 Class diagram of EvoCluster

is important as it shows how the solution is enhancing through different iterations.

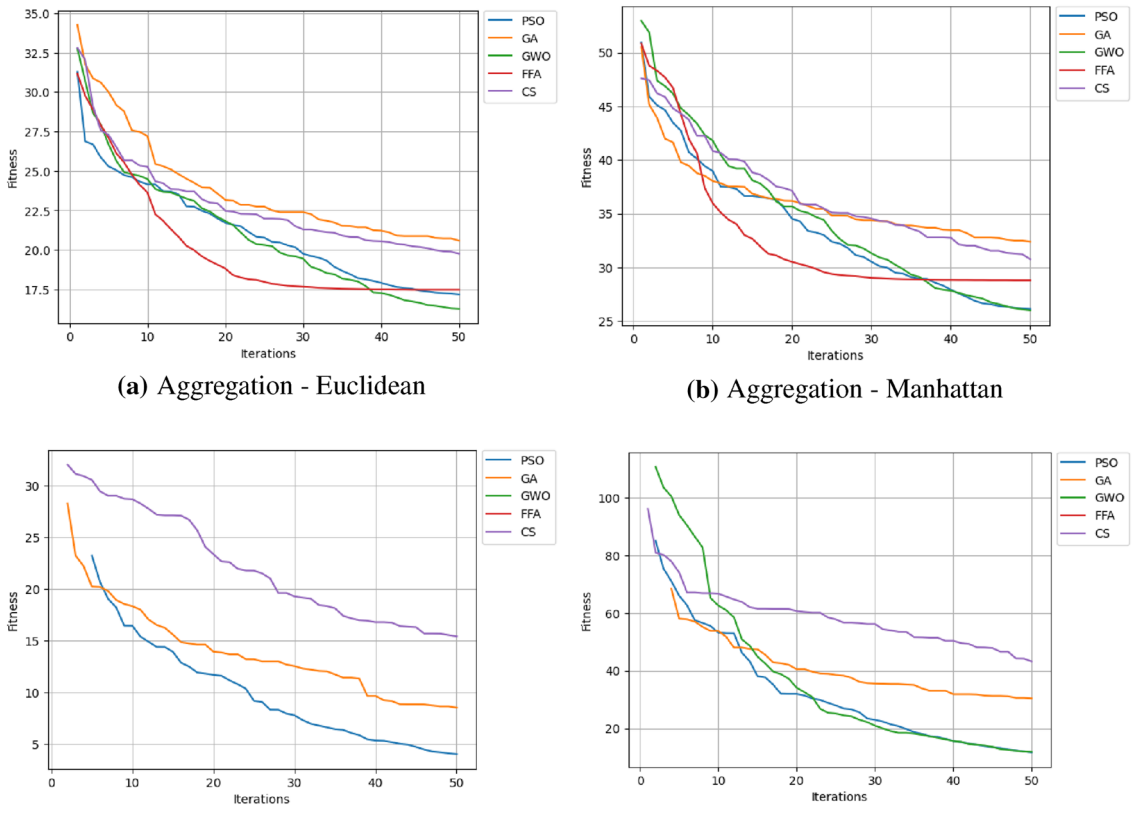
- Box plot: Plots are generated to represent the evaluation measures for each data set for several runs of the framework. Each box in a plot represents one of the selected optimizers. The interquartile range, best value, and worst value are represented as the box, the upper whiskers, and the lower whiskers, respectively [3]. This file helps in identifying the stability of the optimizer and showing the differences of the evaluation values between different runs.

Design Issues

The nature-inspired metaheuristic optimizers, which are included in the framework, use a population of individuals (s) at each iteration. Each individual represents one of the optimizers suggested clustering solution of centroids. Thus, consisting of the features (f) of each centroid for (k) clusters. Figure 1 shows how a population at a certain iteration is formed. For each individual in the group of s individuals in the population, there are k centroids having f features for each centroid.

In EvoCluster, populations are defined using the `Numpy` open-source Python package which is based on the `N-dimensional array` data structure. The `metrics` module of the `sklearn` package are used for the evaluation measures for HS, CS, VM, AMI, ARI, FM, SC, and DB. Distance measures are calculated using `scipy.spatial.distance` package. In addition, the `normalize` function of the `preprocessing` module in the `sklearn` package is used to normalize the values of the features for a data set to the interval [0, 1], to give similar weights to the features of the points. Each individual of a population in a certain iteration generates a corresponding vector of predicted labels which represents the cluster number for each point of the data set. This vector is evaluated using a selected objective function in the framework.

EvoCluster components and their relationships are illustrated as a class diagram in Fig. 2. EvoCluster contains fourteen classes which include ten classes for the metaheuristic algorithms and four other main classes. The main classes are `Optimizer`, `Solution`, `Objectives`, and



(a) Aggregation - Euclidean **(b)** Aggregation - Manhattan

(c) Iris - Euclidean **(d)** Iris - Manhattan

Fig. 3 Convergence curve plots for PSO, GA, GWO, FFA, and CS using SSE objective function and Elbow method for **a** aggregation with Euclidean distance; **b** aggregation with CityBlock/Manhattan distance; **c** Iris with Euclidean distance; and **d** Iris with CityBlock/Manhattan distance

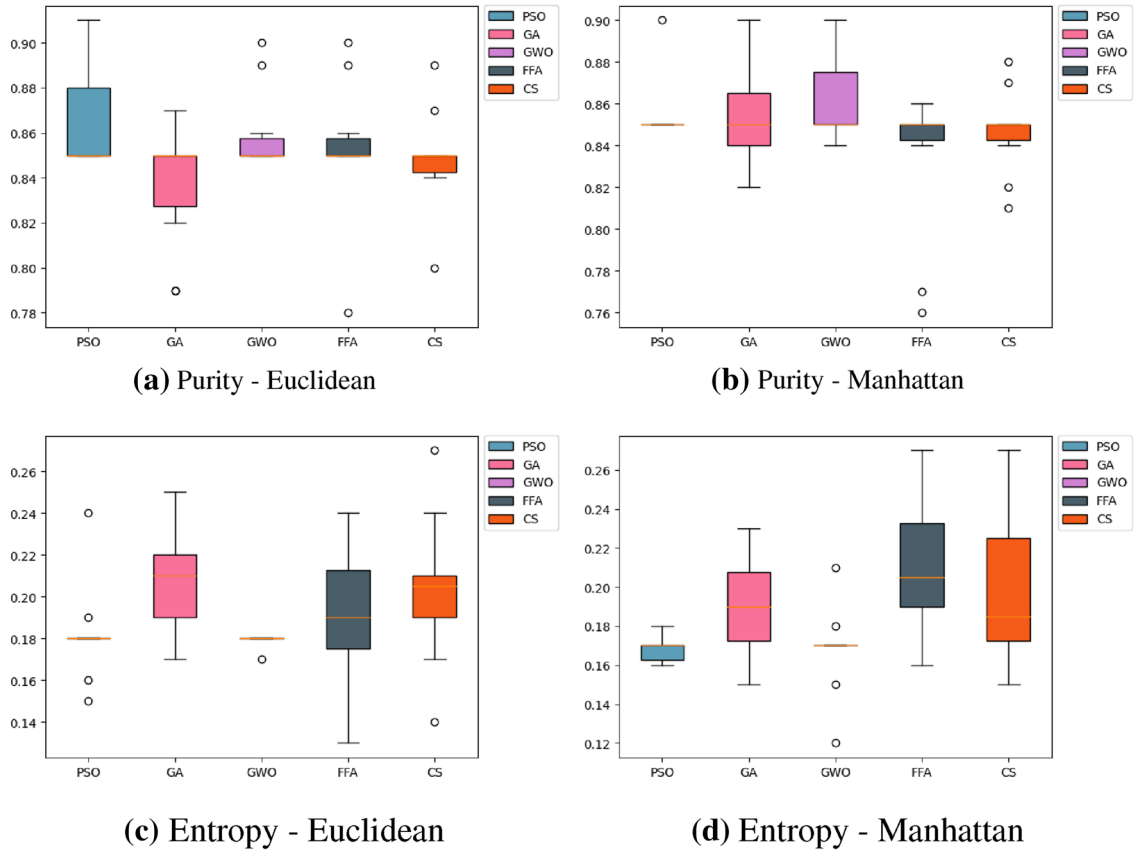


Fig. 4 Box plots of **a** purity with Euclidean distance; **b** purity with CityBlock/Manhattan distance; **c** entropy with Euclidean distance; and **d** entropy with CityBlock/Manhattan distance for PSO, GA, GWO, FFA, and CS using SSE objective function and Elbow method for the aggregation data set

`Measures`. The `Optimizer` class solves the partitional clustering task by using one of the metaheuristic algorithm which finds a `Solution` using an `Objective` function. This generates the global solution of the best clustering labels. The `Optimizer` then evaluates the labels using the `Measures` class.

Experiments and Visualizations

In this section, we show some examples of conducting some experiments using EvoCluster. The framework is run 10 times using a population size of 50 and iterations value of 100.

Figure 3 shows the convergence curve for selected optimizers which are PSO, GA, GWO, FFA, and CS using the SSE objective function for Aggregation and Iris datasets for different distance measures which are Euclidean and Manhattan. The convergence curve represents the values of the objective function over the course of iterations. The convergence curve shows the progress of the optimizers toward the optimal solution by minimizing the value of the objective function. As observed from the figure, the optimizers show different behaviors. In addition, the same algorithm behaves differently for different data sets and with different distance measures. Figure 4 shows the box plot for the selected optimizers which are PSO, GA, GWO, FFA, and CS using the SSE objective function for the Aggregation data set for Purity and Entropy evaluation measures for Euclidean and Manhattan distance measures. The box plot represents the range of values of different runs. It also shows the max, min, and mean values of the evaluation measure.

The experiments in this section show that EvoCluster framework facilitates the work of the practitioners and researchers by performing comparisons between different metaheuristic algorithms based on different configurations and evaluation metrics. In addition, the end-point results are also provided in EvoCluster in various formats including results files and plots, which allows practitioners and researchers observe different views of the results and helps them evaluate their choice of the optimizers and configuration settings.

Conclusion and Future Work

EvoCluster is an open-source framework implemented in Python language which provides so-far ten nature-inspired metaheuristic optimizers that are customized to solve partitional clustering tasks. EvoCluster provides the ability to select one of five well-known objective functions and twelve well-known evaluation measures. The framework is designed in a flexible way so that developers and researchers can customize it by implementing other optimizers, objective functions, and evaluation measures. In this paper, we extend EvoCluster framework with different distance measures for the objective function, different ways to specify the k value, and a user option to consider either supervised or unsupervised datasets. We also show some visualizations that are automatically generated based on the results of the conducted experiments using the extended EvoCluster framework in the form of the convergence curves and box plots. As future work, it is planned to support the framework with more well-regarded and recent metaheuristics. In addition, the framework will be supported with different initialization mechanisms that can support the performance of the optimizers, and other encoding mechanisms and problem formulations. Further, EvoCluster framework can be upgraded to work on parallel architectures for handling larger datasets and speeding up the performance of the metaheuristic algorithms.

Acknowledgements This work has been supported in part by: Ministerio español de Economía y Competitividad under project TIN2017-85727-C4-2-P (UGR-DeepBio).

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Achtert E, Kriegel HP, Zimek A. Elki: a software system for evaluation of subspace clustering algorithms. In: International conference on scientific and statistical database management. Springer. 2008. p. 580–585.
2. Al-Madi N., Aljarah I, Ludwig SA. Parallel glowworm swarm optimization clustering algorithm based on MapReduce. 2014 IEEE Symposium on Swarm Intelligence, Orlando, FL, USA, 2014. pp. 1–8. https://doi.org/10.1109/SIS.2014.7011794.
3. Aljarah I, Ala'M AZ, Faris H, Hassonah MA, Mirjalili S, Saadeh H. Simultaneous feature selection and support vector machine optimization using the grasshopper optimization algorithm. Cogn Comput. 2018;10(3):478–495.
4. Aljarah I, Ludwig SA. Parallel particle swarm optimization clustering algorithm based on MapReduce methodology. 2012 Fourth World Congress on Nature and Biologically Inspired Computing (NaBIC), Mexico City, Mexico, 2012, pp. 104–111. https://doi.org/10.1109/NaBIC.2012.6402247.
5. Aljarah I, Ludwig SA. MapReduce intrusion detection system based on a particle swarm optimization clustering algorithm. 2013 IEEE Congress on evolutionary computation, Cancun, Mexico, 2013, pp. 955–962. https://doi.org/10.1109/CEC.2013.6557670.
6. Aljarah I, Ludwig SA. A new clustering approach based on Glowworm Swarm Optimization. 2013 IEEE congress on evolutionary computation, cancun, Mexico, 2013, pp. 2642–2649. https://doi.org/10.1109/CEC.2013.6557888.
7. Aljarah I, Mafarja M, Heidari AA, Faris H, Mirjalili S. Clustering analysis using a novel locality-informed grey wolf-inspired clustering approach. Knowl Inf Syst. 2020;62(2):507–539.
8. Aljarah I, Mafarja M, Heidari AA, Faris H, Mirjalili S. Multiverse optimizer: theory, literature review, and application in data clustering. In: Nature-inspired optimizers. Springer; 2020. p. 123–141.
9. Beyer HG, Schwefel HP. Evolution strategies: a comprehensive introduction. Nat Comput. 2002;1(1):3–52. https://doi.org/10.1023/A:1015059928466.
10. Cahon S, Melab N, Talbi EG. Paradiseo: a framework for the reusable design of parallel and distributed metaheuristics. J Heuristics. 2004;10(3):357–80. https://doi.org/10.1023/B:HEUR.0000026900.92269.ec.
11. Chang DX, Zhang XD, Zheng CW. A genetic algorithm with gene rearrangement for k-means clustering. Pattern Recognit. 2009;42(7):1210–22.
12. Chang S, Shihong Y, Qi L. Clustering Characteristics of UCI Dataset. 2020 39th Chinese Control Conference (CCC), Shenyang, China, 2020, pp. 6301–6306. https://doi.org/10.23919/CCC50068.2020.9189507.
13. Chowdhury K, Chaudhuri D, Pal AK. A novel objective function based clustering with optimal number of clusters. In: Methodologies and application issues of contemporary computing framework. Springer, Singapore; 2018; pp. 23–32.
14. Davies DL, Bouldin DW. A cluster separation measure. IEEE Trans Pattern Anal Mach Intell. 1979;2:224–7.
15. Durillo JJ, Nebro AJ. jmetal: a java framework for multi-objective optimization. Adv Eng Softw. 2011;42:760–71.
16. Faris H, Aljarah I, Mirjalili S, Castillo P, Merelo J. EvoloPy: an Open-source Nature-inspired optimization framework in python. In: 2020 Proceedings of the 8th international joint conference on computational intelligence - Volume 3: ECTA, (IJCCI 2016) pp. 171–177. ISBN: 978-989-758-201-1. https://doi.org/10.5220/0006048201710177.
17. Finch H. Comparison of distance measures in cluster analysis with dichotomous data. J Data Sci. 2005;3(1):85–100.
18. Fortin FA, De Rainville FM, Gardner MA, Parizeau M, Gagné C. DEAP: evolutionary algorithms made easy. J Mach Learn Res. 2012;13:2171–5.
19. Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH. The weka data mining software: an update. ACM SIGKDD Explor Newsl. 2009;11(1):10–8.
20. Hartmut Pohlheim: Geatbx: the genetic and evolutionary algorithm toolbox for matlab (2006). http://www.geatbx.com/. Accessed 28 Feb 2021.
21. Hassani M, Seidl T. Using internal evaluation measures to validate the quality of diverse stream clustering algorithms. Vietnam J Comput Sci. 2017;4(3):171–83.
22. Holland J. Genetic algorithms. New York: Scientific American; 1992. p. 66–72.
23. Huang A. Similarity measures for text document clustering. In: Proceedings of the sixth New Zealand computer science research student conference (NZCSRSC2008), vol. 4. New Zealand: Christchurch; 2008. pp. 9–56.

24. Hubert L, Arabie P. Comparing partitions. J Classif. 1985;2(1):193–218.
25. Hughes EJ. Evolutionary multi-objective ranking with uncertainty and noise. In: International conference on evolutionary Multi-Criterion optimization. Springer, Berlin, Heidelberg; 2001. pp. 329–343.
26. Keijzer M, Merelo, JJ, Romero G, Schoenauer M. Evolving objects: a general purpose evolutionary computation library. In: International conference on artificial evolution (Evolution Artificielle). Springer, Berlin, Heidelberg; 2001. pp. 231–242.
27. Kennedy J, Eberhart R. Particle swarm optimization. Proceedings of ICNN'95 - International conference on neural networks. Perth, WA, Australia. 1995. pp. 1942–1948. https://doi.org/10.1109/ICNN.1995.488968.
28. Khurma RA, Aljarah I, Sharieh A, Mirjalili S. Evolopy-fs: An open-source nature-inspired optimization framework in python for feature selection. In book: Evolutionary machine learning techniques. Springer, Singapore. 2020. pp. 131–173.
29. Kingrani SK, Levene M, Zhang D. Estimating the number of clusters using diversity. Artif Intell Res. 2018;7(1):15–22.
30. Klawonn F, Keller A. Fuzzy clustering based on modified distance measures. In: International symposium on intelligent data analysis. Springer; 1999. p. 291–301.
31. Korošec P, Šilc JA. distributed ant-based algorithm for numerical optimization. In: Proceedings of the 2009 workshop on Bio-inspired algorithms for distributed systems-BADS 09. Association for computing machinery (ACM). 2009. p. 37–44. https://doi.org/10.1145/1555284.1555291.
32. Krishna TS, Babu AY, Kumar RK. Determination of optimal clusters for a Non-hierarchical clustering paradigm K-Means algorithm. In: Proceedings of international conference on computational intelligence and data engineering; Springer, Singapore. 2018. pp. 301–316.
33. Kumar S, Pant M, Kumar M, Dutt A. Colour image segmentation with histogram and homogeneity histogram difference using evolutionary algorithms. Int J Mach Learn Cybern. 2018;9(1):163–183.
34. Liu A, Su Y, Nie W, Kankanhalli MS. Hierarchical clustering multi-task learning for joint human action grouping and recognition. IEEE Trans Pattern Anal Mach Intell. 2017;39(1):102–14.
35. Magni P, Ferrazzi F, Sacchi L, Bellazzi R. Timeclust: a clustering tool for gene expression time series. Bioinformatics. 2007;24(3):430–2.
36. Matthew Wall: Galib: A c++ library of genetic algorithm components (1996). http://lancet.mit.edu/ga/. Accessed 28 Feb 2021.
37. Mhembere D, Zheng D, Priebe CE, Vogelstein JT, Burns R. clusternor: a numa-optimized clustering framework. 2019. arXiv preprint arXiv:1902.09527
38. Mirjalili S. Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm. Knowl Based Syst. 2015;89:228–49. https://doi.org/10.1016/j.knosys.2015.07.006.
39. Mirjalili S, Gandomi AH, Mirjalili SZ, Saremi S, Faris H, Mirjalili SM. Salp swarm algorithm: a bio-inspired optimizer for engineering design problems. Adv Eng Softw. 2017;114:163–91.
40. Mirjalili S, Lewis A. The whale optimization algorithm. Adv Eng Softw. 2016;95:51–67.
41. Mirjalili S, Mirjalili SM, Hatamlou A. Multi-verse optimizer: a nature-inspired algorithm for global optimization. Neural Comput Appl. 2016;27(2):495–513. https://doi.org/10.1007/s00521-015-1870-7.
42. Mirjalili S, Mirjalili SM, Lewis A. Grey wolf optimizer. Adv Eng Softw. 2014;69:46–61.
43. Nanda SJ, Panda G. A survey on nature inspired metaheuristic algorithms for partitional clustering. Swarm Evol Comput. 2014;16:1–18.
44. Paukkeri MS, Kivimäki I, Tirunagari S, Oja E, Honkela T. Effect of dimensionality reduction on different distance measures in document clustering. In: International conference on neural information processing. Springer; 2011. p. 167–176.
45. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, *et al*. Scikit-learn: machine learning in python. J Mach Learn Res. 2011;12(Oct):2825–30.
46. Peng P, Addam O, Elzohbi M, Özyer ST, Elhajj A, Gao S, Liu Y, Özyer T, Kaya M, Ridley M, *et al*. Reporting and analyzing alternative clustering solutions by employing multi-objective genetic algorithm and conducting experiments on cancer data. Knowl Based Syst. 2014;56:108–22.
47. Prakash J, Singh PK. Particle swarm optimization with k-means for simultaneous feature selection and data clustering. In: 2015 second international conference on soft computing and machine intelligence (ISCMI). IEEE; 2015. p. 74–78.
48. Qaddoura R, Al Manaseer W, Abushariah MA, Alshraideh MA. Dental radiography segmentation using expectation-maximization clustering and grasshopper optimizer. Multimed Tools Appl. 2020;79:22027–45.
49. Qaddoura R, Faris H, Aljarah I. An efficient clustering algorithm based on the k-nearest neighbors with an indexing ratio. Int J Mach Learn Cybern. 2020;11(3):675–714.
50. Qaddoura R, Faris H, Aljarah I. An efficient evolutionary algorithm with a nearest neighbor search technique for clustering analysis. J Ambient Intell Humaniz Comput. 2020;1–26.
51. Qaddoura R, Faris H, Aljarah I, Castillo PA. Evocluster: an open-source nature-inspired optimization clustering framework in python. In: International conference on the applications of evolutionary computation (Part of EvoStar). Springer; 2020. p. 20–36.
52. Qaddoura R, Faris H, Aljarah I, Merelo J, Castillo P. Empirical evaluation of distance measures for nearest point with indexing ratio clustering algorithm. In: Proceedings of the 12th International joint conference on computational intelligence - Vol 1. NCTA, pp. 430-438. ISBN 978-989-758-475-6 2020. https://doi.org/10.5220/0010121504300438.
53. Qaddoura R, Aljarah I, Faris H, Mirjalili S. A grey Wolf-Based clustering algorithm for medical diagnosis problems. In: Aljarah I, Faris H, Mirjalili S. (eds) Evolutionary data clustering: algorithms and applications. Algorithms for intelligent systems. Springer, Singapore. 2021. pp. 73–87. https://doi.org/10.1007/978-981-33-4191-3_3.
54. Qaddoura R, Aljarah I, Faris H, Almomani I. A. classification approach based on evolutionary clustering and its application for ransomware detection. In: Aljarah I, Faris H, Mirjalili S. (eds) Evolutionary data clustering: algorithms and applications. Algorithms for intelligent systems. Springer, Singapore. 2021. pp. 237–248. https://doi.org/10.1007/978-981-33-4191-3_11.
55. Raitoharju J, Samiee K, Kiranyaz S, Gabbouj M. Particle swarm clustering fitness evaluation with computational centroids. Swarm Evol Comput. 2017;34:103–118.
56. Rand WM. Objective criteria for the evaluation of clustering methods. J Am Stat Assoc. 1971;66(336):846–50.
57. Risso D, Purvis L, Fletcher RB, Das D, Ngai J, Dudoit S, Purdom E. clusterexperiment and rsec: a bioconductor package and framework for clustering of single-cell and other large gene expression datasets. PLoS Comput Biol. 2018;14(9):e1006378.
58. Robles-Berumen H, Zafra A, Fardoun HM, Ventura S. Leac: an efficient library for clustering with evolutionary algorithms. Knowl Based Syst. 2019;179:117–9.
59. Romano S, Vinh NX, Bailey J, Verspoor K. Adjusting for chance clustering comparison measures. J Mach Learn Res. 2016;17(1):4635–66.

60. Rosenberg A, Hirschberg J. V-measure: a conditional entropy-based external cluster evaluation measure. EMNLP-CoNLL. 2007;7:410–20.
61. Sheikh RH, Raghuwanshi MM, Jaiswal AN. Genetic algorithm based clustering: a survey. In: First international conference on emerging trends in engineering and technology. IEEE; 2008. p. 314–319.
62. Shi Y, Eberhart R. A modified particle swarm optimizer. In: 1998 IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence (Cat. No. 98TH8360). IEEE; 1998. p. 69–73.
63. Shukri S, Faris H, Aljarah I, Mirjalili S, Abraham A. Evolutionary static and dynamic clustering algorithms based on multi-verse optimizer. Eng Appl Artif Intell. 2018;72:54–66.
64. Vergara VM, Salman M, Abrol A, Espinoza FA, Calhoun VD. Determining the number of states in dynamic functional connectivity using cluster validity indexes. J Neurosci Methods. 2020;337:108651.
65. Vinh NX, Epps J, Bailey J. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. J Mach Learn Res. 2010;11(Oct):2837–54.
66. Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, Burovski E, Peterson P, Weckesser W, Bright J, van der Walt SJ, Brett M, Wilson J, Millman KJ, Mayorov N, Nelson ARJ, Jones E, Kern R, Larson E, Carey C, Polat İ, Feng Y, Moore EW, ErPlas JV, Laxalde D, Perktold J, Cimrman R, Henriksen I, Quintero EA, Harris CR, Archibald AM, Ribeiro AH, Pedregosa F, van Mulbregt P, Contributors S. SciPy 1.0: fundamental algorithms for scientific computing in Python. Nat Methods. 2020;17:261–72. https://doi.org/10.1038/s41592-019-0686-2.
67. Vrbančič G, Brezočnik L, Mlakar U, Fister D, Fister I Jr. Niapy: Python microframework for building nature-inspired algorithms. J Open Sour Softw. 2018;3:613.
68. Wagner S, Affenzeller M. The heuristiclab optimization environment. Tech. rep., University of Applied Sciences Upper Austria (2004). http://dev.heuristiclab.com/trac.fcgi/. Accessed 28 Feb 2021.
69. Wilson GC, Mc Intyre A, Heywood MI. Resource review: three open source systems for evolving programs-lilgp, ecj and grammatical evolution. Genet Program Evol Mach. 2004;5(1):103–5.
70. Wiwie C, Baumbach J, Röttger R. Comparing the performance of biomedical clustering methods. Nat Methods. 2015;12(11):1033.
71. Yang XS. Firefly algorithm, stochastic test functions and design optimisation. Int J Bioinspired Comput. 2010;2(2):78–84. https://doi.org/10.1504/IJBIC.2010.032124.
72. Yang XS. A new metaheuristic bat-inspired algorithm. In: González JR, Pelta DA, Cruz C, Terrazas G, Krasnogor N, editors. Nature inspired cooperative strategies for optimization (NICSO 2010). Berlin: Springer; 2010. p. 65–74. https://doi.org/10.1007/978-3-642-12538-6_6.
73. Yang XS, Deb S. Cuckoo search via levy flights. In: World congress on nature biologically inspired computing, NaBIC; Coimbatore, India; 2009. p. 210–214. https://doi.org/10.1109/NABIC.2009.5393690