Research Article

Nature-inspired metaheuristic techniques for automatic clustering: a survey and performance study

Absalom E. Ezugwu[1]

Received: 29 November 2019 / Accepted: 17 January 2020 / Published online: 25 January 2020

Abstract
The application of several swarm intelligence and evolutionary metaheuristic algorithms in data clustering problems has in the past few decades gained wide popularity and acceptance due to their success in solving and finding good quality solutions to a variety of complex real-world optimization problems. Clustering is considered one of the most important data analysis techniques in the domain of data mining. A clustering problem refers to the partitioning of unlabeled data objects into a certain number of clusters based on their attribute values or features, with the objective of maximizing intra-clusters homogeneity and inter-cluster heterogeneity. This paper presents an up-to-date survey of major nature-inspired metaheuristic algorithms that have been employed to solve automatic clustering problems. Further, a comparative study of several modified well-known global metaheuristic algorithms is carried out to solve automatic clustering problems. Also, three hybrid swarm intelligence and evolutionary algorithms, namely, particle swarm differential evolution algorithm, firefly differential evolution algorithm and invasive weed optimization differential evolution algorithm, are proposed to deal with the task of automatic data clustering. In contrast to many of the existing traditional and evolutionary computational clustering techniques, the clustering algorithms presented in this paper do not require any predetermined information or prior-knowledge of the dataset that is to be classified, but rather they are capable of spontaneously identifying the optimal number of partitions of the data points during the course of program execution. Forty-one benchmarked datasets that comprise eleven artificial and thirty real world datasets are collated and utilized to evaluate the performances of the representative nature-inspired clustering algorithms. According to the extensive experimental results, comparisons and statistical significance, the firefly algorithm appeared to be more appropriate for better clustering of both low and high dimensional data objects than were other state-of-the-art algorithms. Further, an experimental study demonstrates the superiority of the three proposed hybrid algorithms over the standard state-of-the-art methods in finding meaningful clustering solutions to the problem at hand.

Keywords Automatic clustering · Clustering analysis · Differential evolution · Particle swarm optimization · Firefly algorithm · Invasive weed optimization · Artificial bee colony · Bees algorithm · Biogeography-based optimization · Harmony search · Symbiotic organisms search · Teaching–learning-based optimization · DB validity index · CS validity index

✉ Absalom E. Ezugwu, Ezugwua@ukzn.ac.za | [1]School of Computer Science, University of KwaZulu-Natal, King Edward Road, Pietermaritzburg Campus, Pietermaritzburg, KwaZulu-Natal 3201, South Africa.

1 Introduction

Clustering is the process of organizing objects into groups whose members have similar traits to one other. It segregates a collection of objects by partitioning it into groups called "clusters", such that objects in a cluster are more similar to each other than to objects in another cluster [1]. A clustering task is said to be an instance of unsupervised learning if there is no external information provided with the objects, such as class labels [2]. The most common unsupervised learning method is cluster analysis, which is used for exploratory data analysis to find hidden patterns or grouping in data [3]. Data clustering has wide application, because it uses valuable information that may have been hidden within groups, and it is applied in many fields such as engineering, computer science, medical science, earth science, life science, economics and bioinformatics [2]. Some specific interesting real-world applications of clustering are given below [2, 4].

- Marketing: finding groups of customers with similar behavior given a large database of customer data containing their profiles and past purchase records,
- Biology: classification of plants and animals given their features,
- Libraries: book ordering,
- Insurance: identifying groups of motor insurance policy holders with a high average claim cost, identifying frauds,
- Data Mining: analyzing data from different perspectives and summarizing it into useful information in order to improve decision-making, reduce cost, develop new products and so on,
- Medicine: classification of patients as self-care patients or complete-care patients and patient illness as chronic or non-chronic in order to improve service delivery,
- Community detection: as has been recently developed, detection of community commonalities in real-world graphs such as social networks (e.g., LinkedIn, Facebook, Twitter, etc.), biological networks, and web graphs.

In the past few decades, clustering has proven to be vital in the exploration and analysis of data. More so, clustering analysis has also been applied to problems in machine learning, artificial intelligence, pattern recognition, web mining, spatial database analysis, image segmentation, software evolution, and anomaly detection [2]. Despite the ubiquity and the great importance of clustering, clustering is a difficult problem to solve, and as a consequence, tremendous research efforts have been devoted to designing new clustering algorithms since Karl Pearson's earliest research in [5]. Some of the reasons that make clustering difficult include: (1) clustering is NP-hard, (2) there are no widely accepted theories for clustering that could be applied to every type of dataset, and (3) the definition of a cluster is determined by the characteristics of the dataset and the understanding of the user and is therefore, quite arbitrary [2].

There are several clustering algorithms that have been implemented in the recent years, which can be grouped into hierarchical and partitional methods [6, 7]. Figure 1 shows the taxonomy of clustering algorithms. These groupings are defined strictly based on the analyzed datasets arrangement. For example, in hierarchical clustering methods, data are arranged in a hierarchical tree structure based on the similarity among data points, while the total shape and size of clusters are ignored. In order words, at the onset of the clustering analysis process, a data point can only be assigned to one cluster at a time, which, in this method, makes the formation of a cluster static. In the case of partitional methods, the dataset inside a set of separate clusters are directly analyzed, such that the dissimilarity found for intra-cluster is minimized and maximized for inter-cluster. Although, these two clustering methods are still used today, there usage and effectiveness centers on having a prior knowledge of the number of clusters in a dataset [8]. Therefore, the existing methods cannot be applied to solve real world problems, wherein prior information regarding the number of naturally occurring groups in the data is often not available and determining the optimal amount of clusters for such dataset becomes very difficult. More so, this is the reasons why the data clustering problem is classified as an NP-hard problem. To address this challenge, automatic clustering of datasets was introduced [6, 9]. Automatic clustering of a dataset involves spontaneously determining the number and structure of clusters in a dataset, without any prior information of the datasets attribute values [8]. However, the implementation of automatic clustering for large volume dataset is an extremely challenging task, especially when clusters are very different in terms of shape, size, or density and more so when there is an overlap between groups or clusters [8, 10].

The most popular and simplest clustering algorithm is the k-mean, which has been used to solve a wide range of clustering problems. The algorithm is very efficient and effective, due specifically to its linear time complexity, but also because the deterministic local search used in its implementation usually converges the algorithm to the nearest local optima. However, in recent years, a few automatic data clustering algorithms have been implemented, most of which are inspired by either natural or physically occurring phenomena, among which can be included genetic algorithm (GA) [11, 12], differential evolution (DE)

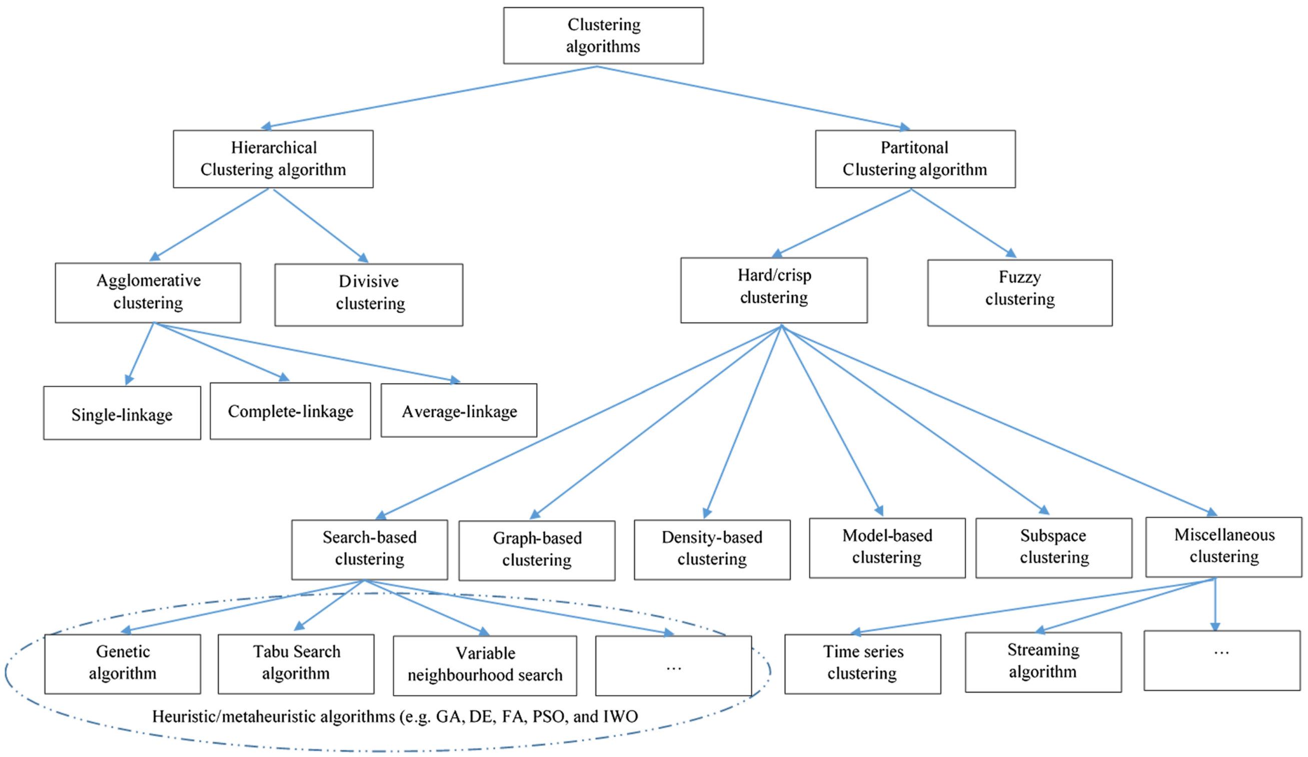


Fig. 1 A taxonomy of clustering algorithms

[13], particle swarm optimization (PSO) [14, 15], gravitational search algorithm (GSA) [16], symbiotic organisms search (SOS) [17], bee colony optimization algorithm (BCA) [18, 19], invasive weed optimization (IWO) [20], and bacterial evolutionary algorithm (BEA) [21]. More detailed discussion of some related algorithm implementations applied to automatic clustering are presented in Sect. 2 below. For further reading, interested researchers are referred to [10]. The automatic clustering algorithms can be classified into two main groups namely, evolutionary and swarm intelligence metaheuristic algorithms. The evolutionary algorithms include GA and DE, while the swarm intelligence algorithms include PSO, SOS, BCA, and IWO. In both the evolutionary and swarm intelligence implementation methods, clustering analysis is considered to be an optimization tasks, which involves the minimization of dissimilarity within a cluster and maximization of dissimilarity between clusters [10]. The aforementioned two classes of metaheuristic methods are preferred to the traditionally-based approaches for solving high dimensional data clustering problems because of their superior convergence speeds and ability to obtain good quality solutions. Furthermore, the two automatic clustering algorithms (GA and DE) belong to the search-based clustering method highlighted with a dotted oval in Fig. 1 above. It is equally noteworthy to mention that this current study utilizes a search-based algorithm with specific focus on both the evolutionary and swarm intelligence algorithms.

Nature-inspired metaheuristics, in particular GA, DE, PSO, firefly algorithm (FA), IWO, and GA, have been used by researchers to solve a wide range of continuous and discrete combinatorial optimization problems. For instance, the GA is an evolutionary algorithm with a wide area of applications, which has been employed to solve several real-world optimization problems such as location problems [22], flow shop problems [23], and data clustering problems [6, 11, 12]. The DE, as another evolutionary algorithm, has received considerable research interest, where it has been applied to solve difficult optimization problems in computer science [24] and engineering [25, 26]. The DE and its hybrid variants have equally been used to solve many unsupervised data clustering problems [27, 28]. The PSO has several important applications with high profile successes in terms of good quality solutions and convergence speed. It has been applied to solve problems such as hierarchical and partitional data clustering [29, 30]. The FA is another high profile metaheuristic algorithm that has dozens of successful applications recorded in solving many complex engineering problems [31, 32], data clustering problems [33], and unrelated parallel machine scheduling problems [34]. The IWO is another interesting algorithm that has found application in solving difficult problems like the travelling salesman problem [35] and design of antenna configuration [36]. It is noteworthy to mention that these five basic algorithms stand out in their ability to obtain good quality solutions for complex and large-scale problems, short running times, ease of adaptability, and implementation simplicity.

This study investigates the application of five modified standard metaheuristic algorithms namely, GA, DE, PSO, FA and IWO to solve automatic data clustering problems. As stated, they have proved their worth already. Furthermore, three hybrid automatic data clustering algorithms are implemented. The particle swarm optimization differential evolution (PSODE), firefly algorithm differential evolution (FADE), and invasive weed optimization differential evolution (IWODE) algorithms are also considered. On the basis of improving the qualities of clustering results and convergence speeds of the modified representative algorithms. However, the main goal of the study centers on conducting detailed comparative performance evaluations of several metaheuristic algorithms in order to determine which of these algorithms have better performance in terms of convergence speed and ability to obtain good quality clustering solutions. The computational experiment carried out is based on a systematic evaluation of the seven proposed algorithms implemented in the course of the study on forty-one standard benchmark clustering dataset problems. The experimental results show that the FA and its hybrid counterpart, that is the FADE algorithm, not only achieve superior solution accuracy, but also achieve higher levels of stabilities than the other compared algorithms. This paper technically contributes with five main aspects:

1. A critical survey of existing literature on both single-objective and multi-objective, nature-inspired metaheuristics applied to automatic clustering analysis problems.
2. A comprehensive evaluation of five nature-inspired metaheuristic algorithms namely, PSO, FA, IWO, GA, and DE for the tasks of solving automatic data clustering problems. Further performance evaluation comparison with an additional six new generation metaheuristic algorithms are investigated and possible outcomes reported subsequently.
3. A proposal of three hybrid metaheuristic algorithms for solving the automatic data clustering task; these algorithms include PSODE, FADE, and IWODE. Notably, the proposed algorithms are able to conveniently handle high density and high dimensionality datasets.
4. For a fairer comparison, eight new generation metaheuristics algorithms are implemented for automatic data clustering and compared according to the number of function evaluation consumed by the individual algorithms.
5. Compilation of forty-one (41) synthetic and ground truth benchmark datasets for easy evaluation of the proposed algorithms. These datasets are often used by researchers to evaluate the performance of their new clustering algorithms. This list (which cannot be easily collated from a single source) provides an easily accessible collection of standard benchmark datasets for clustering analysis. More so, the dataset can be adopted as standard problems for evaluating new proposed metaheuristic algorithms in the field of data-mining.

It is noteworthy to mention that the current study only used the existing standard clustering indices to compute fitness functions and to validate the practicality and performance capability of the eight representative nature-inspired clustering algorithms. Therefore, the study does not focus on evaluating the validity indices. Finally, it is equally important to specifically state here that despite the relevance of these five representative algorithms, to the best of our knowledge, no single paper on the comprehensive performance analysis study of several nature-inspired metaheuristics with their hybrid variants for automatic clustering has been published.

The rest of the paper is organized as follows: Sect. 2 presents a detailed literature review of previous work done in the field of swarm and evolutionary automatic clustering techniques. Section 3 introduces clustering problem preliminaries with emphases on the clustering validity index and similarity measure. Section 4 describes the methodologies of the representative algorithms, specifically, GA, DE, PSO, FA, and IWO, and outlines the implementation procedures for the proposed hybrid algorithms. Section 5 describes the experimental study. The section also covers discussions on synthetic and real-world datasets, experimental setups, numerical results, and comparison of proposed algorithms with literature results. Finally, concluding remarks and statements for future directions are suggested in Sect. 6.

2 Literature review

A simple heuristic algorithm that can automatically detect any number of well separated clusters has been described by Chowdhury et al. [37]. The clusters may be of any shape e.g. convex and non-convex, and so the algorithms differs from the existing clustering algorithms, which assume a value for the number of clusters or a particular cluster structure with predefined features. The algorithm proposed in [37] draws inspiration from the foraging behavior of ants and it iteratively partitions a dataset based on its proximity matrix. Analysis of the runtime complexity showed that the proposed algorithm runs in quadratic time, which depends on the size of the dataset. A novel technique was afterwards formulated by modifying the traditional sequential schemes with some simple strategies, which can be attributed to the movements of ants. The ant dynamic-based clustering algorithm was compared with other automatic clustering hybrid algorithms such as the genetic algorithm (GA)-CGUK proposed in [38], the particle swarm optimization (PSO)-DCPSO proposed in [39], and the fuzzy ant-based clustering algorithm. The ant based algorithm consistently outperformed the other methods in terms of quality of solutions and computational cost. In a comparison, other methods were also able find a good number of clusters, except for the fuzzy ant-based algorithm, which required the correct number of clusters be supplied a priori. Nevertheless, the ant dynamic-based system result had better cluster compactness,

Differential evolution (DE), which is one of the most robust evolutionary algorithms, has over time proved able to solve clustering problems successfully and efficiently, in that, its algorithmic design is able to automatically assign clusters and has less computational time while retaining accuracy in test results. However, there have been consistent improvements on the DE algorithm to better bridge the anomalies that might come with its application in other field of studies. Presented in [40] is an automatic differential evolution fuzzy clustering algorithm for the automatic grouping of image pixels into different homogenous regions, which also does not require the number of clusters to be stated a priori. The optimization problem here was formulated as a fuzzy clustering task in the intensity space of an image. The improved DE was compared with fuzzy variable string genetic algorithm (FVGA) and the classical fuzzy c-means algorithm (FCM) over a range of test suite dataset problems comprised of ordinary grayscale images and remote sensing satellite images. Performance evaluation by the authors in [40] was judged according to the quality of solution obtained by the respective algorithms, their ability to find an optimal number of clusters and computational cost. According to the results presented in [40], the FCM algorithm failed to handle an unknown number of clusters, while the improved DE and FVGA produced better results even with clustering of high dimensionality. The authors carried out further statistical analysis tests and showed that the improved DE outperformed both FVGA and FCM in terms of speed, accuracy and robustness.

A genetic based automatic clustering algorithm for unknown K (AGCUK) was proposed in [41]. In this algorithm, noise selection and division-absorption mutation were designed to have a balance between selection pressure and population diversity. The validity of the number of clusters was measured by adopting the Davies–Bouldin (DB) index. The experiments exploited both artificial and real-life datasets. The AGCUK algorithm showed effectiveness in automatically evolving the number of clusters and providing cluster partitioning. The chromosomes (as in the genetic algorithm) of the AGCUK algorithm are made up of real numbers, which represent the coordinates of the cluster centers. The AGCUK algorithm is similar to the GA, in that, AGCUK also has the 'noising selection stage' (adopted to avoid the selected population from being occupied by several fitter individuals and to maintain population diversity) and 'division-absorption mutation' stage (introduced to determine the best candidate and other candidates) but they differ in terms of crossover operator. In the experimentation, the AGCUK algorithm was able to keep the balance between selection pressure and population diversity created and neglected by the standard GA; thus filling a knowledge gap. Noising selection was also incorporated into the design of the algorithm to maintain diversity, so as to avoid the solution search process being trapped into local minima, which can occur in the classical GA, while the division-absorption mutation was designed to maintain selection pressure so as to accelerate the convergence of the clustering algorithm. A super individual in the population after the selection operation is regarded as the best solution candidate, meaning that the stronger super individual is, the higher the selection pressure, and the weaker the super individual is, the lower the selection pressure. The AGCUK algorithm was evaluated in comparison to the other clustering techniques presented by Bandyopadhyay and Maulik [42, 43], Lai [44] and Lin et al. [45]. It was observed that the AGCUK found the correct number of clusters and provided optimal clustering partition, although its run time was greater than for the method proposed in Lin et al. [45]. The methods used by Bandyopadhyay and Maulik [42, 43] and Lai [44], were unable to give accurate cluster partition within the defined number of generations. The limitation of the AGCUK algorithm is the high computational cost required to produce solutions of better quality as compared to that of the Lin et al. method, and it may also accept bad individuals in its noising selection stage during the evolution process. Conclusively, AGCUK algorithm provided an accurate number of clusters for artificial and real-life datasets, with lower misclassified rates than shown by the other compared methods. AGCUK and the Bandyopadhyay and Maulik [42, 43] algorithms, as well as the Lai [44] methods produced the same time complexity. Also, due to the noising selection of AGCUK algorithm, the run time was optimized to get a better quality clustering cost although, it may absorb bad individuals in the process. This finally inferred that, by increasing the population size of AGCUK method against Lin et al.'s algorithm, the former outperformed the latter.

The author suggested that further important research may be carried out regarding improvement of the convergence speed of the AGCUK algorithm, without decreasing its search capability.

In [46], improved versions of the DE algorithm and GA were employed to solve the automatic fuzzy clustering problem in a multi-objective approach. The existing study also compared the performance of a hybrid genetic algorithm and differential evolution algorithm (GADE) over a fuzzy clustering problem, whereby a case of two (2) conflicting fuzzy validity indexes were being simultaneously optimized. Further in the study, the conjugation of GADE was made in comparison and contraction with two (2) other prominent multi-objective schemes (MOCK and NSGA II). For the computation experiment, six artificial datasets and two real-life datasets were used for testing. Numerical results showed that GADE produced a better clustering outcome than did the multi-objective schemes.

In [18], the shortcomings and weaknesses in automatic kernel clustering was addressed by the proposal of a new automatic clustering algorithm based on the bee colony optimization algorithm, referred to as AKC-BCO. This algorithm helps to determine the appropriate number of clusters and also assigns the correct number of clusters to data points. This is achieved by the kernel function, which enhances clustering capability to the said data points. AKC-BCO was furthermore applied to a real-world scenario of a medical problem (prostate cancer case), as well as being tested on seven of the benchmark datasets (Iris, Wine, Glass, Vowel, R15, D31 and bank marketing) to validate its effectiveness and accuracy. The resulting tests showed AKC-BCO to be more stable and accurate than were other automatic clustering algorithms.

The AKC-BCO works so that an initial population is generated by employed bees, and then the onlooker bees implement the neighborhood searches. Eventually, a global search is implemented by scout bees which is done through an iterative search until an optimal solution is obtained. Although in comparison to other clustering methods, the AKC-BCO took longer computational time, which is due to the AKC-BCO adopting, in each of its iterations, three different roles for bees (the employed bees, the onlooker bees and the scout bees) and for each iteration each bee is updated with each data element that has been assigned to the nearest active cluster. AKC-BCO also proved from test results that it converges better with few number of iterations than did the compared evolutionary algorithms. From the results for mean and standard deviation, AKC-BCO performed better than the other three evolutionary algorithms on all the individual benchmark datasets. In the case of the prostate cancer dataset, cells in the prostate can mutate and multiply out of control and can metastasize to other body organs. A stepwise regression approach was used in this case and it showed that biopsy, initial PSA and DRE readings are critical to the accuracy of a prognosis, which served as the clustering features. Since the three data types (biopsy, initial PSA and DRE) served as the number of clusters, DCPSO, AKC-MEPSO, DCPG and AKC-BCO algorithms were directly applied on them. Nevertheless, in comparison with DCPSO, AKC-MEPSO, and DCPG, the AKC-BCO had better improvement performance ratio, better stability, and greater accuracy compared to others and also converged better. It also showed a better survival chance for prostate patients. Although, the result seemed as accurate as those from the AKC-MEPSO algorithm, with both having a close outcome, AKC-BCO predicted a higher survival chance for patients and it also clustered the patients' test data better than did the other three algorithms. The authors suggested that further improvements could involve integrating the AKC-BCO with other evolutionary algorithms to increase its performance.

An in-depth review of all the common and major nature-inspired metaheuristics algorithms that have been used to solve automatic data clustering problems over the past few decades was reported in [10]. The study further reviewed and presented some of the important components, such as encoding schemes, validity indexes and proximity measures, datasets and applications involved in the formulation of these major metaheuristics algorithms. A total of sixty-five potential metaheuristics approaches for automatic clustering were identified, giving an up-to-date overview on single-solution, single-objective and multi-objective metaheuristic techniques. From the authors' observation, automatic clustering algorithms based on single-objectives are adequate for clustering datasets of linear separability without overlapping and datasets of near separability with overlapping, while few of the automatic clustering algorithms were used in an attempt to cluster datasets of non-linear separability without overlapping.

For the single-solution metaheuristics, the simulated annealing (SA) and K-mean search techniques were considered, in which the SA outperformed K-means algorithm. The SA was able to jump between clusters of different dimensions as well as automatically provide the appropriate number of clusters with better index values. Similarly, a revised tabu search (TS) algorithm was considered in the extensive review study. The TS is an enhanced search method that enables it to escape from being entrapped into local minima by using a tabu memory. Logically, single-solution metaheuristics are exploitation-oriented, and for this reason, they are seldom applied to solve automatic clustering problems. Consequently, a single-solution approach would, undoubtedly, require additional mechanism to explore the whole search space for the optimal solution, although with the risk of solutions being entrapped within the local minima. This brought about the concept of a single-objective solution, which offers a better and more accurate solution.

Single-objective metaheuristics, otherwise known as population-based metaheuristics, are processes that iteratively attempt to improve a population of solutions, whereby the fittest individual emerges as the best candidate solution, being picked after a thorough search or exploration has been carried out on all the search space. Unlike single-solution metaheuristics, which are exploitation based, single-objective are exploration-based. Exploration gives a better diversification of the search than does exploitation [10, 47]. The population-based algorithms are very able to find multiple candidates simultaneously for clustering solutions. These algorithms have over time shown their ability to solve clustering problems. Forty-five of these algorithms that can been used to solve automatic data clustering problems were studied in [10]. It is important to note that only three (3) of the revised genetic algorithm-based algorithms were tested with complete evaluation on both synthetic datasets and real-life datasets. Furthermore, twenty-nine (29) of the revised algorithms were evaluated with synthetic datasets, with most of them been linear separable. In an attempt to solve non-linear separable clusters, six (6) algorithms were evaluated, but they obtained inadequate clustering solutions, because the objective function considered not only the clusters compactness but also the separation of clusters represented by their respective single prototypes. To overcome the problem posed by this single-objective algorithms, the multi-objective and hybrid algorithm approaches were suggested as clustering solution methods. These approaches could integrate other objective functions to measure both cluster compactness and the connectedness of clusters, so were suggested as more robust and efficient alternatives.

Multi-objective algorithms endeavor to optimize multiple objective functions because the optimal solution of multi-objective optimization problem is not a single-solution as in the case of the single-solution metaheuristics, but rather it is a set of Pareto-optima solutions (i.e., solutions where it is impossible to improve a given objective without disintegrating at least one another). In turn, this set of solutions presents solutions that compromise between two conflicting objectives [10, 43] in which clustering tasks are formulated as a multi-objective problem whereby multiple objective criteria are set and evaluated simultaneously. The most important distinction of the multi-objective metaheuristics approach is that the algorithms partitions the datasets into appropriate number of clusters automatically along with the correct grouping. The study presented in [10] reviewed eighteen (18) multi-objective algorithms that had been used to solve automatic clustering problems. Only two of these algorithms underwent a complete evaluation test with synthetic datasets, real-world datasets and application field. Fourteen of these algorithms used synthetic datasets to evaluate performances, in particular they were not considered in linear separable clusters. Only four of the revised algorithms considered non-linear separable clusters and they obtained better results than did single-objective methods.

Out of the sixty-five revised algorithms reviewed in [10], single-solution metaheuristics algorithms amounted to 3% solvability of automatic clustering problem, single-objective metaheuristics turned in 69% solvability, which proved to be the most preferred approach of all, while there was a 28% tendency for the multi-objective metaheuristic algorithms to solve automatic clustering problems. The authors suggested further that distinct datasets in different scenarios should be used to evaluate the clustering performance of automatic clustering algorithms. The authors identified three (3) main cases, as discussed earlier, to simulate their datasets (linear separability without overlapping, near separability with overlapping, and non-linear separability without overlapping). It is however important to note that the success of any nature-inspired metaheuristic algorithm depends strongly on its design.

According to the study presented by José-García and Gómez-Flores [10], no extensive and comprehensive research has been conducted to demonstrate the superiority of a specific automatic clustering algorithm or nature-inspired metaheuristics algorithm. Their further suggestions included the recommendation that several comparative studies should be used to develop at least a common benchmark data or application field. Conclusively, they suggested that more deep learning and research should still be done on the best way or approach to solving automatic data clustering problems.

2.1 Summary of related studies

Table 1 summarizes relevant details about some of the existing automatic clustering algorithms. The clustering techniques are based on nature-inspired metaheuristic algorithms, with corresponding columns for the author name, application area, dataset used as the test problem and the type of cluster validity index used.

2.2 Representative algorithm clustering networks

A total of 1649 publications indexed in Scopus databases, which had been published from the inception of the five selected algorithms to the year 2020, were collected and analysed to show the relevance of each algorithm to the

Table 1 A summary of some existing automatic data clustering approaches

Authors	Application area	Datasets	Cluster validity index
Omran et al. [84]	Cluster analysis and image segmentation	Real life datasets	Dunn index (DI), Turi index, and S_Dbw index
Masoud et al. [85]	Cluster analysis and combinatorial optimization problem	Artificial and real life datasets	Variance Ratio, Criterion (VRC) and DB index
Ling et al. [86]	Cluster analysis	Artificial and real life datasets	Rand index (RI)
Kuo and Zulvia [87]	Cluster analysis	Real life datasets	VI index
Satyasai and Ganapati [88]	Cluster analysis and 3D human models	Artificial and real life datasets	–
Das et al. [15]	Cluster analysis	Artificial and real life datasets	CS index
Kao and Chen [89]	Cluster analysis and cell formation	Artificial and real life datasets	–
Abubaker et al. [90]	Cluster analysis	Artificial and real life datasets	DB index, Symmetry (Symm) index, and Conn index
Das et al. [13]	Cluster analysis and image segmentation	Real life datasets	DB and CS indices
Lee and Chen [91]	Cluster analysis	Real life datasets	I index
Saha et al. [92]	Cluster analysis	Artificial and real life datasets	Xie-Beni (XB) index
Maulik and Saha [93]	Image segmentation	Real life datasets	XB index
Suresh et al. [28]	Cluster analysis	Artificial and real life datasets	XB index, FCM index, Rand index (RI), and Silhouette index (SI)
Kundu et al. [46]	Cluster analysis	Artificial and real life datasets	XB index and FCM index
Zhong et al. [94]	Cluster analysis and remote sensing	Real life datasets	XB and J_m indices
Liu et al. [95]	Cluster analysis	Artificial and real life datasets	DB index
He and Tan [11]	Cluster analysis	Artificial and real life datasets	Calinski–Harabasz (CH) index and RI
Rahman and Islam [96]	Cluster analysis	Real life datasets	Adjusted rand index (ARI) and XB index
Ozturk et al. [97]	Cluster analysis	Real life datasets	VI index and correct classification percentage (CCP)
Kuo et al. [18]	Cluster analysis and medicine (Prostate Cancer)	Artificial and real life datasets	CS_{kernel} and VI index
Kuo and Zulvia [98]	Customer segmentation	Real life datasets	VI index and ARI
Chowdhury et al. [20]	Cluster analysis	Artificial and real life datasets	Sym-K index
Murthy et al. [99]	Cluster analysis	Real life datasets	CS index
Das et al. [21]	Cluster analysis	Artificial and real life datasets	CS index
Peng et al. [100]	Cluster analysis and membrane computing	Real life datasets	CS index
Kumar et al. [101]	Image segmentation	Real life datasets	Inter-intra cluster ratio and fitness function evaluation
Liu et al. [41]	Image segmentation	Real life datasets	PMB index
Kumar et al. [16]	Cluster analysis and image segmentation	Real life datasets	Point symmetry-based cluster validity index
Kapoor et al. [102]	Cluster analysis and image segmentation	Real life datasets	Inter-intra cluster ratio and DB index
Anari et al. [103]	Cluster analysis and image segmentation	Real life datasets	S_Dbw index
Zhou et al. [17]	Cluster analysis	Real life and artificial datasets	Fitness function evaluation
Pacheco et al. [104]	Cluster analysis	Real life datasets	SI
Elaziz et al. [105]	Cluster analysis	Real life and artificial datasets	Dunn index, SI, DB index and Calinski–Harabasz (CH) index
Chowdhury and Das [37]	Pattern recognition	Real life and artificial datasets	Huang's accuracy measure
Sheng et al. [106]	Miscellaneous	Real life and artificial datasets	DB, CH, I-index
Zhou et al. [107]	GPS data based trajectory	Real life: Taxi GPS Datasets	DB index
Agbaje et al. [108]	Cluster analysis	Real life datasets	DB and CS indices

problem at hand. From this analysis, GA has 887, PSO has 524, DE has 180, FA has 49, and DE has 9 published documents. Based on these documents, a full counting co-occurrence network analysis was constructed. The co-occurrence network visualization was constructed and illustrated for each algorithm to show how relevant and impactful the selected representative algorithms are to the field of data clustering and clustering analysis as whole.

Figures 2, 3, 4, 5 and 6 present visualizations of the co-occurrence of the GA, PSO, DE, FA, and IWO algorithms in relation to their applications to different clustering related problems. Problem specific domains are represented with circles and labels. The size of the label and the circle on the graph represent the weight or publication count for each algorithm to the corresponding problem type. The network lines represent the links between the candidate algorithm, the version of clustering algorithm and the nature of the problem being addressed. Colours in the graph are used to represent the clusters of algorithms to which a specific problem belongs. Further, each of the network graphs clearly depicts a densely connected network of clusters and sub-clusters, which are linked, either directly or indirectly, to the selected candidate algorithms, showing that the algorithms have wider applications to the field of clustering analysis or data clustering problems. However, for the FA algorithm illustrated in Fig. 5, which is a more recent algorithm than the GA, PSO and DE, the network graphs still reveals a moderate relevance impact

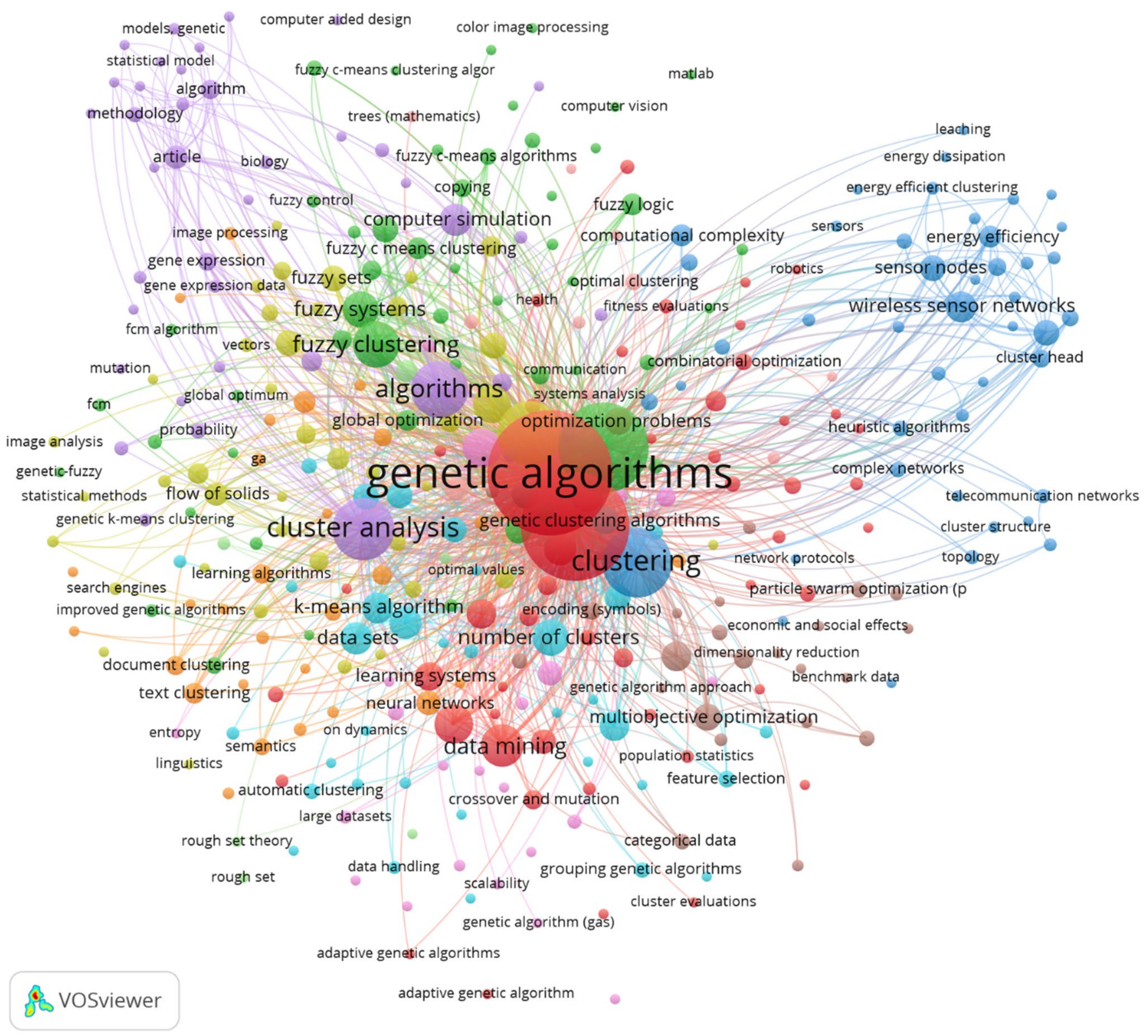


Fig. 2 Visualization of GA application to clustering related studies

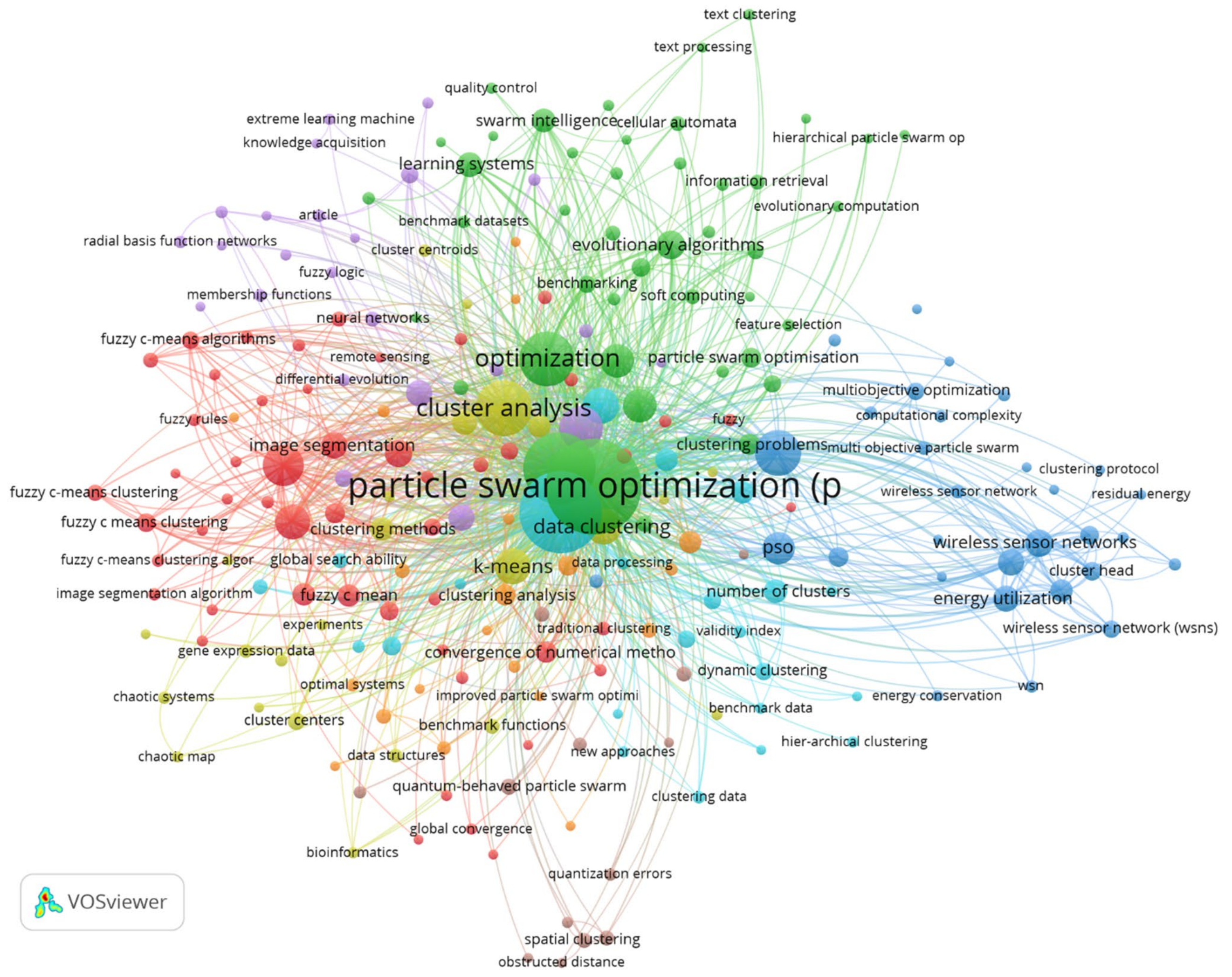


Fig. 3 Visualization of PSO application to clustering related studies

of the FA application to clustering problem. The IWO algorithm, as shown in Fig. 6, has sparse links and fewer cluster connections, which clearly shows that the algorithm has received limited applications for clustering problems. Despite this observation, IWO was selected as part of the representative algorithm on the ground of exploiting and exploring some of its latent performance potential, as highlighted in existing studies [48–50].

2.3 Representative algorithms strengths and weakness

The five algorithms whose performances are being investigated in this paper have their general strengths and weaknesses. For some of them, the optimization techniques require much iterative power and they even perform very poorly without adequate parameter fine-tuning. However, these algorithms have potentials or merits that have been widely explored and applied to solve several complex optimization and real-world problems. More so, the strengths of these algorithms are better tapped when they are combined with other traditional techniques or even similar global metaheuristic algorithms. Some of the sought after merits for these algorithms include their innate mechanisms for information sharing, ability to escape entrapment from local optima and great potential for exploitation and exploration capability. Further, these algorithms are very simple to understand, design and implement. In Table 2, some of the specific strengths and weaknesses of the five algorithms evaluated in this paper are outlined. Similarly, additional information on the advantages and disadvantages of these algorithms can be obtained from [51].

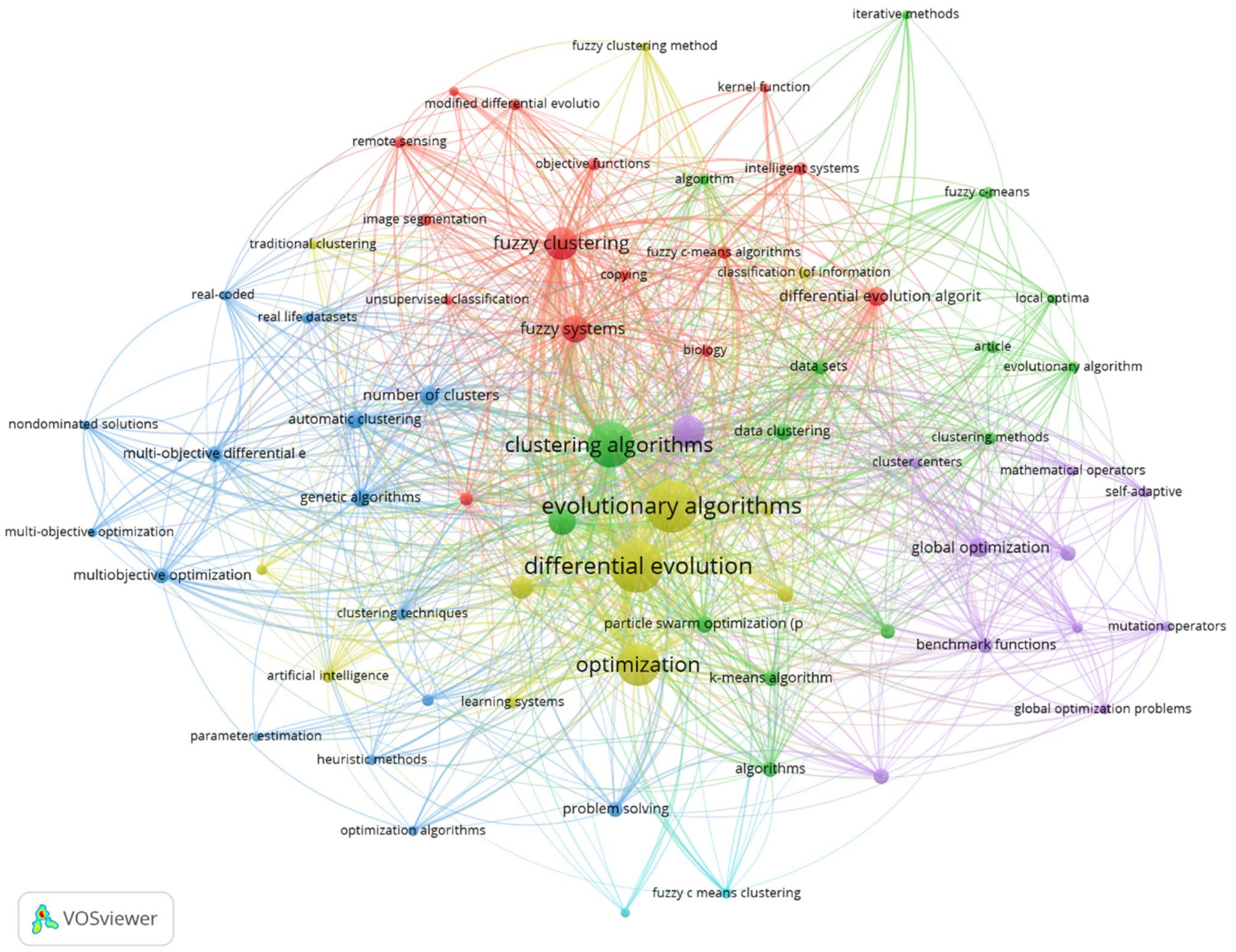


Fig. 4 Visualization of DE application to clustering related studies

3 Clustering problem preliminaries

Data clustering is considered to be an unsupervised learning process, whereby similar objects are clustered together in the same group, while dissimilar objects are clustered together in different groups. It is however, important to mention here that although there are instances where information regarding the number of clusters is predefined, in most cases, especially with large or high dimensional data objects, the number of clusters is not usually predetermined. More so, the task of clustering, or the clustering analysis process, is based on the similarity and dissimilarity measures between two data points. Therefore, in principle, the main goal of clustering analysis is to obtain high intra-cluster similarity and low inter-cluster similarity. An example illustrating the clustering analysis process is represented in Fig. 7.

The data clustering problem can be formulated as follows: Consider a dataset X, which contains n data points $x_1, x_2, \dots, x_n$, with each of the date items having d-dimensional attributes, features, variables, components. Formally, $X = \{x_1, x_2, \dots, x_n\}$ is a set of n data points, each having d real-valued features. In other words, $x_i \in \mathbb{R}^d, \forall i = 1, 2, \dots n$, $x_i = \{x_{i1}, x_{i2}, \dots, x_{ij}, \dots x_{id}\}$, where x_{ij} denote all the features of x_i. The dataset X can also be expressed in matrix form as follows:

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_i \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,d} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,d} \\ \vdots & \vdots & \vdots & \vdots \\ x_{i,1} & x_{i,2} & x_{i,j} & x_{i,d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,d} \end{bmatrix}. \tag{1}$$

The data points $X = \{x_1, x_2, \dots, x_n\}$, can be classified into a certain number of clusters, $C = \{c_1, c_2, \dots, c_k\}$, for some $k > 1$, such that the following conditions hold [1]:

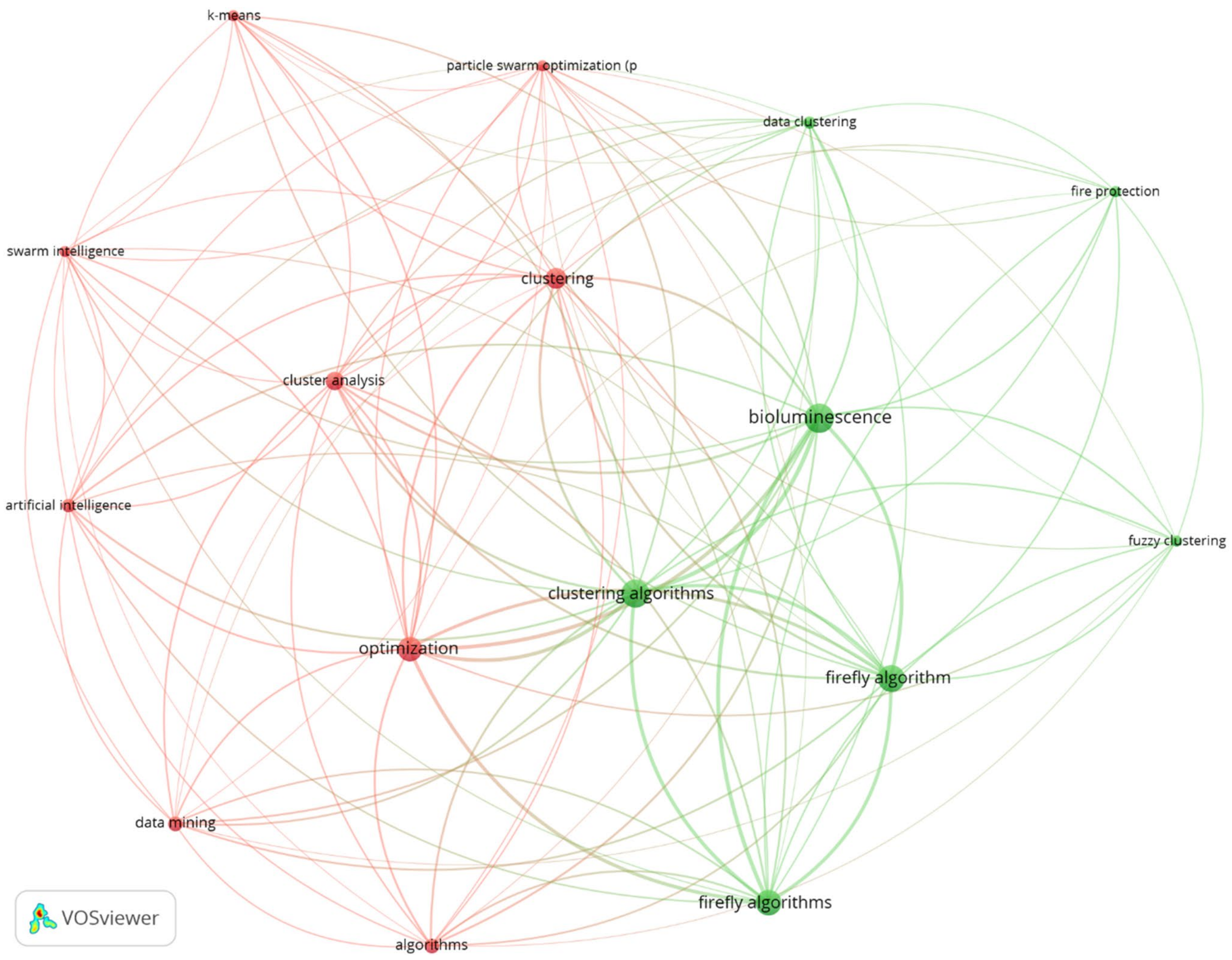


Fig. 5 Visualization of FA application to clustering related studies

1. $\cup_{i=1}^{k} c_i = X$
2. $c_i \cap c_j = \emptyset \quad \forall i, j \in \{1, 2, \dots k\}, i \neq j$
3. $c_i \neq \emptyset \quad \forall i \in \{1, 2, \dots k\}, i \neq j$.

Furthermore, assuming that the number of cluster k is predetermined, then each cluster must have one center denoted by $g_i (i = 1, 2, \dots, k)$. Then it is also assumed that $G = \{g_1, g_2, \dots, g_k\}$ represents the collections of centroids of $X = \{x_1, x_2, \dots, x_n\}$. In this case the centroid g_i is defined as follows:

$$g_i = \frac{1}{|c_i|} \sum_{x_j \in c_i} x_j \tag{2}$$

where $|c_i|$ represents the number of points in a classified collections of c_i.

In this paper, clustering is considered to be an optimization problem and similarly its belongs to the family of NP-hard optimization problems [2], for the reason that with a complex clustering task, the number of clusters are not determined prior to the clustering analysis. In other words, the number of clusters is not defined as an input variable. Therefore, the process of clustering $C = \{c_1, c_2, \dots, c_k\}$ of X data items is such that the function $f(C, D)$ is optimized over all possible clusterings of X, where the function f denotes the global validity index used to capture the notion of high quality clustering solutions and D is the distance metric.

3.1 Clustering validity index

There are several quantitative evaluation methods that are akin to statistical mathematical functions employed to analyze the quality of results obtained by clustering algorithms. Among these, to mention but a few, we find

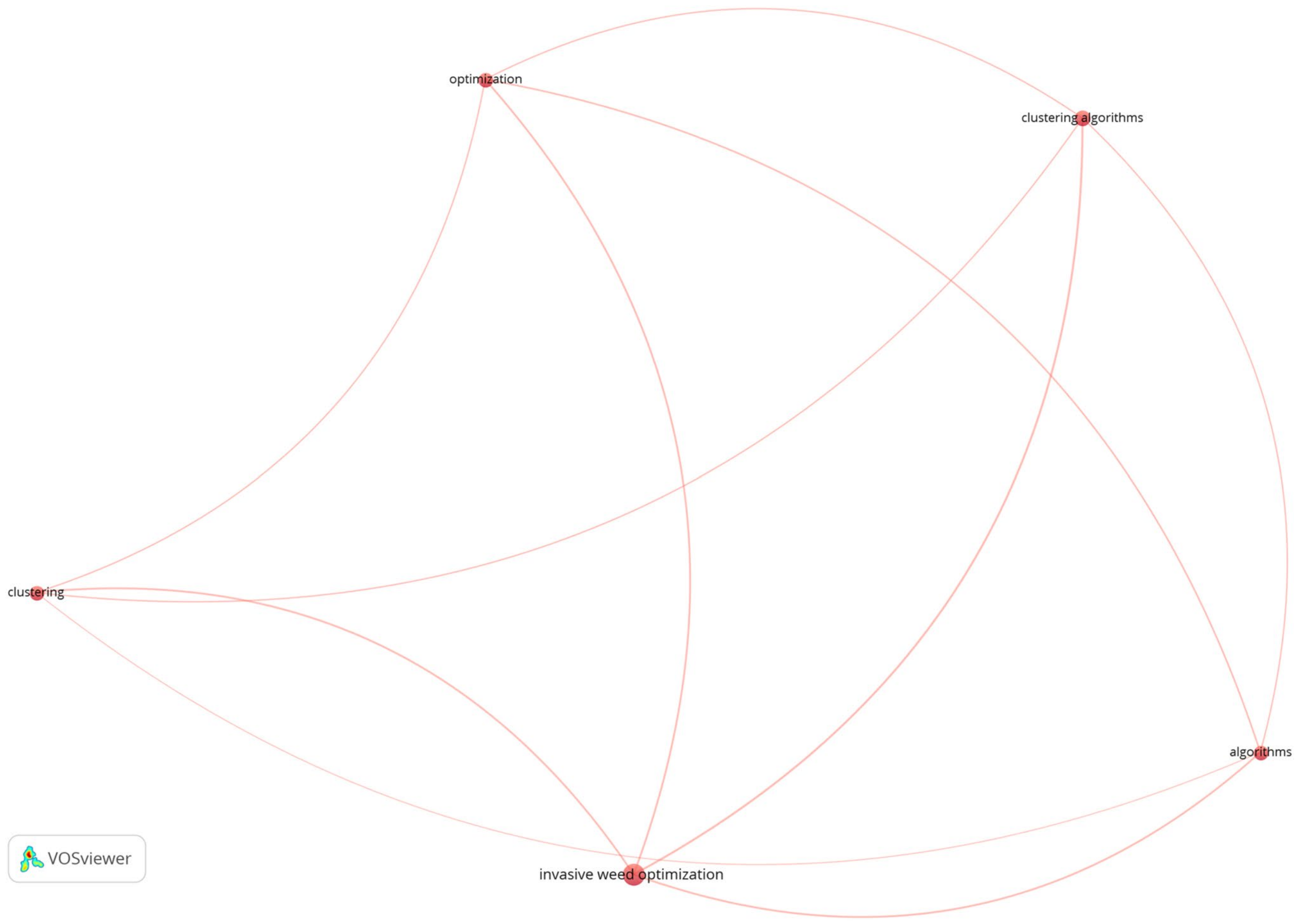


Fig. 6 Visualization of IWO application to clustering related studies

the Davies–Bouldin (DB) validity index [52], Dunn validity index [53], Calinsky-Harabasz validity index [54] and Pakhira Bandyopadhyay Maulik validity index [55]. According to Das [56], a cluster validity index serves two main purposes. Firstly, it helps to determine the actual number of clusters in a dataset and secondly it is also used to find the corresponding best cluster portioning. More so, in terms of clustering, the validity index is principally concerned with clustering cohesion, or compactness and separation, as explained subsequently. For the current study, focus is given only to the DB clustering validity index, which is used as the proposed representative algorithms' objective function, upon which the quality of each individual algorithm clustering solution is evaluated and compared.

The Davies–Bouldin (DB) validity index is used to analyze the quality of the clustering solution across different dimensionality of benchmark dataset used in this study. The DB index treats clustering as a minimization case of optimization, because its attempts to minimize the average distance between each chosen cluster and the one most similar to it. However, in this research, the choice of selecting the DB validity index is based on computational cost efficiency, capability of finding feasible high quality results for data of arbitrary dimensionality, and limited number of parameter specifications with little or no user interference. In general, the DB validity is capable of efficiently determine the number of clusters with best partitioning over a large number of data points with high level of intra-cluster cohesion (data points belonging to the same cluster should be very close or similar) and of inter-cluster separation (dissimilarity between clusters should be well separated) [1, 3]. In this regard, the validity index is calculated as a function $f(C, D)$, which returns a real number that expresses the quality of the clustering solutions. The DB validity index of a clustering C is calculated as follows:

$$f_{DB}(C, D) = \frac{1}{K}\sum_{i=1}^{K} D_i \tag{3}$$

Table 2 Strengths and limitations of the five representative metaheuristic algorithms

Algorithm	Strength of algorithm	Limitation of algorithm
Differential evolution	DE has excellent intensification and diversification features DE has the capacity to deal with cost functions that are non-differentiable, multimodal and nonlinear DE can handle cost functions with high computational complexity DE is easy to use; it requires only few parameters compared to other similar metaheuristic algorithms	DE convergence is not stable DE easily falls into regional optimum DE requires parameter tuning In DE, using the same parameter setting may not produce the global best solution
Genetic algorithm	GA is easy to implement GA has the ability to handle random types of objectives and constraints GA can be used independently to solve a given problem. It does not depend on other algorithms or heuristics for it to function well GA can be used to handle problems whose constraints and objective functions are nonlinear or discontinuous GA uses simple operators and can be used to solve problems that have high computational complexity, such as the travelling salesman problem and data clustering problem as well	GA has high likelihood of getting trapped in the local maxima GA does not have a standard method for defining a good fitness function. The best solutions majorly depend on the problem defined fitness function and hence the fitness function must be very accurate In GA, premature convergence seldom occurs, thus losing the population diversity GA does not have a standard termination criterion; neither does it have a standard method for adjusting its parameters. Therefore, parameter fine tuning is necessary for optimal solution realization GA can be time consuming, especially for problems with large number of variables
Particle swarm optimization	PSO is very simple to implement PSO is very robustness and has excellent ability to control parameters PSO is computationally efficient when compared with mathematical algorithm and other heuristic optimization techniques PSO is useful in scientific research and in engineering and therefore has wide coverage in its application to solving real world problems	In PSO, solutions are more likely to converge prematurely and consequently loses the population diversity PSO suffers from partial optimism PSO has the potential to easily fall into local optimum in high-dimensional space
Firefly algorithm	FA has the ability to automatically divide the population into different groups, hence it is good for diversification FA has good information-sharing mechanism which can promote the algorithm to converge faster under certain conditions FA has a lower probability of entrapment into local optima	Similar to some metaheuristic algorithms, FA performance depends on adequate parameter tuning Diversification in FA can lead to reduced speed and reduced convergence rate
Invasive Weed optimization	In IWO, all the candidates in the solution space are involved in the reproduction stage, whereas some algorithms (such as GA) does not allow individuals with less fitness value to reproduce IWO is very easy to implement	IWO is not very effective for problems with large search space. This is because, a larger number of seeds will be applied to each plant so that the search space can be totally examined. This can lead to increase in computation time In IWO, competitive exclusion may shrink search space and place most seeds in the same local area IWO has low search accuracy In IWO, determining the accurate value of standard deviation is often very difficult

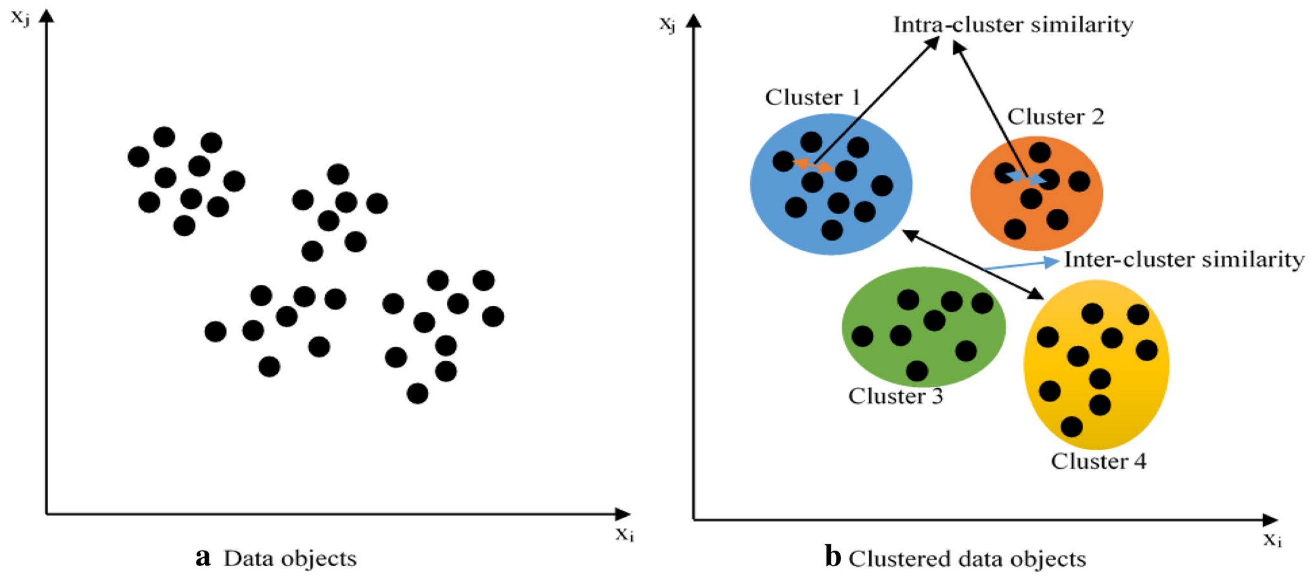


Fig. 7 Clustering example with intra-and-inter- clustering illustrations

$$D_i = max\left\{ \frac{s_i + s_j}{\beta_{ij}} | 1 \le i,j \le k, i \ne j \right\} \tag{4}$$

$$s_i = \left[\frac{1}{|c_i|} \sum_{x_i \in c_i} D(x_i, g_i)^p \right]^{\frac{1}{p}} \tag{5}$$

$$\beta_{ij} = D(g_i, g_j), \quad i \ne j \tag{6}$$

where s_i is the average distance of all data points in a given cluster c_i from its centroid g_i and β_{ij} represents the inter-cluster distance between two centroids g_i and g_j. The function $D(x_i, g_i)$ is the distance measure between data point $X_i = \{x_1, x_2, \dots x_n\}$ and its centroid g_i, while $p \in \{1, 2\}$ is an integer variable that is independently selected. For example, when $p = 1$, s_i is the average Euclidean distance between data points in a cluster and when $p = 2$, s_i is defined as the standard deviation of the distance of the data points in a cluster from their centroid. The smaller the value of the DB validity index obtained, the better the clustering method is able to separate the dataset [4].

The compact-separated (CS) validity index follows a similar conceptual rationale as described for DB validity index. However, it integrates and transforms the DB and Dunn validity indices with the main aim of tackling clusters of varied densities and features [57].

$$f_{CS}(C, D) = \frac{\sum_{i=1}^{K} \left(\frac{1}{N_i} \sum_{x_j \in C_i} \max_{x_l \in C_i} D(x_l, x_j) \right)}{\sum_{i=1}^{K} \left(\max_{j \in K, j \ne i} D(g_l, g_j) \right)}. \tag{7}$$

Here K is the number of clusters in C, while N_i denotes the number of data objects belonging to the cluster C_i. As with the DB index, small values of CS correspond to good clustering results. However, the CS index is only implemented in this paper with a view to comparing the proposed algorithms with existing literature results, and this appears in the later part of the study discussed in Sect. 4 below.

3.2 Similarity measure

In order to accurately evaluate and analyze the grouping patterns of similar objects in datasets, there is a need to use an appropriate distance measure, which is able to identify clusters irrespective of the volume of data items under consideration. The Euclidian distance metrics has gained wide acceptance and therefore, it is often used as a similarity measure for computing the similarity between two data points $\vec{x}_i$ and $\vec{x}_j$, each with n attributes or features. The aforementioned distance metric is calculated using the following expression:

$$D(\vec{x}_i,\vec{x}_j) = \sqrt{\sum_{n=1}^{d}(x_{in} - x_{jn})^2} = \left\|\vec{x}_i - \vec{x}_j\right\|. \tag{8}$$

However, in this study, a special case of the modified Minowski metric, here referred to as the cosine distance or vector dot product, is used as the similarity measure. It is given by [56] as

$$\langle \vec{x}_i, \vec{x}_j \rangle = \frac{\sum_{n=1}^{d} x_{i,n} \cdot x_{j,n}}{\left\|\vec{x}_i\right\| \left\|\vec{x}_j\right\|}. \tag{9}$$

The vector dot product determines the angular difference of the two data points expressed as vector points $\langle \vec{x}_i, \vec{x}_j \rangle$, instead of their magnitudes. The objective functions of clustering problems are usually non-linear and non-convex so some clustering algorithms may fall into a local optimum, for which the *k*-means algorithm is an example. The *K*-means algorithm minimizes the squared error function, which in this case is considered as the clustering algorithm objective function. It is given as follows.

$$f(X,G) = \sum_{i=1}^{n} min\left\{ \left\|x_i - g_k\right\|^2 | k = 1,2,\ldots,K \right\} \tag{10}$$

where g_k is the *k*th cluster center. However, since the study is focused on using a set of metaheuristic algorithms to solve the proposed automatic data clustering problem, the fitness of an individual is computed using the DB validity index given in Eq. (3) above. Accordingly, with respect to the current study, the DB-based fitness function, which is evaluated for the *i*th individual in the population, is defined as follows:

$$f(x_i) = \frac{1}{\psi + \varepsilon} \tag{11}$$

where ε is an arbitrarily small positive value and ψ denote any of the aforementioned validity indices, namely, DB and CS indices.

4 Study motivation of representative algorithms

This study was motivated by the design and performance stability of the selected representative algorithms, which is supported by both theoretical and practical discussions in the literature providing unambiguous clarity of proofs as to why these algorithms work well. Although there is a considerable amount of literature covering the selected representative algorithms, most of the studies are, however, concerned with the various application areas of the respective algorithms. Only a handful of publications deal with the theoretical or qualitative analyses of these algorithms. In general, there is a significant gap between theory and practice as far as the validity of the claims of superior performances for the individual algorithms is concerned. While, it is equally important to note that these algorithms have successful practical applications and proven track records in the literature, nevertheless their theoretical analysis lags far behind, as earlier mentioned. Therefore, the main motivation for selecting the representative algorithms is based on the availability of theoretical analysis, which support the design, performance and suitability of the candidate algorithms for any specific application areas.

The theoretical proofs of the selected algorithms, based on mathematical analysis, was investigated by Yang [58]. The convergence and stability concerning particles genetic algorithms was studied by Suzuki [59]. The PSO stability and convergence were analyzed by Clerc and Kennedy [60]. From algebraic and analytical perspectives, Clerc and Kennedy analyzed a particle's trajectory under the PSO as it moves in discrete and continuous time. Similarly, the convergence analysis of IWO using Markov chain was discussed by Zhang et al. [61]. Yang and He [62] provided an extensive theoretical support for why the FA works well. Yang and He explained in their study that the FA is a good combination of accelerated particle swarm optimization (APSO), simulated annealing (SA), harmony search (HS) and DE enhanced in a nonlinear system. Therefore, it has been theoretically proved that those individual algorithms actually work well and, furthermore, the reasons for their working in practice have been qualitatively justified; there are also several empirical analyses that show the algorithms to be very stable, so Yang and He [62] concluded that the FA algorithm should work equally well in practice. Nevertheless, several theoretical and application suitability analyses have also been carried out on the FA by Yang [63]. For the DE algorithm, Ghosh et al. [64] conducted a comprehensive study on the convergence of canonical DE algorithm over a class of continuous functions with a unique global optimum. The aim of their study was to substantiate the applicability of the DE algorithm to a class of continuous and real-valued objective functions that possesses a unique global optimum, with the probability of having multiple local optima. Therefore, because no theoretical analysis has been carried out on many of the existing algorithms to substantiates their performances and capabilities, it is considered worthy to focus on the current representative algorithms that works well in practice and for which there is reasonable theoretical support as to why they algorithms work.

Summarized descriptions of the algorithmic concepts for each of the five representative metaheuristic algorithms is presented in "Appendix 1", while their strengths and limitations were already itemized in Table 2 of this paper. The discussion covers background information, design concepts, and motivations of each algorithm implementation approach. However, interested readers are referred to the primary literature sources of information on each algorithm, see (GA [65], PSO [66], DE [67], FA [68], and IWO [69]).

It should equally be highlighted here that another motivation for the selection criterion for the five chosen algorithms was their all having already been studied deeply and applied to the solution of various clustering problems. Therefore, the current study is on track with the task of investigating and presenting a detailed comparative evaluation study on the five algorithms, for which the current effort is to further validate their initial claimed superior performances from the perspective of clustering analysis.

4.1 Solution representation and encoding

In this paper, the five representative algorithms namely, GA, DE, PSO, FA and IWO are applied to solve the automatic data clustering problem. Adopting similar clustering and solution representation mechanisms presented in [13, 20], a common search space matrix is adopted for all the algorithms. More so, this is due to the fact that the algorithms belong to the same class of population-based metaheuristic. The population matrix is initialized as dataset $X = \{x_1, x_2, \dots, x_n\}$ for $i = 1, 2, \dots, n$ individuals following the expression in Eq. (1) above. The population matrix in this case comprises of strings which are composed of real numbers that encode the centers of the cluster partitions. The individual in the population encodes the number of clusters k_i. The grouping of individuals in the population into similar class is estimated based on the lower bound denoted by k_{min} and upper bounds denoted by k_{max} of the number of groups in the population. The number of clusters for each individual is evaluated as shown in Eq. (12).

$$k_i = rand(0,1) \times \left(k_{min} + \left(k_{max} - k_{min}\right)\right) \tag{12}$$

where the function $rand()$ is a vector of uniformly distributed random numbers between 0 and 1. For example, in a d-dimensional space, the length of d-dimensional dataset is $d \times k_{max}$, while an individual or a particle consist of a vector of real numbers of dimension $k_{max} + k_{max} \times d$. The length of ith individual is given as $d \times k_i$. Note that the first k_{max} values are positive floating points numbers in [0,1], and these values are similarly used to determine the suitability of the corresponding clusters for data points classification. Afterwards, the remaining k_{max} values are set aside for k_{max} clusters centers, each d dimensional space [13]. Note that the initial population are generated randomly. For example, in the case of the PSO, a random number between [0, 1] is generated for each particle position in the first part. While for the initial centroid, a data point is picked randomly for each cluster centroid. In the subsequent section, the details of the hybrid algorithms implementation methodology are discussed. In Fig. 8, an example illustration is used to explain the solution representation and individual particle encoding scheme for the proposed representative algorithms. Consider a particle in a 3-dimensional vector space, which may be denoted as follow: $\langle 3.3\ 2.4\ 6.21\ 7.82\ 8.32\ 5.24\ 6.43\ 6.91\ 10.42\ 11.2\ 2.38\ 6.91 \rangle$. Therefore, this particle will account for a total of four cluster centers (g_1, g_2, g_3, g_4) that have been encoded for the aforementioned 3-dimensional dataset as shown in the figure below.

4.2 Hybrid algorithms for clustering problems

The reason for proposing the three hybrid automatic clustering algorithms namely, PSODE, FADE, and IWODE is to improve the quality of clustering solution of the standard PSO, FA, IWO, and DE algorithms. Though studies revealed that each of the four aforementioned global optimization techniques have their individual setbacks of being susceptible to premature convergence, which can lead to potential solutions being trapped in local optima. This drawback is even more pronounced in these algorithms when using them to solve problems with nonlinear and non-continuous search spaces, as it is case with the clustering problems. Therefore, to overcome this disadvantage, many researchers have resulted to implementing several hybrid metaheuristic algorithms for the purpose of improving the performance and problem-solving capabilities of the participating standard or classical optimization algorithms [70, 71].

The main idea of hybridizing PSO, FA and IWO with DE is to integrate the DE operators (mutation, crossover and selection) into the three participating algorithms, and thus increasing the diversity of the population and the ability to have the participating algorithms escape being trapped in the local minima. However, to avoid any potential cost associated with most hybrid algorithm implementations, such as increase in computational cost and weakening of the fast convergence ability of the individual algorithms, the DE operators is incorporated into the PSO, FA and IWO only at a specified interval of iterations. In this interval of iterations, the PSO, FA and IWO swarms serves as the population for the DE algorithm, and the DE is executed for a number of generations.

Specifically, the DE is chosen because it is a high profile global optimization algorithm that has consistently been used to solve diverse complex optimization problems, one of which is the data clustering problem. However, some

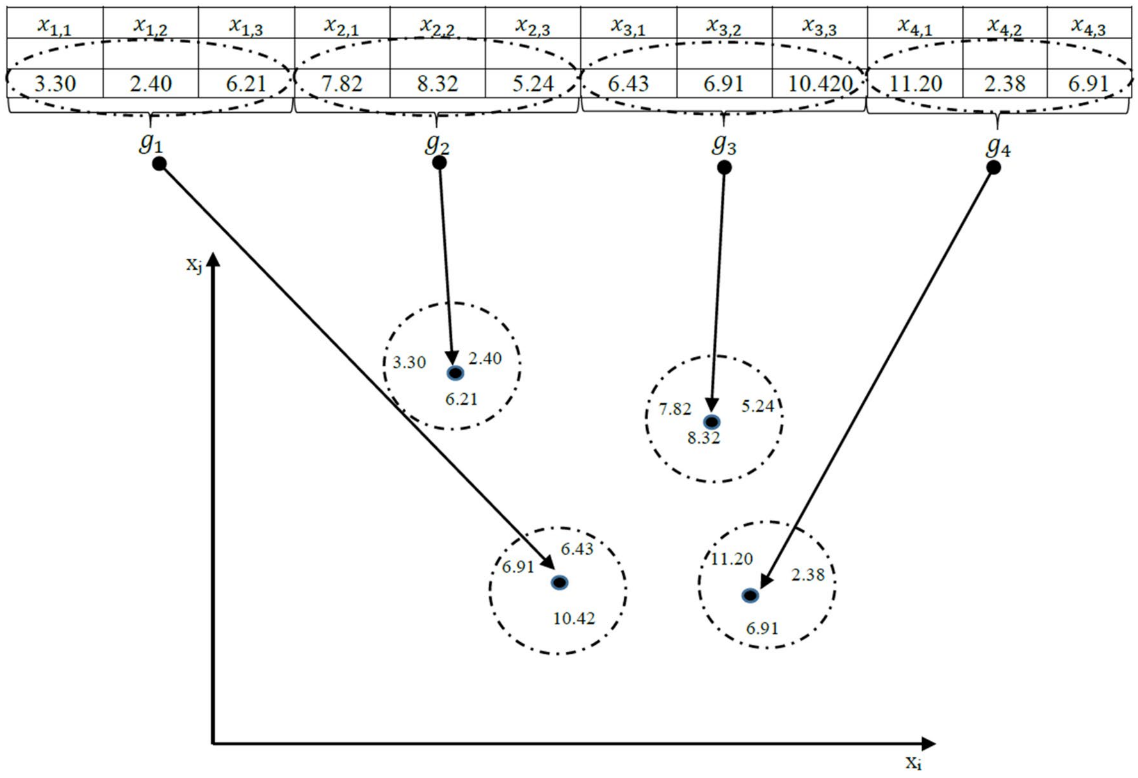


Fig. 8 Individual particle encoding scheme for the proposed metaheuristic automatic clustering methods

of the advantages of DE algorithm that were considered for the hybridization task aside the fact that it is simple to implement are as follow: DE is good at intensification and diversification of search space, it has the capacity to deal with cost functions that are non-differentiable, multimodal and nonlinear, it can handle cost functions with high computational complexity, it requires only few parameters and it has the ability to converge to the global minimum [51, 72]. More so, the objectives of the modified PSO, FA and IWO are to provide better initial centroids for the DE algorithm, so that the hybridized algorithms are able to achieve high compactness of cluster classification and separability of classes of data points into clusters.

The hybrid algorithm implementation methods comprise of two stages, the first stage engages either of the modified PSO, FA and IWO algorithm by randomly generating initial swarm, where the number of fireflies, particles or weeds are equal to the number of clusters and the swarm population are uniformly distributed across the dimension of the dataset, which in this case is the clustering problem search space. After the swarm initialization, the next task is the evaluation of the best swarm according to the fitness function determined by the DB validity index. Note that the best swarm position for example represents the data point that achieves the minimum distance to the swarm from its previous searches. Iteratively the fitness function of the best swarm position is calculated and the best cost updated subsequently. The second stage of the hybrid algorithm involves activating the three DE operators for all swarm, which in this case are the particles, fireflies and plants. For the predefined interval of iterations, the DE algorithm is executed; and the worst previous best position vectors are replaced with the best trial vectors obtained after executing three DE operators. Figure 9 depicts a summary flowcharts of the three proposed hybrid algorithms.

For the proposed hybrid methods, the mutation operator is applied to the previous best position swarm vectors P_i (for PSO) and X_i (for FA and IWO) using the following revised version of Eq. 16 (see "Appendix 1"):

Hybrid PSO-DE updating formula The proposed hybrid PSO-DE algorithm updating formula is given in Eq. (13).

$$V_{i,g+1} = P_{i,g} + F\left(P_{r1,g} + P_{r2,g}\right) - F(P_{r3,g} + P_{r4,g}) \tag{13}$$

The global and the previous best positions P_g and P_i values are updated according to the new fitness values computed using the DB index. More so, if the best position of all new particles is better than the current global best position, then P_g is replaced by the new solution. Similarly, the same updating approach is applied to the local best

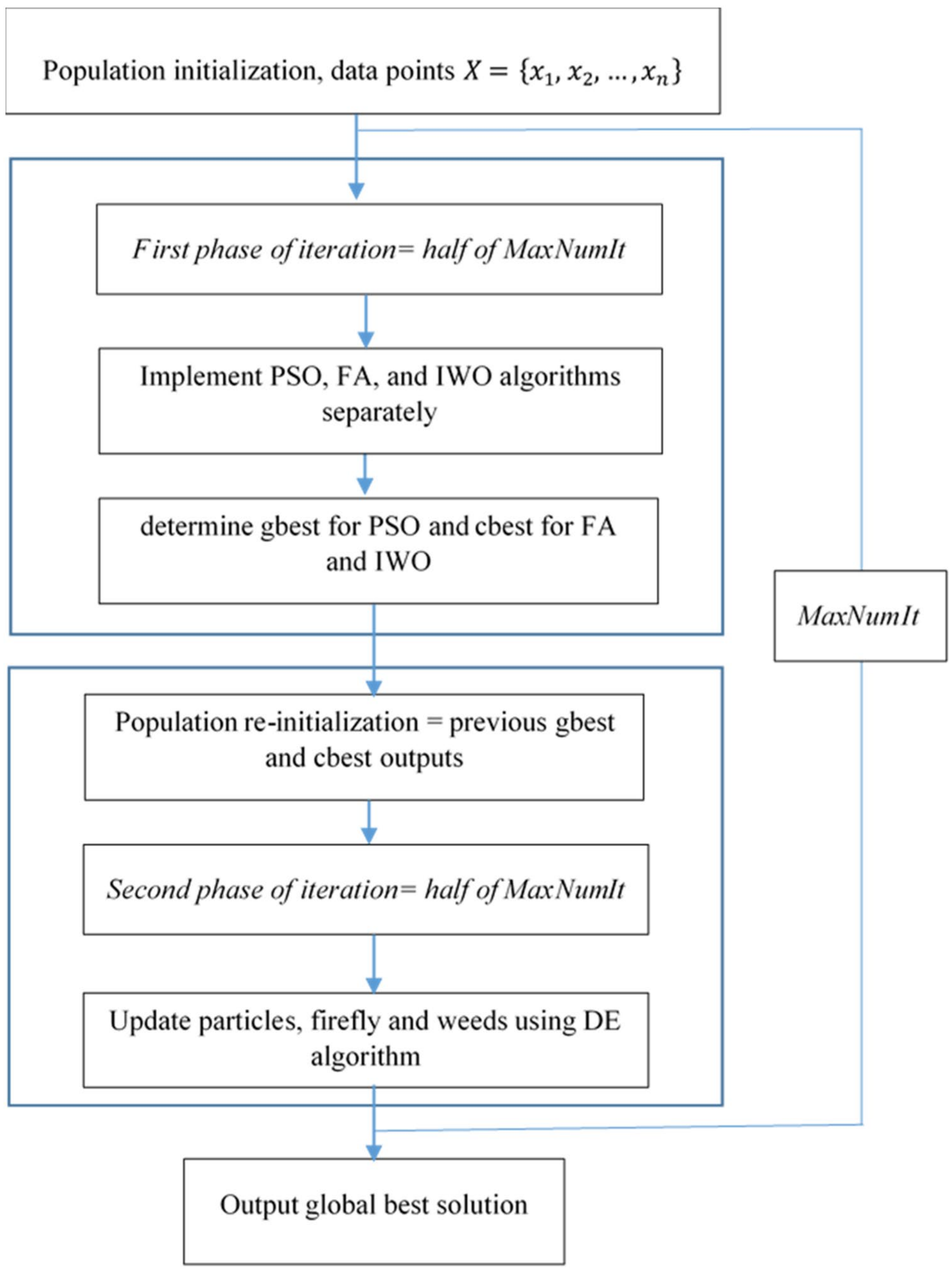

Population initialization, data points $X = \{x_1, x_2, \dots, x_n\}$
First phase of iteration= half of MaxNumIt
Implement PSO, FA, and IWO algorithms separately
determine gbest for PSO and cbest for FA and IWO
MaxNumIt
Population re-initialization = previous gbest and cbest outputs
Second phase of iteration= half of MaxNumIt
Update particles, firefly and weeds using DE algorithm
Output global best solution

Fig. 9 Summary flowchart of hybrid algorithms

of other particles in the population if the computed new fitness function value is better than the previous.

Hybrid FA-DE updating formula The proposed hybrid FA-DE algorithm updating equation is as represented in Eq. (14).

$$X_{i,\varphi+1} = X_{i,\varphi} + F\left(X_{r1,\varphi} - X_{r2,\varphi}\right) \tag{14}$$

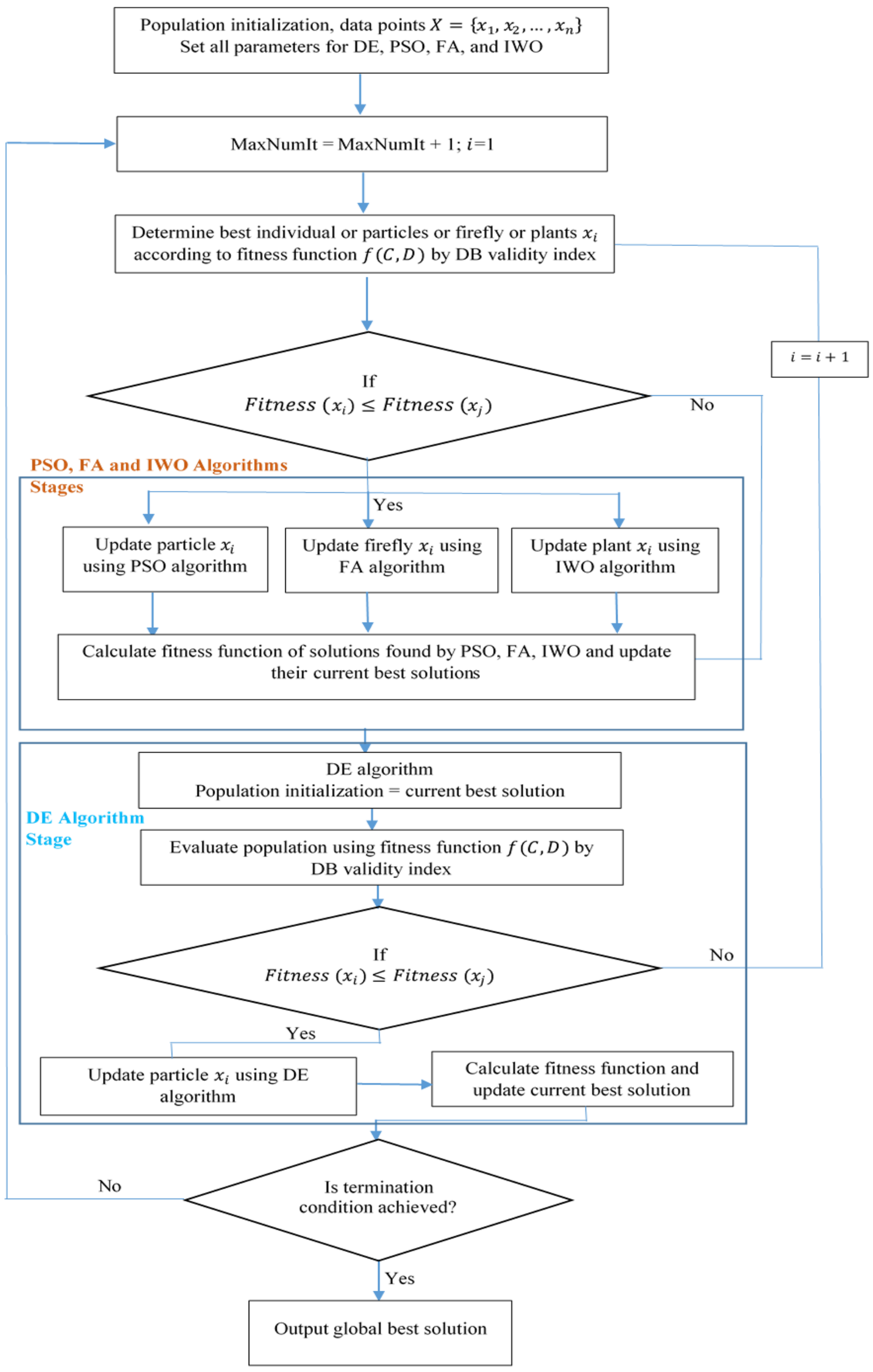
where X_i, X_{r1} and X_{r2} are randomly selected fireflies from population. The $X_{i,\varphi+1}$ is the new firefly which is created from the mutation operator at $\varphi + 1$iteration.

Hybrid IWO-DE updating formula In Eq. (15), the proposed hybrid FA-DE algorithm updating equation is represented as follows.

$$X_{i,\varphi+1} = X_{i,\varphi} + F\left(X_{best,\varphi} - X_{i,\varphi}\right) + \left(X_{r1,\varphi} + X_{r2,\varphi}\right) \tag{15}$$

where the parent vector X_{best} is the best individual of the current iteration. Note that Eq. (15) is only activated after the execution of Eqs. (26) and (27) presented in "Appendix 1".

Fig. 10 Flowchart of hybrid PSODE, FADE, and IWODE algorithms


Population initialization, data points $X = \{x_1, x_2, \ldots, x_n\}$
Set all parameters for DE, PSO, FA, and IWO
MaxNumIt = MaxNumIt + 1; i=1
Determine best individual or particles or firefly or plants x_i according to fitness function $f(C,D)$ by DB validity index
$i = i + 1$
If $Fitness(x_i) \leq Fitness(x_j)$
No
PSO, FA and IWO Algorithms Stages
Yes
Update particle x_i using PSO algorithm
Update firefly x_i using FA algorithm
Update plant x_i using IWO algorithm
Calculate fitness function of solutions found by PSO, FA, IWO and update their current best solutions
DE algorithm
Population initialization = current best solution
DE Algorithm Stage
Evaluate population using fitness function $f(C,D)$ by DB validity index
If $Fitness(x_i) \leq Fitness(x_j)$
No
Yes
Update particle x_i using DE algorithm
Calculate fitness function and update current best solution
No
Is termination condition achieved?
Yes
Output global best solution

The algorithm is terminated after the 200 set iteration criterion is reached, after which the best possible global clustering cost outcome is displayed. The detailed procedure of the proposed hybrid algorithms is illustrated in Fig. 10.

5 Experimental study

The experimental study carried out in this section evaluate the various performances of the five metaheuristic algorithms in terms of their respective success rates and computation time complexity. All the algorithms are executed in MATLAB 2018 and all the experiments are performed using an Itel Dual Core i7-7600U CPU @ 2.80 GHz and 15.8 GB RAM. The effectiveness of the algorithms is measured based on the following criteria:

1. Average best fitness value, which determines the quality of clustering solution for the representative algorithms.
2. Performance speed based on computational cost and convergence
3. Statistical significance difference over 20 replications

In this paper it is assumed that performance of the representative algorithms depends on the initial population size generation, considering the facts that the employed algorithms are population based algorithms. To allow for the replication of this experiment and its results, each algorithm is executed for 200 iterations and for 20 replications for all the forty-one dataset. The parameter settings for GA, DE, PSO, FA and IWO are depicted in Table 3 and the forty-one dataset used for the experiments are described and presented in Table 4 below.

5.1 Datasets

The datasets considered in this study is summarized in Table 4, consisting of three set of A-dataset, three birch datasets, one low-dimensional DIM dataset, seven high-dimensional DIM datasets, three image datasets, four generated S-datasets, nine UCI datasets, eight shape sets, two Mopsi locations datasets and one miscellaneous dataset. The *A* sets are two-dimensional synthetically generated datasets with increasing number of clusters, the *Birch* sets are two-dimensional synthetically generated datasets with predefined clusters of k = 100 and varying structure types, the high-dimensional *DIM* datasets have clusters that are well separated from one another with k = 16 predefined clusters, the low-dimensional *DIM* set is synthetically generated dataset of varying dimensional space with k = 9 Gaussian clusters of varying N number of data points or vectors. The points in the first set (*bridge*) of the *Image* sets is a 4 × 4 non-overlapping vectors or data points extracted from a gray-scale image, the second set is the h*ouse* (Housec5 and Housec8) dataset, which set consists of color values of the *RGB* image. The sets S1-S4 are two-dimensional synthetically generated datasets with varying complexity in terms of spatial data distributions with k = 15 predefined clusters.

In summary, the forty-one (41) dataset used to test the five representative algorithms and three hybrid algorithms have sizes ranging from 150 to 100,000 items or data points, number of features or dimension ranging from 2 to 1024, and number of classifications or clusters ranging from 2 to 256. It is important to note here that although the ground truth classification is provided (the ground truth refers to the classifications given in the dataset), it is not in any way utilized by the eight algorithms used for automatic clustering analysis. In other words, the eight algorithm does not use the classification or number of cluster information provided original with the dataset.

5.2 Experiment 1: Five representative algorithms on twenty datasets

The first set of experiments conducted covers the five representative algorithms, namely GA, DE, PSO, FA, and IWO. Table 5 summarizes the results obtained by each of the aforementioned algorithms for the problem at hand, over 20 datasets. The values reported in the table cover the best

Table 3 Parameter configurations for the five basic metaheuristic algorithms namely, GA, DE, PSO, FA, and IWO

GA		DE		PSO		FA		IWO	
Parameter	Value	Parameter	Value	Parameter	Value	Parameter	Value	Parameter	Value
Pop_size	25	Pop_size	25, 50, 100, 150	Pop_size	25, 50, 100, 150	Pop_size	25, 50, 100, 150	Pop_size	25, 50, 100, 150
P_c, P_m	0.8, 0.3	CR_{min}, CR_{max}	0.2, 1.0	W_1, W_2	1.0, 0.99	β_0	2	s	5
mr	0.02	F	0.8	τ_1, τ_2	1.5, 2.0	γ	1	Σ_1, Σ_2	0.5, 0.001
K_{min}	2	K_{min}	2	K_{min}	2	K_{min}	2	*e*	2
K_{max}	16	K_{max}	256	K_{max}	256	K_{max}	256	K_{max}	256
MaxGen.	200	MaxIt	200	MaxIt	200	MaxGen.	200	MaxGen.	200

Pop_size, population size; P_c, crossover probability; P_m, mutation probability; W_1, inertia weight; W_2, inertia weight damping ratio; τ_1, personal learning coefficient; τ_2, global learning coefficient; K_{min} and K_{max} denote respectively the maximum and minimum number of clusters that can be encoded in a single trial solution vector for GA, DE, PSO, FA and IWO; CR, crossover rate; F, scaling factor; mr, mutation rate; β_0, attractiveness; γ, light absorption coefficient; *e*, variance reduction exponent; Σ_1, initial value of standard deviation; Σ_2, final value of standard deviation; s, maximum number of seeds. Note that PSODE, FADE, and IWODE use the same parameter representation scheme as their respective standard algorithms

Table 4 Characteristics of the 41 clustering datasets of which 11 are artificial and 30 are real world (real life)

Datasets	References	Type of dataset	Number of data points (*N*)	Dimension of data (*D*)	Number of clusters (*k*)
A1	[40, 42, 107]	Synthetically generated	3000	2	20
A2	[40, 42, 107]	Synthetically generated	5250	2	35
A3	[40, 42, 107]	Synthetically generated	7500	2	50
Aggregation	[46, 107]	Shape sets	788	2	7
Birch1	[40, 43, 107]	Synthetically generated	100,000	2	100
Birch2	[40, 43, 107]	Synthetically generated	100,000	2	100
Birch3	[40, 43, 107]	Synthetically generated	100,000	2	100
Breast	[101, 107]	UCI dataset	699	9	2
Bridge	[103, 107]	Gray-scale image blocks	4096	16	256
Compound	[18, 107]	Shape sets	399	2	6
D31	[95, 107]	Shape sets	3100	2	31
Dim002	[40, 44, 107]	Synthetically generated	1351–10,126	2–15	9
Dim016	[40, 45, 107]	High-dimensional	1024	16	16
Dim032	[40, 45, 107]	High-dimensional	1024	32	16
Dim064	[40, 45, 107]	High-dimensional	1024	64	16
Dim128	[40, 45, 107]	High-dimensional	1024	128	16
Dim256	[40, 45, 107]	High-dimensional	1024	256	16
Dim512	[40, 45, 107]	High-dimensional	1024	512	16
Dim1024	[40, 45, 107]	High-dimensional	1024	1024	16
Flame	[37, 107]	Shape sets	240	2	2
Glass	[101, 107]	UCI dataset	214	9	7
Housec5	[103, 107]	RGB image	34,112	3	256
Housec8	[103, 107]	RGB image	34,112	3	256
Iris	[101, 107]	UCI dataset	150	4	3
Jain	[16, 107]	Shape sets	373	2	2
Leaves	[101, 107]	UCI dataset	1600	64	100
Letter	[101, 107]	UCI dataset	20,000	16	26
Joensuu	[106, 107]	Mopsi locations	6014	2	4
Finland	[106, 107]	Mopsi locations	13,467	2	4
Pathbased	[10, 107]	Shape sets	300	2	3
R15	[21, 107]	Shape sets	600	2	15
S1	[40, 41, 107]	Synthetically generated	5000	2	15
S2	[40, 41, 107]	Synthetically generated	5000	2	15
S3	[40, 41, 107]	Synthetically generated	5000	2	15
S4	[40, 41, 107]	Synthetically generated	5000	2	15
Spiral	[47, 107]	Shape sets	312	2	3
T4.8k	[20, 107]	Miscellaneous	8000	2	3
Thyroid	[101, 107]	UCI dataset	215	5	2
Wdbc	[101, 107]	UCI dataset	569	32	2
Wine	[101, 107]	UCI dataset	178	13	3
Yeast	[101, 107]	UCI dataset	1484	8	10

and worst solutions obtained, the averages over 20 simulations with standard deviations to indicate the range of values to which the representative algorithms converge. The results are presented to four decimal places. The first consideration is the resulting DB validity index obtained by the respective algorithms. As shown in Table 5, of the overall average values reported for the five algorithm reported, the FA algorithm had the smallest overall average DB validity index for the twenty selected datasets. Although, there are few cases where the PSO algorithm has a good clustering result, for example with the S1 dataset as shown by the validity index, these are not significantly better than those for the FA algorithm. The convergence curves for each algorithm are illustrated in Fig. 11, in which a smooth and rapidly declining curve indicates a superior performance. From the convergence curves, covering eight selected datasets

Table 5 Computational results for the five representative algorithms on twenty datasets

Dataset	GA				DE				PSO				FA				IWO			
	Best	Worst	Avg.	SD	Best	Worst	Avg.	SD	Best	Worst	Avg.	SD	Best	Worst	Avg.	SD	Best	Worst	Avg.	SD
A1	0.6113	0.7799	0.6907	0.0432	0.5919	0.6317	**0.6016**	0.0116	0.5901	0.7631	0.6662	0.0424	0.5901	0.6943	0.6089	0.0341	0.5939	0.7006	0.6308	0.0265
A2	0.713	0.8045	0.7549	0.0293	0.6786	0.7955	0.7081	0.0348	0.6751	0.7635	0.7134	0.0284	0.6776	0.7406	**0.6842**	0.0143	0.6821	0.8234	0.7483	0.0468
A3	0.6745	0.8537	0.7758	0.0418	0.7014	0.7776	0.7349	0.0184	0.6692	0.7893	0.7279	0.0287	0.6671	0.7561	**0.6844**	0.0241	0.7206	0.8705	0.7545	0.0339
Aggrega-tion	0.5409	0.7589	0.663	0.0673	0.6265	0.7329	0.6682	0.0298	0.5672	0.8409	0.6439	0.0643	0.5347	0.7199	**0.5816**	0.0386	0.5665	0.72	0.5977	0.0389
Birch1	0.726	0.8496	0.7976	0.0309	0.7227	0.7776	0.748	0.0171	0.7187	0.7921	0.7528	0.0169	0.6872	0.75	**0.7091**	0.0203	0.7346	0.8223	0.7644	0.0211
Birch2	0.507	0.669	0.6197	0.0361	0.5074	0.5134	**0.5086**	0.0016	0.507	0.6618	0.5876	0.0358	0.507	0.5418	0.5081	0.0078	0.5088	0.5249	0.5168	0.0052
Birch3	0.7159	0.8616	0.7718	0.0431	0.7074	0.8333	0.7488	0.0297	0.6745	0.7924	0.7213	0.0306	0.6788	0.716	**0.6974**	0.0127	0.7226	0.8252	0.7568	0.0248
Com-pound	0.4932	0.4935	**0.4932**	0.0001	0.4942	0.5282	0.5053	0.0078	0.4932	0.7465	0.6659	0.0651	0.4932	0.537	0.4954	0.0098	0.4932	0.6129	0.5002	0.0269
Dim002	0.5276	0.7771	0.6772	0.0664	0.603	0.6954	0.6685	0.0263	0.376	0.7117	0.5823	0.0772	0.4787	0.6849	**0.5807**	0.054	0.6053	0.7589	0.6661	0.0403
Flame	0.7738	0.7791	0.7758	0.0022	0.6605	0.8005	0.7261	0.0318	0.6294	0.8124	0.7329	0.0406	0.597	0.6974	**0.6496**	0.0449	0.5982	0.7737	0.6715	0.0442
Jain	0.6491	0.6524	0.6515	0.0008	0.6472	0.6707	0.6554	0.0047	0.6112	0.7084	0.6534	0.0243	0.6372	0.6514	**0.6446**	0.0048	0.6347	0.6792	0.6541	0.0096
Joensuu	0.4993	0.6727	0.519	0.0393	0.5067	0.5375	0.5205	0.0089	0.5253	0.69	0.6069	0.0393	0.4876	0.5961	**0.5038**	0.0249	0.4957	0.5162	0.5019	0.0075
Finland	0.4396	0.4418	**0.4407**	0.0009	0.446	0.4817	0.4575	0.0082	0.4393	0.6601	0.5761	0.063	0.4393	0.5462	0.4446	0.0239	0.4552	0.5829	0.5036	0.0398
Pathbased	0.6694	0.6695	0.6695	0.0000	0.6569	0.7268	0.6829	0.0161	0.6509	0.747	0.6905	0.0289	0.6238	0.6892	**0.6490**	0.022	0.6358	0.7106	0.6644	0.0193
S1	0.6508	0.7474	0.7006	0.0293	0.7115	0.7729	0.7363	0.015	0.5597	0.7173	**0.6472**	0.047	0.6315	0.6991	0.6793	0.0255	0.6837	0.8306	0.7572	0.0315
S2	0.6786	0.8097	0.7417	0.0393	0.6741	0.7548	0.7256	0.0221	0.6233	0.753	0.6878	0.0318	0.6657	0.7391	**0.6814**	0.0229	0.7237	0.8268	0.7625	0.0296
S3	0.6904	0.8461	0.7682	0.0395	0.7126	0.7839	0.7265	0.0179	0.689	0.8026	0.7261	0.0282	0.6691	0.7157	**0.7080**	0.0117	0.7219	0.8666	0.7576	0.0333
S4	0.7326	0.8039	0.7663	0.024	0.7224	0.7894	0.7635	0.018	0.6964	0.8072	0.7455	0.0316	0.7103	0.769	**0.7254**	0.0158	0.7406	0.8119	0.7809	0.0191
Spiral	0.7308	0.8088	0.7895	0.0224	0.7429	0.8309	0.7707	0.0231	0.728	0.8022	0.7664	0.0235	0.727	0.7965	**0.7364**	0.0171	0.7291	0.8021	0.7518	0.0237
T4.8k	0.0227	0.0227	**0.0227**	0.0000	0.0227	0.0231	0.0228	0.0001	0.0227	68.2531	17.5004	22.6863	0.0227	0.0227	**0.0227**	0.0000	0.0302	0.3241	0.1119	0.0665
Overall average	0.6023	0.7051	0.6545	0.0278	0.6068	0.6729	0.6340	0.0172	0.5723	4.1307	1.5197	1.1717	0.5763	0.6532	**0.5998**	0.0215	0.6038	0.7192	0.6427	0.0294

All text in bold represents the best results obtained by the individual representative algorithms

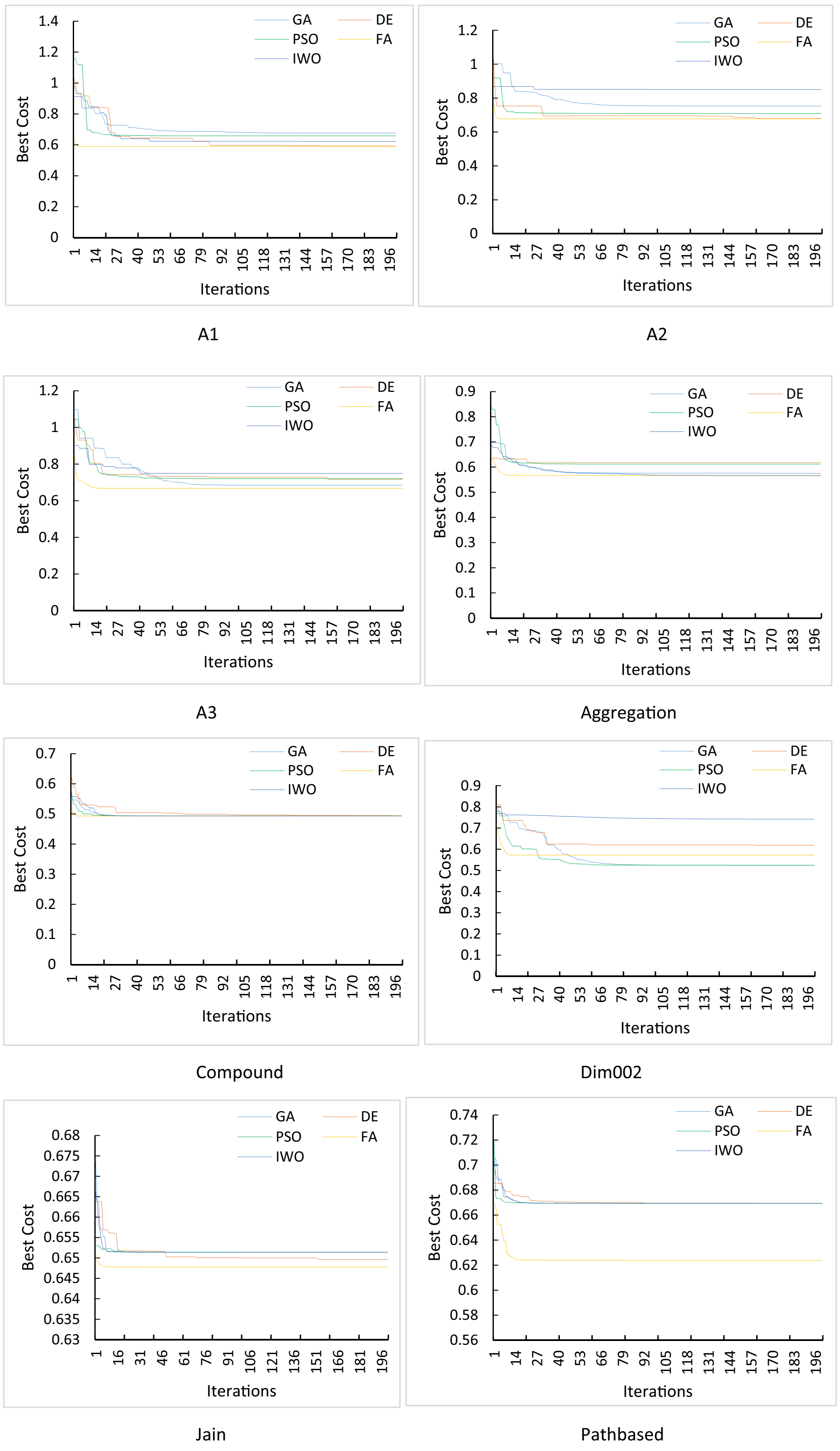


Fig. 11 Convergence curves of each algorithm on eight test datasets (based on objective function values)

(A1, A2, A3, Aggregation, Compound, Dim002, Jain and Pathbased), the FA algorithm clearly has the best performance: in all the graphs shown, the algorithm was able to produce smooth and rapidly declining curves. Furthermore, analysis of the standard deviation values obtained by the five representative algorithms in Table 5 shows that again the FA algorithm obtained consistently lower standard deviation (SD) values than the other algorithms, across all the twenty datasets. More so, the FA also obtained the best results, as shown in the "Best" column in Table 5 for 15 of the 20 databases, which shows that that, judging by the DB validity index values, the FA has the best fitness function computation with the least values. This indicates that the algorithm is superior to any of the other algorithms considered. The performance hierarchy in terms of which of the representative algorithms produced the smallest values for the computed DB validity index is as follows: FA being the best, is followed by the DE, then GA and IWO, with PSO being the worst performing of the five algorithms.

In Table 6, the results of the execution times obtained by each algorithm for 200 generations is shown. The average computation cost for each algorithm on the twenty datasets ranges from 18.4831 s for IWO to 53.5796 s for FA, with DE, PSO and GA falling in sequence between these values. The average execution cost therefore shows that although the FA was able to obtained high quality solution, it did so at the expense of computational time. Despite IWO having the best computational times, in terms of solution quality, it is inferior to the FA. The DE algorithm seems to have performed well, as the algorithm was able strike a good balance between better solutions and computational cost. From the experiment reported in Tables 5 and 6, it was observed that for the GA algorithm, as the problem dimension and features increased, it actually struggled to compute the fitness function, here the DB validity index. Therefore, in the remaining set of experiments the GA algorithm will be discarded, while hybridization attempts with the DE algorithm will be considered, while seeking a good balance between solution quality and execution time for the remaining set of representative algorithms. The option of the hybrid approach is to see if it is possible to improve the performance of the algorithms in terms of both solution quality and computational cost, specifically for the FA algorithm.

It is important to note that the high computational cost recorded for some of the algorithms can be attributed to the implementation of the exploration mechanism, which also happens to be one of the special features possessed by them, as was highlighted in Table 2. Even though the exploration mechanism provides the algorithms with unique performance capability and advantages, it does this at the expense of incurring additional computation cost, as can be seen in the case of the FA.

Table 6 Computational time cost for five representative metaheuristics on twenty datasets

	GA	DE	PSO	FA	IWO
A1	25.1799	19.9490	21.9353	23.1691	14.4312
A2	30.0072	20.6826	21.3371	27.7366	15.4780
A3	33.5669	22.1976	21.7788	33.0137	16.3230
Aggregation	20.9999	44.4496	26.0378	21.0443	14.3309
Birch1	119.1541	110.3298	79.6950	242.8657	43.1278
Birch2	64.4762	47.1180	58.7877	171.2215	32.4669
Birch3	84.2933	64.1067	96.7476	229.0219	36.1856
Compound	20.4090	19.0033	19.6271	18.8446	17.6630
Dim002	18.3266	18.5908	19.4369	19.3585	13.9877
Flame	20.3483	17.7655	23.2215	19.4257	13.3214
Jain	28.5519	16.5459	18.6040	18.9921	13.6465
Joensuu	18.0393	18.9126	24.0024	28.9638	14.0140
Finland	31.5844	20.8059	25.7042	36.6961	15.0975
Pathbased	38.6054	17.5430	28.0911	19.0709	14.1964
S1	28.6861	26.6109	28.7531	29.0218	16.4213
S2	28.6019	19.8328	20.7292	30.5368	15.6235
S3	29.6679	19.6668	21.4548	26.0717	15.4535
S4	27.7202	21.0218	39.6490	28.3019	15.6406
Spiral	35.3300	17.6803	30.7869	19.2370	17.4290
T4.8k	40.3651	21.2385	43.0841	28.9985	14.8247
Overall average	37.1957	29.2026	33.4732	53.5796	18.4831

5.3 Experiment 2: Representative and hybrid algorithms on forty datasets

The second experiment demonstrates extensive performance evaluation of the remaining four representative algorithms, that is without the GA, on forty clustering datasets. Table 7 shows the computational results obtained by the DE, PSO, FA and IWO on the forty datasets, while Table 8 shows the results obtained by the for hybrid algorithms. The hybrid algorithms used DE with the other three algorithms, giving PSODE, FADE and IWODE. In Table 9, the computational costs or execution times for all the algorithms are presented. In Table 7, the DB validity index results obtained for the respective algorithms show that the FA still performed better than other algorithms, which is a clear indication that the FA is an effective population-based algorithm for solving automatic data clustering problems. Consistently, FA has maintained a more stable lead in its solution quality performance across all the test datasets, which also shows the superior stability of the algorithm in comparison with DE, PSO, and IWO. In other words, FA can be said to maintain a high level of stability, as shown in the convergence graphs in Fig. 5 and the DB validity index results in Table 7 for the forty dataset show FA having an overall average of 0.6749, against 0.8117 for DE, 0.9021 for IWO and 1.5818 for PSO. The results

Table 7 Computational results for four representative algorithms on forty-one datasets

Comparator	DE				PSO				FA				IWO			
	Best	Worst	Avg.	SD	Best	Worst	Avg.	SD	Best	Worst	Avg.	SD	Best	Worst	Avg.	SD
A1	0.5919	0.6317	**0.6016**	0.0116	0.5901	0.7631	0.6662	0.0424	0.5901	0.6943	0.6089	0.0341	0.5939	0.7006	0.6308	0.0265
A2	0.6786	0.7955	0.7081	0.0348	0.6751	0.7635	0.7134	0.0284	0.6776	0.7406	**0.6842**	0.0143	0.6821	0.8234	0.7483	0.0468
A3	0.7014	0.7776	0.7349	0.0184	0.6692	0.7893	0.7279	0.0287	0.6671	0.7561	**0.6844**	0.0241	0.7206	0.8705	0.7545	0.0339
Aggregation	0.6265	0.7329	0.6682	0.0298	0.5672	0.8409	0.6439	0.0643	0.5347	0.7199	**0.5816**	0.0386	0.5665	0.7200	0.5977	0.0389
Birch1	0.7227	0.7776	0.7480	0.0171	0.7187	0.7921	0.7528	0.0169	0.6872	0.7500	**0.7091**	0.0203	0.7346	0.8223	0.7644	0.0211
Birch2	0.5074	0.5134	**0.5086**	0.0016	0.5070	0.6618	0.5876	0.0358	0.5070	0.5418	0.5087	0.0078	0.5088	0.5249	0.5168	0.0052
Birch3	0.7074	0.8333	0.7488	0.0297	0.6745	0.7924	0.7213	0.0306	0.6788	0.7160	**0.6974**	0.0127	0.7226	0.8252	0.7568	0.0248
Breast	0.7068	1.0267	0.8091	0.0758	0.9654	1.1996	1.1137	0.0596	0.6519	1.0546	**0.7667**	0.1650	0.6723	1.0462	0.7414	0.0908
Bridge	0.7377	0.9477	0.8292	0.0516	0.6124	1.1725	0.9786	0.1331	0.6109	0.7829	**0.6199**	0.0384	0.9785	1.2942	1.1575	0.0967
Compound	0.4942	0.5282	0.5053	0.0078	0.4932	0.7465	0.6659	0.0651	0.4932	0.5370	**0.4954**	0.0098	0.4932	0.6129	0.5002	0.0269
D31	0.8330	0.9447	0.8757	0.0302	0.6767	0.8508	**0.7630**	0.0514	0.7095	0.8497	0.7808	0.0436	0.7465	0.8704	0.7896	0.0312
Dim002	0.6030	0.6954	0.6685	0.0263	0.3760	0.7117	0.5823	0.0772	0.4787	0.6849	**0.5807**	0.0540	0.6053	0.7589	0.6661	0.0403
Dim016	1.0631	1.1892	1.1314	0.0350	0.8551	1.1214	1.0170	0.0614	0.6351	1.0717	**0.9677**	0.1003	1.2845	1.5309	1.4296	0.0648
Dim032	1.1963	1.3287	1.2979	0.0318	1.0392	1.3233	1.1582	0.0820	0.8336	1.0919	**0.9994**	0.0609	1.5269	1.6896	1.6268	0.0448
Dim064	1.3861	1.4644	1.4202	0.0212	1.0720	1.4922	1.3217	0.1091	0.7304	1.2098	**0.9983**	0.1290	1.6094	1.8024	1.7134	0.0458
Dim128	1.4726	1.5733	1.5282	0.0282	1.1370	1.5982	1.4900	0.1046	0.8237	1.1232	**1.0760**	0.0658	1.7590	1.8831	1.8147	0.0355
Dim256	1.6022	1.6623	1.6281	0.0180	1.2914	1.7342	1.5953	0.1220	0.9016	1.1451	**1.0639**	0.0788	1.8219	1.9496	1.9028	0.0261
Dim512	1.6834	1.7259	1.7089	0.0137	1.3652	1.8167	1.7082	0.1017	1.0231	1.1717	**1.1166**	0.0553	1.9327	2.0015	1.9623	0.0189
Dim1024	1.7417	1.7856	1.7678	0.0114	1.6986	1.8912	1.8224	0.0481	1.1322	1.2322	**1.2051**	0.0276	1.9629	2.0396	2.0105	0.0193
Flame	0.6605	0.8005	0.7261	0.0318	0.6294	0.8124	0.7329	0.0406	0.5970	0.6974	**0.6496**	0.0449	0.5982	0.7737	0.6715	0.0442
Glass	0.6108	0.7232	0.6768	0.0351	0.3611	0.9226	0.6819	0.1360	0.3336	0.6091	**0.5402**	0.1224	0.6092	0.8351	0.6357	0.0535
Housec5	0.5348	0.6803	0.6190	0.0412	0.6017	0.8465	0.7456	0.0679	0.4987	0.6422	**0.5642**	0.0290	0.6028	0.8081	0.7034	0.0430
Housec8	0.4922	0.5498	0.5245	0.0139	0.5057	0.8082	0.6418	0.0858	0.4559	0.5057	**0.4584**	0.0111	0.5014	0.7206	0.6206	0.0546
Iris	0.5815	0.6204	0.6006	0.0116	0.6001	0.8890	0.7474	0.0700	0.5700	0.7010	**0.5768**	0.0293	0.6835	0.9891	0.8616	0.0815
Jain	0.6472	0.6707	0.6554	0.0047	0.6112	0.7084	0.6534	0.0243	0.6372	0.6514	**0.6446**	0.0048	0.6347	0.6792	0.6541	0.0096
Leaves	0.9200	1.2317	1.1345	0.0798	0.6056	1.4366	1.0335	0.2854	0.5834	0.8520	**0.6036**	0.0587	1.3624	1.5625	1.5132	0.0490
Letter	0.9340	1.0481	0.9852	0.0303	0.8136	1.1798	1.0354	0.1084	0.7756	0.8762	**0.8071**	0.0246	1.1385	1.2771	1.2245	0.0416
Joensuu	0.5067	0.5375	0.5205	0.0089	0.5253	0.6900	0.6069	0.0393	0.4876	0.5961	0.5038	0.0249	0.4957	0.5162	**0.5019**	0.0075
Finland	0.4460	0.4817	0.4575	0.0082	0.4393	0.6601	0.5761	0.0630	0.4393	0.5462	**0.4446**	0.0239	0.4552	0.5829	0.5036	0.0398
Pathbased	0.6569	0.7268	0.6829	0.0161	0.6509	0.7470	0.6905	0.0289	0.6238	0.6892	0.6490	0.0220	0.6358	0.7106	**0.6644**	0.0193
R15	0.6710	0.7939	0.7184	0.0296	0.4540	0.7918	0.5988	0.0935	0.4816	0.7814	**0.5818**	0.0787	0.4894	0.7814	0.6086	0.0808
S1	0.7115	0.7729	0.7363	0.0150	0.5597	0.7173	**0.6472**	0.0470	0.6315	0.6991	0.6793	0.0255	0.6837	0.8306	0.7572	0.0315
S2	0.6741	0.7548	0.7256	0.0221	0.6233	0.7530	0.6878	0.0318	0.6657	0.7391	**0.6814**	0.0229	0.7237	0.8268	0.7625	0.0296
S3	0.7126	0.7839	0.7265	0.0179	0.6890	0.8026	0.7261	0.0282	0.6691	0.7157	**0.7080**	0.0117	0.7219	0.8666	0.7576	0.0333
S4	0.7224	0.7894	0.7635	0.0180	0.6964	0.8072	0.7455	0.0316	0.7103	0.7690	**0.7254**	0.0158	0.7406	0.8119	0.7809	0.0191
Spiral	0.7429	0.8309	0.7707	0.0231	0.7280	0.8022	0.7664	0.0235	0.7270	0.7965	**0.7364**	0.0171	0.7291	0.8021	0.7518	0.0237

Table 7 (continued)

Comparator	DE				PSO				FA				IWO			
	Best	Worst	Avg.	SD	Best	Worst	Avg.	SD	Best	Worst	Avg.	SD	Best	Worst	Avg.	SD
T4.8k	0.0227	0.0231	0.0228	0.0001	0.0227	68.2531	17.5004	22.6863	0.0227	0.0227	**0.0227**	0.0000	0.0302	0.3241	0.1119	0.0665
Thyroid	0.5114	0.5598	0.5373	0.0127	0.4813	0.9402	0.5681	0.1195	0.4813	0.5897	**0.4963**	0.0246	0.7376	1.1867	0.9472	0.1258
Wdbc	0.0508	0.0534	0.0511	0.0006	0.0399	38.9332	13.6881	14.0363	0.0508	0.0508	**0.0508**	0.0000	0.0519	0.2864	0.0901	0.0578
Wine	0.9119	1.0238	0.9519	0.0280	0.8023	1.0812	0.9798	0.0867	0.8000	0.8308	**0.8075**	0.0119	1.0579	1.3447	1.2098	0.0689
Yeast	0.7280	0.9074	0.8053	0.0401	0.4380	0.9704	0.7710	0.1219	0.4379	0.7868	0.5943	0.1173	0.4389	0.7859	**0.5700**	0.1055
Average	0.7684	0.8609	0.8117	0.0240	0.6796	3.5516	1.5818	0.9639	0.6109	0.7566	**0.6749**	0.0415	0.8304	1.0017	0.9021	0.0445

All text in bold represents the best results obtained by the individual representative algorithms

presented in Table 8 show the computational values for the DB fitness function (or validity index) obtained by the hybrid algorithms. In comparing the results across the tables, the superiority of the standard FA over the hybrid FADE can be seen. For example, whereas the standard FA obtained an overall average of 0.6749, the corresponding results for hybrid FADE algorithm was 0.7240. This slight deterioration in the fitness function was however offset by improved computational time. As shown in Table 9, a substantial improvement for the hybrid FADE as compared to that of the single FA. Specifically, in the execution time on the forty datasets a result of 43.22 s were recorded for the FA while 32.17 s was recorded for the FADE. The results of hybrid PSODE and IWODE appear to be superior than their respective single algorithms. For example, while the hybrid PSODE obtained a lower overall average DB validity index across the forty datasets (0.8805) than did the PSO (1.5818), thereby making the hybrid PSODE a better clustering algorithm than the standard PSO algorithm. A similar improvement trend was also observed the overall DB validity index for the hybrid IWODE across the forty datasets (0.8954) which was better than that for the standard IWO (0.9021), thereby making IWODE a better algorithm than the standard IWO algorithm. Contrasting patterns were observed for computation cost for PSODE and IWODE, as shown in Table 9. On the one hand, it was observed that the execution time for PSODE was 25.49 s was slightly better than the 28.82 s for PSO. IWODE, on the other hand, obtained considerably higher execution time of 27.65 as against 16.68 s for the IWO. We can say that in an overall comparison between the hybrid algorithms and their single counterparts, although FADE performed less well, computational time was improved; the hybrid algorithms, PSODE and IWODE experienced performance improvement over their individual algorithms in terms of solution qualities with variable effects on computation time costs.

The corresponding convergence curves of the algorithms as observed in the second experiment for some selected ground truth datasets are presented in Fig. 12. Clearly, it can be seen from these figures that the FA algorithm has still maintained a lead with better convergence, having been able to produce consistently smooth and rapidly declining curve, which is an indication of its superior performance in all the ten datasets. In Fig. 13, similar convergence curves are presented for the hybrid algorithms. The hybrid FADE appears to be better because it has all the signs of better performance as described above with respect to convergence curves. However, the figures also show that other hybrid algorithms have good performance as well or have additional improvements as compared to their individual standard algorithm, which is the case for PSO and IWO algorithms.

Table 8 Computational results for three hybrid algorithms on forty-one datasets

Dataset	PSODE				FADE				IWODE			
	Best	Worst	Avg.	SD	Best	Worst	Avg.	SD	Best	Worst	Avg.	SD
A1	0.5901	0.6194	**0.5949**	0.0086	0.5901	0.6932	0.6271	0.0347	0.5958	0.8102	0.6525	0.0621
A2	0.6777	0.7305	**0.6912**	0.0161	0.6776	0.7475	0.6976	0.0215	0.6820	0.8337	0.7296	0.0391
A3	0.6706	0.7443	0.7106	0.0176	0.6672	0.7765	**0.7085**	0.0332	0.7124	0.8433	0.7527	0.0319
Aggregation	0.5359	0.6614	0.5958	0.0391	0.5351	0.7088	**0.5948**	0.0424	0.5450	0.7423	0.6015	0.0510
Birch1	0.6882	0.8215	0.7256	0.0276	0.6872	0.7707	**0.7232**	0.0257	0.7367	0.8200	0.7692	0.0279
Birch2	0.5070	0.5076	**0.5070**	0.0002	0.5070	0.6009	0.5155	0.0235	0.5078	0.5382	0.5176	0.0084
Birch3	0.6780	0.7367	0.7074	0.0151	0.6605	0.7385	**0.7012**	0.0191	0.7221	0.8290	0.7570	0.0247
Breast	0.6519	1.0049	**0.7256**	0.1115	0.6519	1.1464	0.9378	0.2051	0.6686	1.1162	0.7965	0.1177
Bridge	0.6120	0.8742	0.7141	0.1007	0.6110	0.8365	**0.6405**	0.0709	1.0007	1.2585	1.1397	0.0666
Compound	0.4932	0.5271	**0.4983**	0.0092	0.4932	0.6594	0.5273	0.0576	0.4932	0.6330	0.5004	0.0312
D31	0.7174	0.8562	0.8021	0.0407	0.7144	0.8520	**0.7788**	0.0376	0.6839	0.8803	0.7972	0.0514
Dim002	0.5246	0.6900	**0.5975**	0.0445	0.4883	0.7081	0.6280	0.0607	0.5671	0.7248	0.6705	0.0432
Dim016	0.6515	1.1677	**1.0183**	0.1285	0.9760	1.1531	1.0413	0.0512	1.3327	1.5160	1.4336	0.0456
Dim032	1.0182	1.2774	1.1142	0.0915	0.8146	1.2258	**1.0727**	0.0894	1.4861	1.6508	1.5731	0.0574
Dim064	0.7713	1.4420	1.1918	0.1769	0.9074	1.3781	**1.1003**	0.0925	1.6148	1.7762	1.7015	0.0415
Dim128	1.1501	1.5584	1.3363	0.1425	0.9939	1.4660	**1.1980**	0.1173	1.7029	1.9032	1.7773	0.0477
Dim256	1.1083	1.6523	1.5051	0.1740	1.1395	1.5445	**1.2938**	0.1308	1.7924	1.9085	1.8603	0.0309
Dim512	1.4725	1.7304	1.6827	0.0633	1.1352	1.7415	**1.3529**	0.1463	1.8623	1.9995	1.9311	0.0424
Dim1024	1.7457	1.7813	1.7644	0.0112	1.2608	1.6450	**1.4759**	0.1200	1.9124	2.0095	1.9654	0.0261
Flame	0.6297	10.4053	4.4069	4.6725	0.5971	0.7642	**0.6795**	0.0477	0.6051	0.7737	0.6749	0.0391
Glass	0.3336	0.6727	**0.5966**	0.0672	0.3336	0.7931	0.5971	0.0996	0.3339	0.8085	0.5562	0.1384
Housec5	0.4987	11.9045	2.2408	4.0861	0.4987	0.6263	**0.5647**	0.0287	0.6509	0.7292	0.6865	0.0229
Housec8	0.4559	0.5534	0.5022	0.0315	0.4559	0.6011	**0.4707**	0.0383	0.5488	0.7183	0.6344	0.0408
Iris	0.5700	0.6443	**0.5811**	0.0210	0.5700	0.6811	0.5850	0.0350	0.6739	1.0258	0.8384	0.0916
Jain	0.6040	0.6530	**0.6399**	0.0150	0.6341	0.6514	0.6451	0.0062	0.6371	0.6622	0.6500	0.0058
Leaves	0.6100	1.0202	0.7580	0.1514	0.6317	1.1548	**0.7483**	0.1539	1.4255	1.5770	1.5116	0.0396
Letter	0.8240	1.0239	0.9121	0.0628	0.7921	1.0605	**0.8665**	0.0663	1.1094	1.3649	1.2057	0.0571
Joensuu	0.4931	0.5316	**0.5094**	0.0098	0.4876	0.6313	0.5100	0.0327	0.4877	0.4991	0.4972	0.0041
Finland	0.4393	0.4598	**0.4465**	0.0060	0.4393	0.6094	0.4686	0.0547	0.4490	0.5948	0.4864	0.0371
Pathbased	0.6260	0.6824	**0.6526**	0.0166	0.6299	0.6826	0.6561	0.0178	0.6299	0.6899	0.6567	0.0156
R15	0.4891	0.7566	0.6423	0.0722	0.5010	0.8105	**0.6399**	0.0984	0.4955	0.8312	0.5996	0.0894
S1	0.5663	0.7202	**0.6739**	0.0351	0.6288	0.7139	0.6756	0.0270	0.6990	0.8028	0.7501	0.0280
S2	0.6161	0.7225	**0.6844**	0.0280	0.6574	0.7543	0.6939	0.0345	0.7224	0.7948	0.7556	0.0190
S3	0.6701	0.7411	0.7106	0.0199	0.6691	0.7258	**0.7072**	0.0181	0.7243	0.8277	0.7559	0.0317
S4	0.6980	0.7626	**0.7299**	0.0162	0.7103	0.7771	0.7356	0.0226	0.7431	0.8733	0.7896	0.0303
Spiral	0.7285	0.7786	0.7441	0.0136	0.7237	0.7779	**0.7373**	0.0156	0.7278	0.7758	0.7458	0.0143
T4.8k	0.0227	0.0227	**0.0227**	0.0000	0.0227	0.4170	0.0424	0.0882	0.0279	0.2337	0.0928	0.0515
Thyroid	0.4797	0.5353	0.5024	0.0196	0.4814	0.5905	**0.4955**	0.0241	0.8037	1.2485	0.9893	0.1077
Wdbc	0.0508	0.0508	**0.0508**	0.0000	0.0508	0.0508	**0.0508**	0.0000	0.0513	0.1304	0.0814	0.0263
Wine	0.8020	0.9794	0.8891	0.0601	0.7520	1.1008	**0.8648**	0.0973	1.1298	1.3099	1.2312	0.0525
Yeast	0.5586	0.7864	0.7193	0.0677	0.4379	0.8119	**0.6375**	0.1344	0.4386	0.8043	0.5949	0.1144
Average	0.6593	1.3364	0.8805	0.2608	0.6394	0.8580	**0.7241**	0.0615	0.8228	0.9968	0.8954	0.0464

All text in bold represents the best results obtained by the individual representative algorithms

5.4 Experiment 3: Parameter fine-tuning

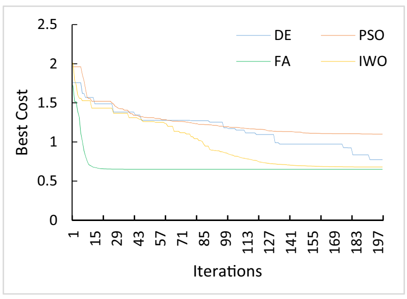
The third experiment involved evaluating the impact of certain predefined control parameters, which might affect the performance of the individual algorithm, either negatively or positively. Fine-tuning is widely used in metaheuristic optimization techniques to evaluate the effects of certain control parameters on the performances

Table 9 Computational time cost for all the algorithms

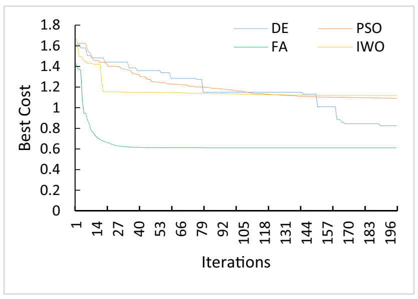
Algorithms	Standard Implementation				Hybrid Implementation		
	DE	PSO	FA	IWO	PSODE	FADE	IWODE
A1	19.9490	21.9353	23.1691	14.4312	40.7302	19.0999	21.2270
A2	20.6826	21.3371	27.7366	15.4780	47.1239	22.0312	22.4148
A3	22.1976	21.7788	33.0137	16.3230	33.1377	24.0918	24.5065
Aggregation	44.4496	26.0378	21.0443	14.3309	10.3857	15.9027	17.9663
Birch1	110.3298	79.6950	242.8657	43.1278	144.1875	191.3201	114.4691
Birch2	47.1180	58.7877	171.2215	32.4669	78.5766	119.5048	49.3644
Birch3	64.1067	96.7476	229.0219	36.1856	125.5172	176.6250	64.3187
Breast	18.5755	19.8743	20.7709	13.4494	10.0605	15.7322	14.1894
Bridge	24.5326	22.2760	33.1045	14.1580	14.6671	23.3594	19.5509
Compound	19.0033	19.6271	18.8446	17.6630	9.3736	15.2473	21.7213
D31	19.1009	23.5727	27.6833	14.7797	19.3097	20.2853	21.2679
Dim002	18.5908	19.4369	19.3585	13.9877	18.9924	15.8502	19.6408
Dim016	17.7505	19.0110	19.1723	13.2421	8.9427	15.7862	17.3102
Dim032	17.0615	24.6876	19.3937	17.2166	9.0909	15.4261	18.3960
Dim064	17.3968	19.1155	20.3916	13.2648	9.2795	15.4340	34.7302
Dim128	17.0821	19.4855	29.7310	13.2505	9.5850	18.9326	21.0818
Dim256	17.6940	19.4731	37.8332	17.5826	10.2746	22.3294	21.6856
Dim512	18.3598	20.1388	48.6561	13.3380	12.2050	27.3212	21.9441
Dim1024	35.4490	21.3741	71.7511	13.1987	16.1302	37.6609	22.4156
Flame	17.7655	23.2215	19.4257	13.3214	10.1488	15.2684	21.4066
Glass	17.5845	17.7427	17.9051	17.0496	9.4288	14.7100	33.5422
Housec5	17.9596	23.9175	23.8241	14.8858	11.3120	17.0716	26.6307
Housec8	27.2971	31.5027	71.8233	17.6232	34.2852	54.4047	40.5323
Iris	16.5722	18.5352	16.8701	13.5175	9.6620	14.9608	24.5881
Jain	16.5459	18.6040	18.9921	13.6465	10.4927	15.2563	17.7379
Leaves	19.0131	49.5173	28.5303	16.1148	14.3337	22.9569	43.9606
Letter	27.8214	48.6456	113.7824	19.2361	40.6652	78.3855	58.7786
Joensuu	18.9126	24.0024	28.9638	14.0140	12.4565	22.2184	28.5891
Finland	20.8059	25.7042	36.6961	15.0975	17.5404	28.9074	20.9981
Pathbased	17.5430	28.0911	19.0709	14.1964	10.5311	15.2590	19.2079
R15	16.6841	20.2720	20.9732	13.6924	10.9669	15.4085	19.7826
S1	26.6109	28.7531	29.0218	16.4213	22.9561	23.1972	22.3291
S2	19.8328	20.7292	30.5368	15.6235	25.1369	23.3005	20.4949
S3	19.6668	21.4548	26.0717	15.4535	24.6599	21.4080	20.4787
S4	21.0218	39.6490	28.3019	15.6406	23.8855	24.5190	22.4664
Spiral	17.6803	30.7869	19.2370	17.4290	24.5136	15.3566	28.7818
T4.8k	21.2385	43.0841	28.9985	14.8247	25.1864	20.6794	23.6252
Thyroid	17.7458	22.4455	17.1986	17.4268	18.1776	14.8244	17.5192
Wdbc	32.6544	22.8051	21.5963	13.5909	21.7783	17.5456	17.3334
Wine	18.1459	23.4927	18.9496	13.4588	20.4608	15.3653	17.1978
Yeast	18.3390	24.1620	20.3478	14.1749	19.0508	16.0801	19.3065
Average	24.80174	28.81732	43.2166	16.68082	25.49266	32.1713	27.64606

of the representative algorithms. Therefore, for the clustering analysis problem discussed in this paper, parameter fine-tuning is restricted to determining only the effect of population size of the respective algorithms on quality of solution improvements. The performances of the representative algorithms with fine-tuning of population sizes of 50, 100, and 150 is shown in Tables 10, 11 and 12, respectively. For the four algorithms, noticeable performance improvements in the solution quality was observed for DE, PSO and IWO, while there was no any significant

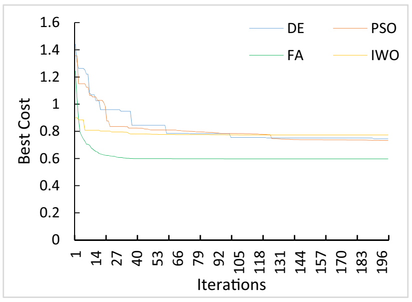
Fig. 12 Convergence curves of each single algorithm on selected ground truth dataset (based on objective function values)



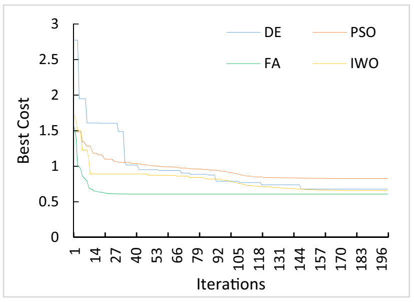
a Breast dataset



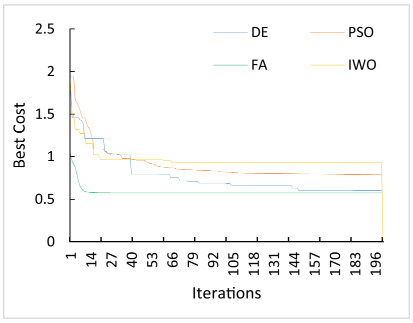
b Bridge dataset



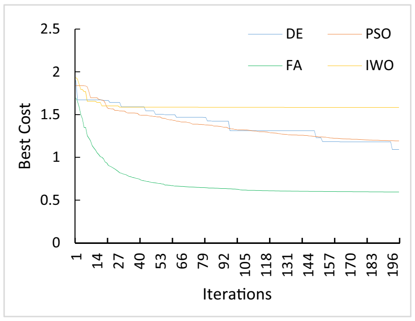
c Flame dataset



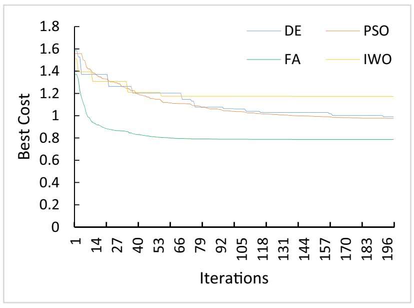
d Glass dataset



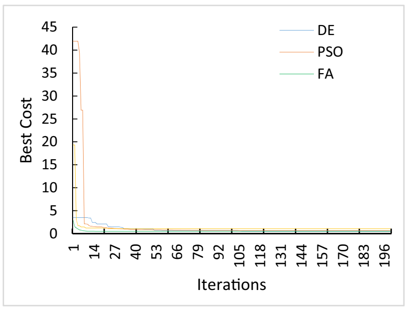
e Iris dataset



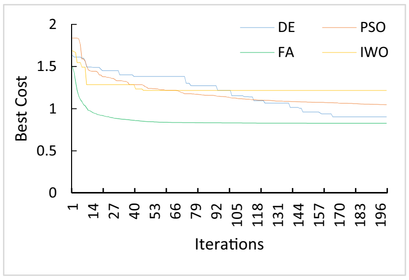
f Leaves dataset



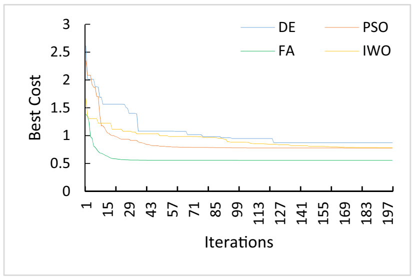
g Letter dataset



h Thyroid dataset

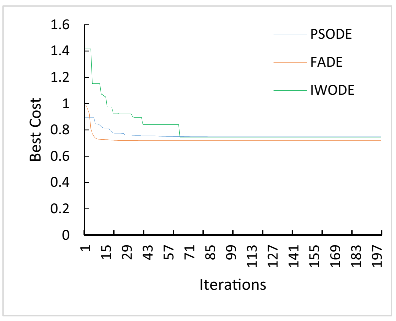


i Wine dataset

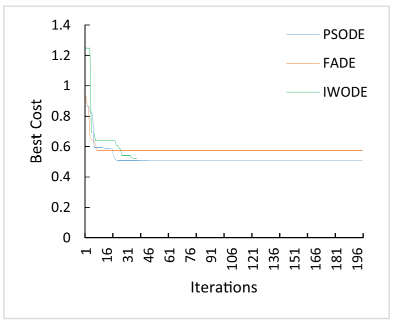


j Yeast dataset

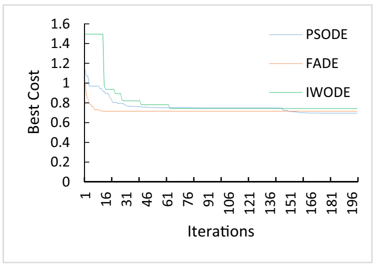
Fig. 13 Convergence curves of each hybrid algorithm on ten test selected datasets (based on objective function values)



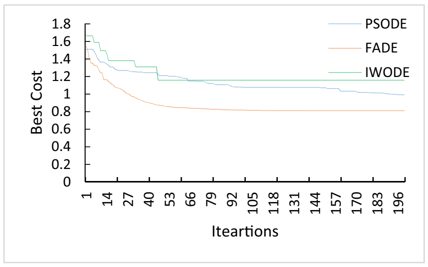
a Birch1 dataset



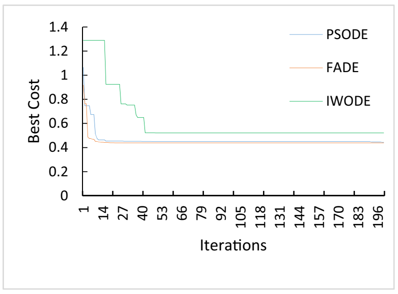
b Birch2 dataset



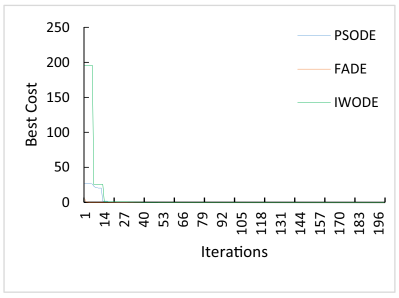
b Birch3 dataset



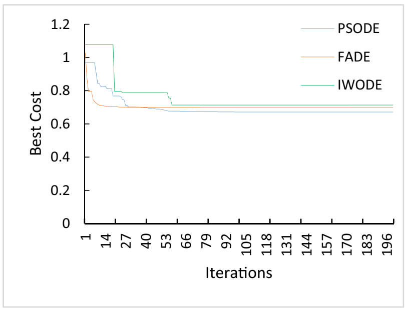
c Letter dataset



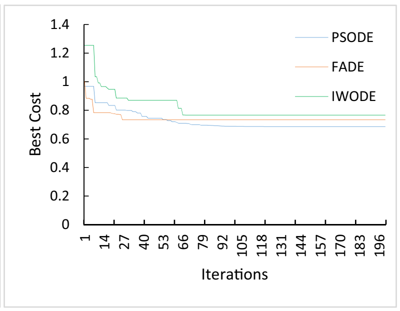
d Finland dataset



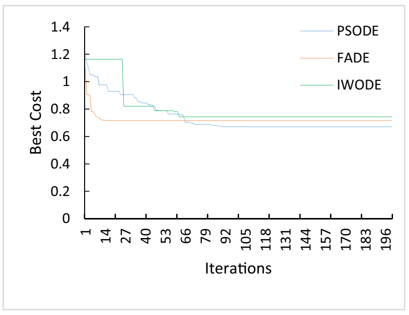
e T4.8k dataset



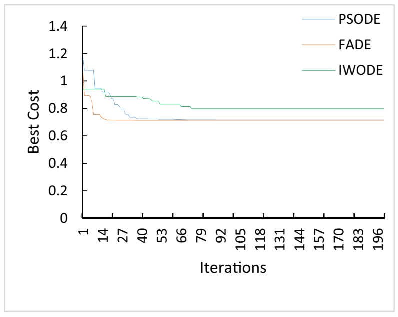
f S1 dataset



g S2 dat aset



h S3 dataset



i S4 dataset

Table 10 Numerical solutions obtained for all algorithms using population size of 50

Datasets	N	DE		PSO		FA		IWO		PSODE		FADE		IWODE	
		Solution	Time	Solution	Time	Solution	Time	Solution	Time	Solution	Time	Solution	Time	Solution	Time
Bridge	4096	0.8491	16.2538	1.0276	15.3392	0.6113	106.8033	1.1003	14.8019	0.6734	27.0772	0.6109	97.1574	0.9997	23.4020
Joensuu	6014	0.5219	15.1827	0.6183	15.9029	0.4899	92.2198	0.4991	14.1198	0.4991	21.8963	0.4876	83.5029	0.4991	22.2471
T4.8k	8000	0.0228	15.8349	0.0227	24.437	0.0227	94.1209	0.0969	15.8536	0.0227	21.7629	0.0227	76.9567	0.0386	24.5677
Finland	13,467	0.4519	16.3046	0.5953	16.1333	0.4393	92.2198	0.524	16.3401	0.4394	28.0952	0.4393	97.1461	0.5090	24.4788
Letter	20,000	0.4614	29.3672	1.0548	20.5335	0.8001	425.0568	1.1766	15.2789	0.9194	67.6421	0.7829	317.1155	1.1752	49.3524
Housec5	34,112	0.6027	14.6923	0.7696	17.7993	0.5386	70.4008	0.6763	13.6958	0.4987	19.0370	0.5652	47.0923	0.6569	21.4191
Housec8	34,112	0.5357	20.6176	0.6602	19.0733	0.4559	251.2809	0.5992	23.696	0.4559	47.9596	0.4559	244.5398	0.5485	40.8230
Birch1	100,000	0.7429	62.7887	0.7508	37.7368	0.6872	1089.5485	0.7558	65.0492	0.6997	163.7943	0.7189	818.3993	0.8085	120.1747
Birch2	100,000	0.508	45.1217	0.6214	30.0169	0.507	625.4514	0.5163	50.1053	0.5070	149.0613	0.5070	637.6880	0.5142	94.1641
Birch3	100,000	0.7271	60.1991	0.7076	40.2529	0.716	715.515	0.7596	66.4509	0.6789	174.5673	0.6911	876.4011	0.7372	104.7143
Average		0.5424	29.6363	0.6828	23.7225	0.5268	356.2617	0.6704	29.5392	0.5394	72.0893	0.5282	329.5999	0.6487	52.5343

Table 11 Numerical solutions obtained for all algorithms using population size of 100

Datasets	N	DE		PSO		FA		IWO		PSODE		FADE		IWODE	
		Solution	Time	Solution	Time	Solution	Time	Solution	Time	Solution	Time	Solution	Time	Solution	Time
Bridge	4096	0.8347	16.9282	0.7503	16.3347	0.6109	422.6785	1.0965	17.188	0.6116	38.2201	0.6109	221.3055	1.0494	28.6025
Joensuu	6014	0.5143	16.2367	0.5567	15.7982	0.4899	392.7489	0.4945	16.3034	0.4913	33.4703	0.4876	251.3377	0.4991	25.7164
T4.8k	8000	0.0227	16.6253	0.0227	16.3172	0.0227	409.2651	0.0534	19.8222	0.0227	30.7366	0.0227	248.9999	0.0583	36.9657
Finland	13,467	0.4529	20.8171	0.5396	22.6746	0.4393	526.2442	0.4818	20.6368	0.4393	44.9168	0.4393	237.9715	0.4738	40.5364
Letter	20,000	0.9856	54.4685	0.9091	48.3691	0.7724	1795.8947	1.1224	49.0817	0.7872	124.1709	0.7950	971.6205	1.1698	75.5472
Housec5	34,112	0.5629	15.1463	0.7107	14.4336	0.4987	257.9266	0.6446	14.7786	0.5598	23.3929	0.4987	184.3469	0.6760	21.9806
Housec8	34,112	0.5198	37.5307	0.599	46.0475	0.4559	1187.0499	0.569	34.7457	0.5209	86.9351	0.4559	1025.8784	0.5151	72.5537
Birch1	100,000	0.7331	118.5387	0.7235	113.0982	0.7121	3857.5953	0.755	117.4207	0.6875	351.0767	0.7186	1855.6013	0.7304	277.3295
Birch2	100,000	0.5073	90.5532	0.567	78.6397	0.5070	2533.9347	0.5136	95.7828	0.5070	240.1282	0.5070	1556.5890	0.5103	284.7349
Birch3	100,000	0.7212	125.3848	0.6809	115.1825	0.6788	3394.0758	0.7436	112.0553	0.6972	367.0218	0.6911	1710.5610	0.7277	157.8827
Average		0.5855	51.223	0.6060	48.6895	0.5188	1477.7414	0.6474	49.7815	0.5324	134.0070	0.5227	826.4212	0.6410	102.1850

Table 12 Numerical solutions obtained for all algorithms using population size of 150

Datasets	N	DE		PSO		FA		IWO		PSODE		FADE		IWODE	
		Solution	Time	Solution	Time	Solution	Time	Solution	Time	Solution	Time	Solution	Time	Solution	Time
Bridge	4096	0.8274	20.5371	0.8046	20.1126	0.6109	869.9744	1.0452	21.0303	0.8031	56.2274	0.6109	644.7866	1.0453	38.3651
Joensuu	6014	0.5101	19.4738	0.5520	20.8018	0.4876	860.5269	0.4968	22.1531	0.5156	50.5109	0.4876	718.8054	0.4991	32.0925
T4.8k	8000	0.0227	17.5294	0.0227	17.6834	0.0227	857.1900	0.0539	26.7567	0.0227	53.2101	0.0227	633.6639	0.0357	59.1146
Finland	13,467	0.4476	27.0377	0.5585	41.6831	0.4393	1093.5739	0.4743	29.9405	0.4394	69.3492	0.4393	792.0254	0.4869	74.7818
Letter	20,000	0.9645	75.4547	0.9254	15.3823	0.7916	3920.7564	1.1339	70.5682	0.8236	200.9118	0.8071	2698.0223	1.1960	233.7011
Housec5	34,112	0.5421	14.6611	0.6689	66.0767	0.4987	565.5380	0.6425	15.8770	0.5903	32.3947	0.4987	348.8764	0.6212	46.1400
Housec8	34,112	0.4997	49.3765	0.6376	15.3823	0.4559	2203.7219	0.5692	50.3676	0.5057	164.0460	0.4559	2768.9353	0.5828	134.0697
Birch1	100,000	0.7332	176.8616	0.7196	178.9059	0.6874	8057.0988	0.7402	167.6180	0.7186	531.0487	0.7081	5512.1880	0.7660	288.5896
Birch2	100,000	0.5073	121.5381	0.5801	146.9822	0.5070	6050.3562	0.5127	133.1948	0.5070	390.2735	0.5070	3490.2260	0.5209	211.0129
Birch3	100,000	0.7224	168.0392	0.6762	200.4805	0.6911	7444.1531	0.7316	151.6145	0.7153	592.1492	0.6787	4712.1929	0.7242	380.0165
Average		0.5777	69.0509	0.6146	72.3491	0.5192	3192.2890	0.6400	68.9121	0.5641	214.0121	0.5216	2231.9722	0.6478	149.7884

improvement for the FA algorithm. In most cases, the improvements are apparent for the three aforementioned algorithms on the ten datasets considered. However, the lack of substantial improvement for the FA algorithm simply means that the algorithm is not tied to any control parameter such as population size. Similarly, considerable improvement in the solution quality for the hybrid algorithms namely PSODE and IWODE was significant and increased as the population size increased. However, the improvements of these two hybrid algorithms were at the expense of computational time, which increased significantly as shown in the three tables below. There was no significant improvement in terms of clustering solution quality for the hybrid FADE, which justifies the above claim that the FA is not restricted by any control parameter, unlike in the case of PSODE and IWODE algorithms, in which clustering solution qualities were greatly affected by the fine-tuning process.

The computational costs for implementation of all four representative algorithms are presented alongside the relevant clustering solutions obtained, in Tables 10, 11, and 12. One of the major drawbacks of parameter fine-tuning is that the running time grows considerably for each algorithm. Considering the implementation time for the FA algorithm for instance, even though it produced the best clustering solution in terms of cohesion and compactness, its execution times grew exponentially with corresponding increase in population size. Similar computation cost characteristics were also displayed by the hybrid PSODE, FADE and IWODE algorithms. It was noticeable that the three hybrid algorithms recorded considerable increase in computation time, although with remarkable improvement in clustering solution quality. Nevertheless, this is as expected, in view of the hybrid implementation process having introduced additional subroutine processing overheads, based on the combined algorithmic computational complexity.

In "Appendix 2", Figs. 21, 22 and 23 provide two dimensional visual impressions of the performance of the different clustering methods after 200 iterations over selected datasets. Generally, the quality of clustering solutions obtained by the representative algorithms are test based on their capability to successfully separate the classes of data points into clusters. Ideally, mixed clusters are an indication of poor clustering solutions, while distinct clusters show good clustering solution. In the figures illustrated in the appendix, each color represents a different class of cluster. "Appendix 2" shows the clustering quality obtained by the DE algorithm, for which it can be clearly seen that the DE was able to properly separate the dataset into a set of unique classes as shown in the represented dataset. However, there are some instances of overlaps, such as the clustering of leaves and bridge ground truth dataset.

Figures 22 and 23 ("Appendix 2") provide a comparison of the clustering quality between PSO and FA using a population size of 150. It can be seen that even though both algorithms were able to distinctively separate the clusters into unique classes, the FA produced better clusters with high cohesion and compactness. Also note that overlapping colored clusters indicate data points that are common between clusters, which is a sign of only partial or incomplete separation, indicating poor clustering solutions.

To determine the performance stability of the individual algorithms, further qualitative representations of the performance of the four algorithms clustering methods over A set, birch set and ground truth dataset (with high dimensionality classifications) were plotted, as shown in Fig. 24 in the "Appendix 2". Although each algorithm maintained performance stability with respect to the quality of clustering results obtained, cohesion and compactness of the clusters classifications were compromised for all the algorithms with several overlaps appearing in each case. Note that other algorithms produced good clustering solutions, however they are not represented here because of space and page limit. In summary, it was observed that all the representative algorithms were able to find non-convex clusters and clusters with unusual shapes including those clusters with different densities. Specifically, among the single objective algorithms, the FA and DE have the capability of handling overlapped clusters among the different dataset considered in this paper, which notably, does includes those datasets with high dimensions and large number of classification and features. Then among the hybrid algorithms, the FADE and PSODE are more capable of dealing with overlapped clusters, while the

Table 14 Computational time cost for all the single algorithms

Dataset	Algorithm			
	DE	PSO	FA	IWO
A1	22.2515	26.4103	75.1581	20.5279
A2	27.7950	33.0206	228.8125	27.4031
A3	32.3685	39.7205	203.9804	34.3687
Birch 1	412.9934	643.5050	2384.2171	410.3153
Birch 2	455.2650	696.9305	1514.3727	380.6237
Birch 3	606.0008	745.8625	2351.4935	515.6563
Bridge	87.7046	169.1921	1240.4218	164.5622
D31	22.5631	35.3123	93.9079	36.6941
Housec5	46.6681	154.0473	279.3254	75.0907
Housec8	477.9579	690.6298	2868.3422	404.5708
Leaves	54.8537	147.5045	372.6992	65.2023
Letter	41.8935	50.3511	258.0772	153.8091
Avg.	190.6929	286.0405	989.2340	190.7354

Table 13 Numerical solutions obtained for all the single algorithms using high dimensionality classification datasets

Algorithm	Dataset											
	A1	A2	A3	Birch 1	Birch 2	Birch 3	Bridge	D31	Housec5	Housec8	Leaves	Letter
DE												
Best	0.5928	0.9047	1.1036	1.4144	63.7233	1.5022	41.3745	0.9701	157.6535	178.1353	34.2424	1.0093
Worst	0.8387	1.0695	1.2696	1.5782	189.8872	1.7344	44.8408	1.1600	190.9515	198.4978	37.1282	1.3927
Average	0.6695	0.9994	1.1801	1.4987	146.8673	1.5991	43.7495	1.0822	178.5097	185.9484	35.6751	1.1607
SD	0.0738	0.0470	0.0477	0.0506	38.5605	0.0652	0.8229	0.0534	8.1516	5.2336	0.8075	0.1028
PSO												
Best	0.6748	0.7263	0.8033	0.9467	118.6541	0.8553	56.8444	0.6563	104.3547	109.4072	47.8092	1.1782
Worst	0.6305	0.6549	0.6857	0.8745	59.0091	0.7210	53.7803	0.5178	89.0581	91.0613	45.4037	1.0967
Average	0.7416	0.8212	1.2212	1.0334	212.5939	1.0164	59.1720	0.7534	119.9863	127.3803	49.4539	1.2839
SD	0.0344	0.0486	0.1071	0.0507	44.6044	0.0785	1.6025	0.0553	8.4766	10.5152	0.9506	0.0579
FA												
Best	0.5901	0.6342	0.6511	0.8062	0.5731	0.6778	21.3250	0.5975	63.1054	65.9173	1.5188	0.7557
Worst	0.6889	0.7154	0.7495	0.8450	0.9123	0.7823	28.1639	0.7965	91.5772	108.9068	26.1365	0.8937
Average	**0.6210**	**0.6772**	**0.6967**	**0.8219**	**0.7052**	**0.7301**	**25.8353**	0.6967	**78.5035**	**83.3241**	19.6928	**0.8285**
SD	0.0321	0.0204	0.0237	0.0122	0.1211	0.0262	1.8231	0.0561	7.8922	10.8563	4.8325	0.0312
IWO												
Best	0.5980	0.7908	0.8855	1.0223	1.1191	1.0064	34.7325	0.7331	88.6939	103.0494	27.3061	1.6253
Worst	0.8670	0.9614	1.0349	1.1894	48.1364	1.1405	40.8760	0.9127	134.5058	137.1766	36.5574	22.3264
Average	0.6858	0.8742	0.9458	1.0873	3.8025	1.0844	37.7892	0.8254	113.2733	117.9512	31.4023	2.7513
SD	0.0804	0.0375	0.0470	0.0404	10.4376	0.0359	1.9805	0.0478	11.4787	8.8246	2.2636	4.6078

All text in bold represents the best results obtained by the individual representative algorithms

FADE also has a lower computation time compared to its single-based FA algorithm because of the influence on the DE on the computation of objective function. Moreover, the FA does not require any parameter tuning to find optimal clustering solutions.

Tables 13 and 14 show the numerical clustering solutions obtained by the representative algorithm on high dimensionality classification datasets and computational cost incurred by the respective algorithms. The results show that the FA algorithm obtains the least DB validity index and so outperforms DE, PSO and IWO algorithms. The best, worst, average and standard deviation values of the final inter-cluster distance over 20 independent runs for each algorithm is reported. A total of twelve datasets with high classification features were used to evaluate the effectiveness of the algorithms. It is interesting to note that substantial performance differences occur for this set of more challenging clustering datasets with a large number of data points and clusters. According to the average inter-cluster distance results reported for the individual algorithm in Table 13, the FA obtained the minimum average inter-cluster distance value of 17.7611 over all the twelve datasets, followed by the IWO with average inter-cluster distance value of 26.0394. However, PSO and DE obtained the highest average values of 47.9548 and 49.9116, respectively, thereby making them the least effective methods for clustering high density datasets with large numbers of clusters. Furthermore, FA produced the best results with the least standard deviation, confirming it to have the best convergence and, most importantly, the best choice for the aforementioned class of complex dataset. The results reported in Table 14 show the computation cost for the four algorithms. The results indicate that the FA, despite obtaining the least DB validity index values, did that at the expense of time; it can be seen to have the highest computational cost compared to IWO and DE, which had the least execution time for the twelve datasets.

5.5 Experiment 4: Comparison with literature results

In the fourth experiment, an extensive study was carried out, where the representative algorithms were compared with other state-of-the-art clustering algorithms, namely, automatic clustering DE (ACDE) algorithm [13], dynamic clustering PSO (DCPSO) [39], genetic clustering with an unknown number of clusters K (GCUK) [43], and classical DE clustering algorithm [67]. The comparisons of the representative algorithms with literature results is based on only the quality of solution, which is determined in this case by the DB and CS validity indices. Comparisons based on computational time are left out because some of the experimental requirements, such as computer system specifications and programming environment used, for both proposed and existing studies vary significantly. For example, while the proposed study was coded in MATLAB 2018 and executed using 8 GM RAM @ 2.80 GHz, the literature study was coded in a Visual C++ platform on 2 GB RAM @ 2.2 GHz. However, the computational time required to find the solution for DB and CS measures are presented for the represented algorithms. Nevertheless, to enhance fairness in comparing speed of execution for all the proposed algorithms, a similar implementation strategy was adopted for all by using the same population size of 40 and maintaining the same inner loop coding structure to ensure a uniform amount of work done for all the algorithms. Each algorithm execution was replicated for 40 independent runs. The results have been reported in each case in terms of average solution values and standard deviations over the 40 runs.

The results reported in Table 15 show that the proposed algorithms were able to obtain the lowest values for the DB and CS validity indices, except for the iris and breast cancer datasets where the ACDE obtained the least values, being better than for the proposed algorithms. However, for the breast cancer dataset, some discrepancies exist in the features of the dataset version used for the two studies. For example, while $N = 699$ for the current study, $N = 683$ for the existing study and this would have affected the quality of the final results reported for the two studies as shown in Fig. 15. Overall, the proposed study showed superior performance of the representative algorithms, as compared to the competitor algorithm results across the four datasets based on the DB and CS validity indices measures. A summary of the clustering performance comparison between the proposed and the existing competitor algorithms on iris, wine, breast and glass dataset over 40 runs using DB and CS validity indices as fitness functions is illustrated in Figs. 14 and 15. More so, the linear trend lines clearly show the individual algorithm steadiness in terms of increasing and decreasing performance pattern with respect to average fitness function results based on DB and CS indices.

Figures 14 and 15 clearly revealed that the performance of the proposed algorithms, including the hybrid PSODE and FADE, maintained consistent improvements even as the dimensionality of the dataset increased. For example, in the case of the wine dataset, according to the DB and CS validity indices, all the representative algorithms performed exceptionally well when compared to the competing algorithms. More so, it can be seen that the best averages of the DB and CS indices are all below 0.70 with dimensionality greater than ten for the wine dataset. Overall, the performance of the representative algorithms can be said to be superior than that of the existing algorithms, considering that each of the algorithms obtained fitness functions averages that are far below 0.90, except for the IWO and IWODE. However, the

Table 15 Comparison between representative algorithms and literature results with the DB and CS validity indices-based fitness functions

Datasets	Algorithm	DB Index		CS Index	
		Cost	SD	Cost	SD
Iris	ACDE	0.465	0.022	0.664	0.097
	DCPSO	0.690	0.008	0.736	0.671
	GCUK	0.738	0.065	0.728	2.003
	Classical DE	0.582	0.067	0.763	0.039
	DE	0.570	0.000	0.531	0.000
	PSO	0.570	0.000	0.659	0.094
	FA	0.570	0.000	0.532	0.003
	IWO	0.805	0.124	0.862	0.119
	PSODE	0.570	0.000	0.531	0.000
	FADE	0.570	0.000	0.531	0.000
	IWODE	0.920	0.070	0.862	0.119
Wine	ACDE	3.043	0.021	0.925	0.032
	DCPSO	4.343	0.232	1.872	0.037
	GCUK	5.342	0.343	1.584	0.328
	Classical DE	3.392	0.092	1.796	0.802
	DE	0.808	0.003	0.680	0.051
	PSO	0.813	0.015	0.883	0.000
	FA	0.802	0.001	0.790	0.095
	IWO	1.262	0.020	1.128	0.197
	PSODE	0.801	0.001	0.679	0.048
	FADE	0.801	0.001	0.693	0.038
	IWODE	1.237	0.046	1.128	0.197
Breast	ACDE	0.520	0.006	0.453	0.034
	DCPSO	0.575	0.007	0.485	0.009
	GCUK	0.633	0.002	0.609	0.016
	Classical DE	0.520	0.007	0.898	0.381
	DE	0.652	0.000	0.600	0.000
	PSO	0.652	0.000	0.918	0.191
	FA	0.652	0.000	0.601	0.002
	IWO	0.652	0.000	0.966	0.205
	PSODE	0.652	0.000	0.600	0.001
	FADE	0.652	0.000	0.600	0.001
	IWODE	0.652	0.000	0.966	0.205
Glass	ACDE	1.009	0.083	0.332	0.487
	DCPSO	1.515	0.073	0.764	0.073
	GCUK	1.837	0.034	1.474	0.236
	Classical DE	1.667	0.004	0.778	0.643
	DE	0.458	0.073	0.061	0.000
	PSO	0.649	0.089	0.062	0.003
	FA	0.336	0.017	0.061	0.000
	IWO	0.609	0.000	0.061	0.000
	PSODE	0.384	0.084	0.061	0.000
	FADE	0.334	0.045	0.061	0.000
	IWODE	0.554	0.123	0.061	0.000

performance of the competitor based clustering algorithms seem to deteriorates as the dimensionality of the datasets increases. For example, for the wine and glass datasets, the best means of the DB and CS indices are above 1.00 in most instances.

The computational time consumed by the proposed algorithms to find the solutions reported in Table 15 above is illustrated in Figs. 16 and 17 for the representative and hybrid algorithms, respectively. In each figure, it is clearly revealed that despite the FA having performed fairly well, the algorithm is highly disadvantaged by high cost of computation time as compared to other algorithms. Overall, it is obvious that among the four representative algorithms presented in the forementioned comparison, the DE and PSO appeared to have performed better across the board in terms of the methods with the least computation cost, according to both DB and CS evaluation metrics. Furthermore, the IWO also showed better performance with respect to execution speed. In the case of the hybrid algorithms, the FADE still performed poorly, while the PSODE outperformed the other two hybrid algorithms. A general observation was that the CS validity measure was more computationally expensive than the DB validity measure. More so, with the breast cancer dataset, it consumed more time for all the algorithms to find the good quality solutions and this could be attributed to dataset being of high density with 699 individual data points and 9-dimensions. In summary, the overall computation speed performance evaluation of the representative and hybrid algorithms seems to be attractive and they can therefore be recommended for solving clustering problems with high densities and dimensionalities. The linear trend lines clearly reveal the individual algorithms' steadiness in terms of increasing and decreasing performance pattern with respect to the average computation cost consumed by those automatic clustering algorithms based on DB and CS indices evaluations.

5.6 Experiment 5: Statistical analysis

In this experiment, the various clustering solution results obtained by the represented algorithms are further validated by using a nonparametric statistical analysis technique. The statistical analysis test carried out involve the use of Friedman's nonparametric test to draw a statistically meaningful conclusion to all the above performance claims on the different automatic data clustering algorithms proposed in this paper. Table 16 reports the computed Friedman's mean rank for both the four single and three hybrid algorithms tested across the forty-one datasets for twenty (20) replications. The mean rank results show that the FA is the best performing algorithm with a minimum rank of 1 and a maximum rank of 1.98. The

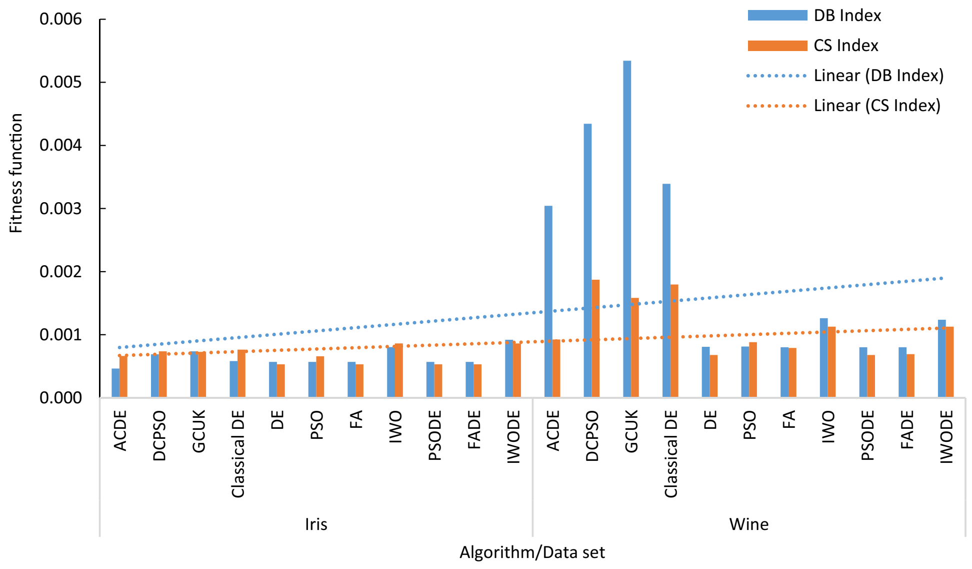


Fig. 14 Clustering performance for all algorithms on iris and wine dataset over 40 runs using DB and CS validity indices as fitness functions

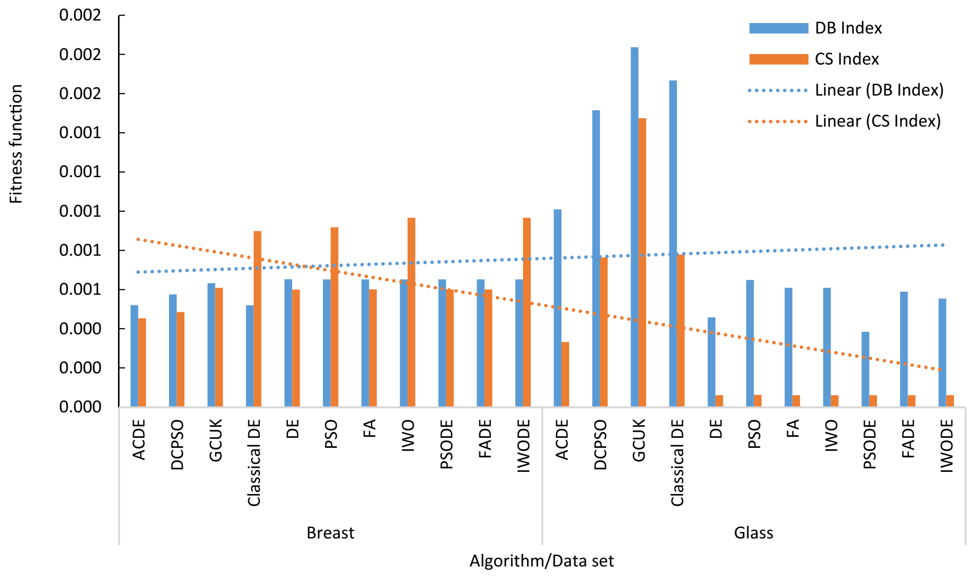


Fig. 15 Clustering performance for all algorithms on breast and glass dataset over 40 runs using DB and CS validity indices as fitness functions

DE also recorded a good performance when compared to PSO and IWO algorithms. For the hybrid algorithms, FADE appears to have outperformed PSODE and IWODE, but PSODE closely follows the performance FADE hybrid algorithm. Similarly, for the comparison of the representative algorithms on high dimension and classification datasets, the FA still emerged as the overall best performing algorithm with the best mean rank for each of the datasets as shown in Table 18 below.

To further verify specifically how each of the algorithm significantly differs, a post hoc test with Wilcoxon signed-rank tests was carried out on the Friedman test results for the representative algorithms clustering results. The results of the post hoc test are a set of p values that determine the statistical significant differences among the representative algorithms. After the initial p values were obtained for the Friedman's mean rank test, a new value of significance level α, which was initially set at 0.05, was computed using Bonferroni adjustment. We obtained $0.05/6 = 0.0083$ for the comparison between the single representative algorithms and $0.05/3 = 0.0167$ for the hybrid algorithms. This means that if the p value is larger than 0.0083, we do not have a statistically significant result. From Tables 16, 17, 18 and 19, the Friedman statistic with post hoc test clearly demonstrates that FA and its hybrid variant achieved significant performance improvements over other algorithms. The p-values in both Tables 17 and 19 clearly show that there are statistically

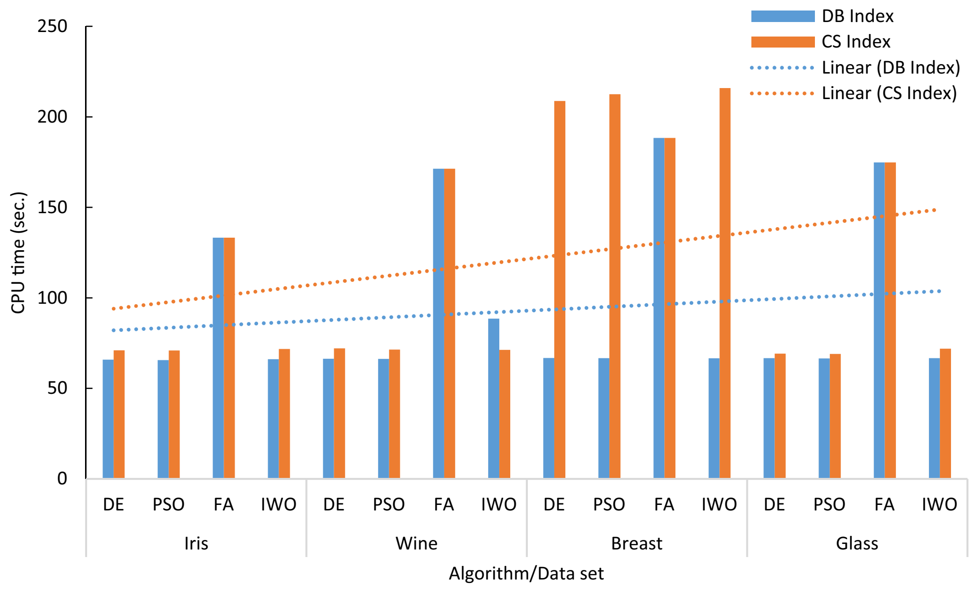


Fig. 16 Computational time for representative algorithm based on DB and CS validity indices

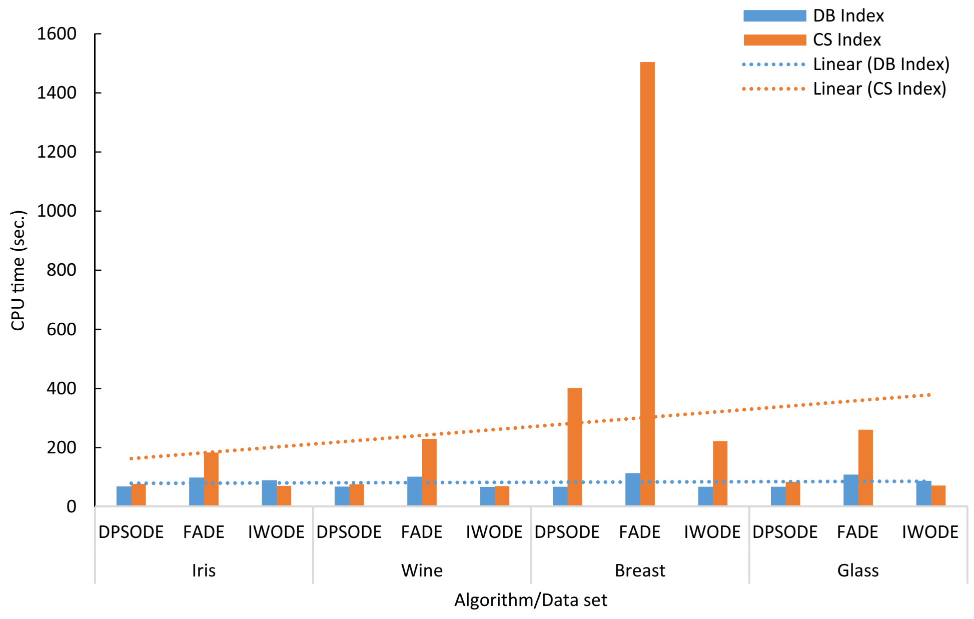


Fig. 17 Computational time for representative hybrid algorithm based on DB and CS validity indices

significant differences in the performance of the representative algorithms.

5.7 Experiment 6: Comparison with new generation metaheuristics

In this section we compare the clustering performance of eight different new generation algorithms based on number of function evaluation instead of number of iterations. By new generation algorithms, we mean those algorithms that were developed from the year 2000 upwards. The selected algorithms considered for these set of metaheuristics include, artificial bee colony (ABC), bees algorithm (BA), biogeography-based optimization (BBO), FA, harmony search (HS), IWO, symbiotic organisms search (SOS), and teaching–learning-based optimization (TLBO). The choice of these algorithms is based on their popularity, which is reflected on the number of related studies and citation impact extracted from Google Scholar. These data are reported in Figs. 18 and 19 Also from a different perspective, we aim to further compare the performance efficiency of the FA, which from the previous experimentation has proven to be a very effective algorithm for the automatic clustering problem with other new algorithms. However, considering the length of the current paper and page limit restriction, we refer interested readers to the following references for all the original algorithm design discussions, which were adopted and used to conduct the

Table 16 Friedman mean rank

	DE	PSO	FA	IWO	PSODE	FADE	IWODE
A1	1.95	3.53	1.63	2.9	1.28	2.18	2.55
A2	2.65	2.65	1.25	3.45	1.55	1.75	2.7
A3	2.85	2.55	1.2	3.4	1.7	1.65	2.65
Aggregation	3.6	3	1.3	2.1	1.95	1.8	2.25
Birch1	2.65	3	1.05	3.3	1.75	1.45	2.8
Birch2	2.05	3.75	1.15	3.05	1.48	1.68	2.85
Birch3	3.15	2.25	1.2	3.4	1.55	1.45	3
Breast	2.6	3.95	1.55	1.9	1.45	2.45	2.1
Bridge	2.15	3	1	3.85	1.9	1.1	3
Compound	2.9	3.9	1.53	1.68	2.25	2.05	1.7
D31	3.95	1.93	1.98	2.15	2.25	1.5	2.25
Dim002	3.3	1.8	1.65	3.25	1.4	1.9	2.7
Dim016	3	1.7	1.3	4	1.55	1.45	3
Dim032	2.95	2	1.05	4	1.6	1.4	3
Dim064	2.8	2.2	1	4	1.85	1.15	3
Dim128	2.6	2.4	1	4	1.8	1.2	3
Dim256	2.5	2.5	1	4	1.9	1.1	3
Dim512	2.4	2.6	1	4	1.95	1.05	3
Dim1024	2.1	2.9	1	4	2	1	3
Flame	3.3	3.3	1.45	1.95	2.55	1.7	1.75
Glass	3.4	3	1.1	2.5	1.9	1.95	2.15
Housec5	1.95	3.65	1.15	3.25	1.75	1.4	2.85
Housec8	2.2	3.4	1	3.4	1.85	1.15	3
Iris	2	3.1	1.05	3.85	1.55	1.45	3
Jain	3.15	2.55	1.53	2.78	1.88	1.75	2.38
Leaves	2.55	2.45	1	4	1.45	1.55	3
Letter	2.25	2.65	1.1	4	1.8	1.2	3
Joensuu	2.9	3.95	1.25	1.9	2.55	1.75	1.7
Finland	2.2	3.75	1.05	3	1.75	1.5	2.75
Pathbased	3.15	2.98	1.73	2.15	1.8	1.95	2.25
R15	3.75	2.3	1.75	2.2	2.2	2	1.8
S1	3.15	1.25	1.8	3.8	1.55	1.5	2.95
S2	2.9	1.7	1.55	3.85	1.4	1.75	2.85
S3	2.6	2.5	1.2	3.7	1.7	1.35	2.95
S4	2.8	2.35	1.35	3.5	1.4	1.65	2.95
Spiral	3.25	2.95	1.4	2.4	2.1	1.55	2.35
T4.8k	2.4	3	1.2	3.4	1.5	1.55	2.95
Thyroid	2.6	2.2	1.25	3.95	1.63	1.38	3
Wdbc	2.45	2.85	1.25	3.45	1.53	1.48	3
Wine	2.4	2.65	1	3.95	1.6	1.4	3
Yeast	3.35	3.4	1.55	1.7	2.4	1.9	1.7

clustering simulations that are reported in Table 20 below. Therefore, see ABC [73], HS [74], BBO [75], TLBO [76], BA [77], and SOS [78] for the original algorithm design and computation model explanations. The parameter configuration for the eight algorithms still remains the same as was used in the original algorithm design papers, except for the population size of 40 and 50,000 maximum number of function evaluation that were used for the current study simulation. The number of function evaluation in each algorithm is computed by taking into consideration the initial population size (*Pop_size*) and number of generation (***t***). For example, in the case of the FA, the number of function evaluation is obtained using the following expression: $NFE = \frac{Pop\_size \times (Pop_{size} - 1)}{2}$, where *NFE* denotes the number of function evaluation. The maximum number

Table 17 *p* values produced by the friedman rank-sum test for equal mediums

	DE-PSO	DE-FA	DE-IWO	PSO-FA	PSO-IWO	FA-IWO	PSODE-FADE	PSODE-IWODE	FADE-IWODE
A1	0	0.575	0	0.003	0.014	0.008	0.003	0	0.218
A2	0.526	0	0.003	0.001	0.014	0	0.478	0.001	0.002
A3	0.433	0	0.1	0	0.015	0	0.575	0	0.002
Aggregation	0.067	0	0	0.001	0.012	0.023	0.737	0.881	0.411
Birch1	0.37	0	0.015	0	0.204	0	0.433	0	0
Birch2	0	0.002	0	0	0	0.002	0.225	0	0.057
Birch3	0.023	0	0.351	0.004	0.001	0	0.279	0	0
Breast	0	0.279	0.005	0	0	0.823	0.003	0.079	0.028
Bridge	0.002	0	0	0	0	0	0.002	0	0
Compound	0	0.002	0.008	0	0	0.465	0.3	0.084	0.123
D31	0	0	0	0.171	0.218	0.502	0.093	0.709	0.145
Dim002	0.001	0	0.852	0.852	0.002	0	0.062	0	0.025
Dim016	0	0	0	0.048	0	0	0.852	0	0
Dim032	0	0	0	0	0	0	0.191	0	0
Dim064	0.001	0	0	0	0	0	0.025	0	0
Dim128	0.279	0	0	0	0	0	0.009	0	0
Dim256	0.526	0	0	0	0	0	0.002	0	0
Dim512	0.313	0	0	0	0	0	0	0	0
Dim1024	0.005	0	0	0	0	0	0	0	0
Flame	0.627	0	0.001	0	0.002	0.093	0.005	0.007	0.654
Glass	0.94	0	0.003	2	0.086	0	0.867	0.794	0.575
Housec5	0	0.001	0	0	0.021	0	0.218	0.073	0
Housec8	0	0	0	0	0.502	0	0.006	0	0
Iris	0	0.002	0	0	0	0	0.937	0	0
Jain	0.575	0	0.455	0.191	0.823	0.001	0.376	0.026	0.022
Leaves	0.279	0	0	0	0	0	0.97	0	0
Letter	0.086	0	0	0	0	0	0.008	0	0
Joensuu	0	0.01	0	0	0	0.067	0.198	0	0.494
Finland	0	0.002	0	0	0.003	0	0.575	0	0.218
Pathbased	0.296	0	0.002	0.002	0.008	0.023	0.687	0.445	0.872
R15	0.001	0	0	0.526	0.823	0.263	0.794	0.093	0.126
S1	0	0	0.004	0.012	0	0	0.911	0	0
S2	0.001	0	0	0.575	0	0	0.351	0	0
S3	0.97	0	0.001	0.017	0.001	0	0.232	0	0
S4	0.048	0	0.017	0.025	0.004	0	0.332	0	0
Spiral	0.794	0.001	0.019	0.004	0.025	0.014	0.093	0.627	0.04
T4.8k	0.01	0	0	0.002	0.01	0	0.655	0	0.002
Thyroid	0.794	0	0	0.004	0	0	0.171	0	0
Wdbc	0.025	0	0	0.003	0.025	0	0.317	0	0
Wine	0.191	0	0	0	0	0	0.351	0	0
Yeast	0.881	0	0	0.001	0.001	0.737	0.073	0.001	0.313

of function evaluation (*MNFE*) is obtained as follows: $MNFE = \frac{Pop\_size \times (Pop_{size} - 1)}{2} \times \boldsymbol{t}$.

For a fairer comparison of all the representative metaheuristic algorithms, the number of function evaluation was used to obtain optimal clustering results. This contrasts with the previous evaluation process reported in Sect. 5 (Experiment 1–4) in which the number of iterations was used as an estimate of the amount of work done by each individual algorithm. However, most of the existing clustering algorithms have been implemented and evaluated based on number of iterations used by the algorithms. Further, performance evaluation that is based on the candidate algorithm's number of function evaluation has frequently been preferred for performance analysis

Table 18 Friedman mean rank for high dimensional datasets

Algorithms	Dataset											
	A1	A2	A3	Birch 1	Birch 2	Birch 3	Bridge	D31	Housec5	Housec8	Leaves	Letter
DE	2.65	4.00	4.00	4.00	3.65	4.00	3.00	4.00	4.00	4.00	2.95	2.50
PSO	3.00	1.75	2.00	2.05	3.35	1.90	4.00	1.35	2.25	2.30	4.00	2.50
FA	1.60	1.25	1.05	1.00	1.00	1.10	1.00	1.65	1.05	1.10	1.00	1.00
IWO	2.75	3.00	2.95	2.95	2.00	3.00	2.00	3.00	2.70	2.60	2.05	4.00

Table 19 *p*-values produced by the Friedman rank-sum test for equal mediums on high dimensional datasets

Dataset	PSO-DE	FA-DE	IWO-DE	FA-PSO	IWO-PSO	IWO-FA
A1	0.478	0.007	0.709	0.002	0.601	0.006
A2	0.000	0.000	0.000	0.003	0.000	0.000
A3	0.000	0.000	0.000	0.000	0.001	0.000
Birch 1	0.000	0.000	0.000	0.000	0.000	0.000
Birch 2	0.048	0.000	0.000	0.000	0.000	0.000
Birch 3	0.000	0.000	0.000	0.000	0.000	0.000
Bridge	0.000	0.000	0.000	0.000	0.000	0.000
D31	0.000	0.000	0.000	0.048	0.000	0.000
Housec5	0.000	0.000	0.000	0.000	0.028	0.000
Housec8	0.000	0.000	0.000	0.000	0.025	0.000
Leaves	0.000	0.000	0.000	0.000	0.000	0.000
Letter	0.575	0.000	0.000	0.000	0.000	0.000

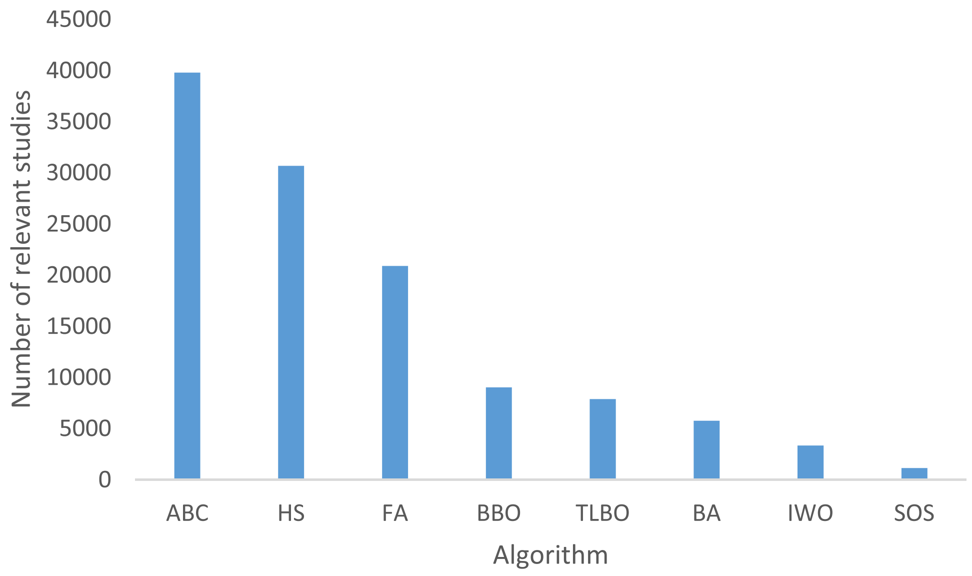


Fig. 18 Number of related published studies on the representative metaheuristics

over the traditional iterative method [51] because some ambiguity is associated with the later. For example, various algorithms associate different meanings and computational effort with iteration. More so, measures of merit that are based on the number of iterations have been discouraged because they present a biased and unfair comparison, which is most likely associated with the discrepancies in the number of functions evaluation consumed by the individual compared algorithm [51]. Therefore, in the current study, we investigate the quality of clustering capability of eight metaheuristic algorithms by employing the required number of function evaluations for acquiring near-optimal clustering solutions for the ten test problems presented in Table 20.

In the previous experiment and results analysis reported in Tables 5, 7 and 13, it was revealed that the FA algorithm

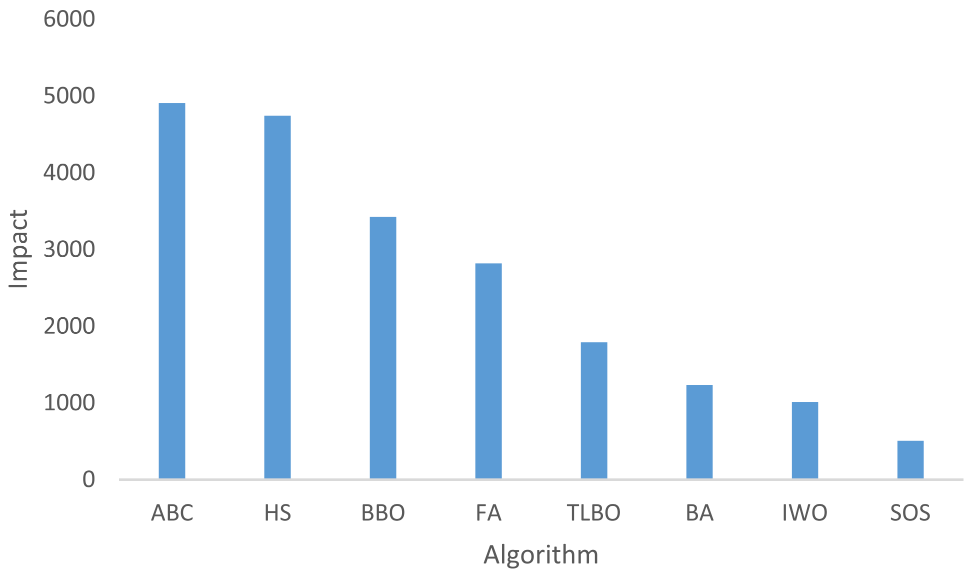


Fig. 19 Number of citation impact of representative metaheuristics

had outperformed the other algorithms. The FA also happens to be classified as a new generation metaheuristic algorithm, Therefore, to further validate its superior performance an additional experiment was conducted to compares eight new generation metaheuristics, including the FA. Ten datasets were selected from the existing forty-one test problems described in Table 4 above and used for the final set of experiments. To ascertain the global performance of any algorithm, five descriptive statistics, that is, average, median, standard deviation, best, and worst result values, can be used to analyze easily which of the algorithms is doing better or is more superior among the group. The numerical results of the experiment are presented in Table 20 and the results are discussed afterwards.

Table 20 orders the algorithms according to their final average, best and worst quality clustering results, with their respective computed median and standard deviation values recorded respectively. The computational results shown in the table reveal that the FA obtained global best clustering solutions in nine out of ten datasets for the DB validity index and six best clustering solutions in ten datasets for the CS validity index. Overall, the FA has the least average values for the two fitness values (DB and CS). Therefore, the FA algorithm still maintains its lead by outperforming its competitors. Overall, the simulation results show that even though the eight metaheuristic algorithms can be used to solve automatic clustering problems, the FA algorithm appears to be the best and most robust algorithm in terms of quality of clustering solution and computational time. For example, in Fig. 20, FA showed reasonable lower and balanced computational time for both DB and CS indices as compared to other algorithms. However, the HS algorithm consumed the least computational time among other algorithms, while BA and SOS algorithms had the worst computational cost, specifically for the CS validity index.

6 Conclusion and future direction

This paper has provided a systematic state-of-the-art review of nature inspired metaheuristic algorithms for automatic clustering. The paper also presented a comparative performance evaluation study of five metaheuristic algorithms and subsequently demonstrated the capability of the representative algorithms to solve automatic clustering problems. The respective population-based algorithms, namely GA, DE, PSO, FA, and IWO, were successfully implemented and used to automatically partition a given dataset without any prior information about the number of naturally occurring groups in the dataset. The results of the computational experiment using forty-one datasets show clearly that the FA algorithm outperformed its competitors, namely GA, DE, PSO, and IWO, in terms of quality of clustering solution obtained by the respective algorithms in most of the datasets. However, it was noticeable that the execution time recorded for the FA method was higher than for the other algorithms and therefore it performance was at the expense of computational cost. Hybridization methods was also considered to address the problem of high computational time incurred by the FA and by improving the FA convergence speed.

Three hybrid algorithms was implemented by combining DE with PSO, FA and IWO. Afterwards the new hybrid variants are employed to solve the same set of automatic data clustering problems akin to their single algorithm implementation. The numerical results of the chosen hybrid methods revealed that the hybrid FA yielded a

Table 20 Computational results for the five representative algorithms on twenty datasets

Dataset	Criteria	ABC		BA		BBO		FA		HS		IWO		SOS		TLBO	
		DB	CS	DB	CS	DB	CS	DB	CS	DB	CS	DB	CS	DB	CS	DB	CS
Aggregation	Average	0.6312	0.7676	0.6603	0.7279	0.8429	0.8456	**0.6086**	**0.5977**	0.7083	0.8207	0.7245	0.8251	0.6809	0.6727	0.6107	0.5930
	Median	0.6242	0.7583	0.6602	0.7286	0.8399	0.8439	0.6120	0.5734	0.7128	0.8172	0.7199	0.8345	0.6795	0.6761	0.6121	0.5912
	SD	0.0293	0.0660	0.0151	0.0395	0.0995	0.0646	0.0122	0.0595	0.0352	0.0410	0.0122	0.1172	0.0232	0.0098	0.0091	0.0245
	Best	0.5784	0.6357	0.6203	0.6239	0.6500	0.6723	0.6120	0.4383	0.6368	0.7360	0.7199	0.7352	0.6284	0.6593	0.5688	0.5231
	Worst	0.6954	0.9123	0.6876	0.8009	1.0914	0.9951	0.5663	0.7350	0.7696	0.9095	0.7565	1.1415	0.7222	0.6821	0.6270	0.6368
Breast	Average	0.9486	1.1155	1.0479	0.8426	1.4469	1.1185	**0.6519**	**0.6096**	0.7042	0.9661	0.6520	0.8055	1.0611	1.1510	0.6538	0.9821
	Median	0.9485	1.1274	1.0495	0.8485	1.4588	1.1439	0.6519	0.6025	0.6966	1.0672	0.6520	0.6862	1.0929	1.1770	0.6533	1.0237
	SD	0.0564	0.0636	0.0640	0.1187	0.1135	0.0768	0.0000	0.0259	0.0336	0.1760	0.0000	0.2171	0.1096	0.0563	0.0025	0.1060
	Best	0.7952	0.8821	0.8143	0.6862	1.1725	0.9401	0.6519	0.5996	0.6734	0.6025	0.6519	0.5996	0.9151	1.0570	0.6519	0.6862
	Worst	1.0460	1.2110	1.1714	1.0502	1.6650	1.2381	0.6519	0.6862	0.8441	1.0897	0.6521	1.1171	1.2527	1.1930	0.6669	1.0352
Compound	Average	0.5089	0.7150	0.5128	0.7179	0.6379	0.7695	**0.4932**	0.5758	0.5099	0.7084	**0.4932**	0.7673	0.5392	0.5527	**0.4932**	**0.5368**
	Median	0.5019	0.7566	0.5120	0.7180	0.6149	0.7732	0.4932	0.5032	0.5098	0.7189	0.4932	0.7732	0.5386	0.5245	0.4932	0.5032
	SD	0.0210	0.0749	0.0068	0.0000	0.0997	0.0265	0.0000	0.1097	0.0082	0.0705	0.0000	0.0169	0.0173	0.0576	0.0000	0.0593
	Best	0.4932	0.5032	0.4992	0.7180	0.5097	0.6647	0.4932	0.4961	0.4942	0.5113	0.4932	0.7180	0.5068	0.5113	0.4932	0.5032
	Worst	0.5917	0.8007	0.5263	0.7180	0.9709	0.8105	0.4932	0.7732	0.5281	0.7732	0.4932	0.7732	0.5831	0.6507	0.4932	0.7732
Flame	Average	0.7777	0.3994	0.7777	**0.3846**	0.8044	0.6245	0.7736	**0.3846**	0.7745	0.4011	0.7754	0.4228	**0.7520**	0.5153	0.7740	0.3868
	Median	0.7778	0.3846	0.7773	0.3846	0.7992	0.3986	0.7737	0.3846	0.7739	0.3846	0.7732	0.3875	0.7553	0.5491	0.7737	0.3846
	SD	0.0022	0.0519	0.0015	0.0000	0.0197	0.3122	0.0002	0.0000	0.0016	0.0517	0.0027	0.0986	0.0109	0.0740	0.0011	0.0036
	Best	0.7732	0.3846	0.7746	0.3846	0.7778	0.3846	0.7737	0.3846	0.7738	0.3846	0.7732	0.3846	0.7325	0.4116	0.7737	0.3846
	Worst	0.7816	0.7142	0.7812	0.3846	0.8487	1.4261	0.7732	0.3846	0.7792	0.7142	0.7788	0.7142	0.7708	0.5803	0.7788	0.3948
Glass	Average	0.8066	0.1140	0.8473	**0.0608**	1.1375	0.1481	**0.5207**	**0.0608**	0.9233	0.7569	0.5954	**0.0608**	0.8271	**0.0608**	0.6026	**0.0608**
	Median	0.8075	0.1147	0.8519	0.0608	1.1010	0.1586	0.6091	0.0608	0.9268	0.7487	0.6092	0.0608	0.8398	0.0608	0.6093	0.0608
	SD	0.0645	0.0488	0.0383	0.0000	0.1505	0.0467	0.1292	0.0000	0.0425	0.0303	0.0608	0.0000	0.0325	0.0000	0.0436	0.0000
	Best	0.6983	0.0608	0.7383	0.0608	0.9388	0.0608	0.6091	0.0608	0.8468	0.7188	0.3336	0.0608	0.7449	0.0608	0.3336	0.0608
	Worst	0.9310	0.2057	0.9034	0.0608	1.5131	0.2483	0.3336	0.0608	1.0075	0.8905	0.6092	0.0608	0.8863	0.0608	0.6105	0.0608
Iris	Average	0.6388	0.7850	0.6561	0.6056	0.8860	0.8225	0.5700	0.5359	0.5909	0.5837	0.9755	0.8582	0.6854	0.6321	0.5700	0.5366
	Median	0.6189	0.7859	0.6575	0.5993	0.8876	0.8135	**0.5700**	0.5375	0.5816	0.5821	0.9850	0.8488	0.6826	0.6312	**0.5700**	**0.5311**
	SD	0.0645	0.0733	0.0265	0.0333	0.1037	0.1261	0.0000	0.0046	0.0287	0.0402	0.0747	0.1032	0.0330	0.0185	0.0000	0.0297
	Best	0.5663	0.6273	0.5947	0.5375	0.6337	0.5968	0.5700	0.5311	0.5717	0.5311	0.8167	0.6599	0.6205	0.6037	0.5700	0.5311
	Worst	0.7853	0.9327	0.7121	0.6801	1.0886	1.0670	0.5700	0.5577	0.7404	0.7052	1.1598	1.0670	0.7820	0.6573	0.5700	0.7191
Leaves	Average	1.4293	0.8379	1.4710	**0.4920**	1.6312	1.0065	**0.5819**	**0.4920**	1.6556	1.0972	1.5484	0.8234	1.3918	0.5103	0.8139	**0.4920**
	Median	1.4509	0.8416	1.4791	0.4920	1.6324	1.0095	0.5825	0.4920	1.6599	1.0971	1.5532	0.8143	1.4040	0.5103	0.7762	0.4920
	SD	0.0838	0.1480	0.0447	0.0000	0.0698	0.1335	0.0045	0.0000	0.0624	0.1311	0.0572	0.1078	0.0381	0.0185	0.1191	0.0000
	Best	1.2375	0.4920	1.3302	0.4920	1.4544	0.7218	0.5951	0.4920	1.4772	0.8088	1.4059	0.5286	1.3210	0.4920	0.6600	0.4920
	Worst	1.5833	1.0916	1.5452	0.4920	1.7285	1.3876	0.5707	0.4920	1.7910	1.4307	1.6272	0.9884	1.4654	0.5286	1.1371	0.4920
Thyroid	Average	0.6477	0.6409	0.6630	0.6409	0.9819	0.6409	**0.4892**	0.6409	0.5042	0.6409	1.0558	0.6572	0.7485	**0.4375**	0.4968	0.6409
	Median	0.6345	0.6409	0.6688	0.6409	0.9667	0.6409	0.4813	0.6409	0.5043	0.6409	1.0435	0.6409	0.7438	0.4403	0.5011	0.6409

Table 20 (continued)

Dataset	Criteria	ABC		BA		BBO		FA		HS		IWO		SOS		TLBO	
		DB	CS	DB	CS	DB	CS	DB	CS	DB	CS	DB	CS	DB	CS	DB	CS
	SD	0.0466	0.0000	0.0352	0.0000	0.1030	0.0000	0.0098	0.0000	0.0036	0.0000	0.1066	0.0634	0.0422	0.0062	0.0175	0.0000
	Best	0.5811	0.6409	0.5743	0.6409	0.8048	0.6409	0.5011	0.6409	0.4913	0.6409	0.9001	0.6409	0.6328	0.4271	0.4814	0.6409
	Worst	0.7522	0.6409	0.7139	0.6409	1.3075	0.6409	0.4813	0.6409	0.5146	0.6409	1.2864	0.9872	0.8122	0.4423	0.5897	0.6409
Yeast	Average	1.0116	1.0998	1.0734	0.8744	1.2581	1.0941	**0.8027**	**0.8362**	1.1234	0.9320	1.2830	1.2314	1.1250	0.9121	0.8330	0.8474
	Median	1.0051	1.0957	1.0724	0.8829	1.2652	1.0303	0.8017	0.8829	1.1370	0.9096	1.2864	1.2004	1.1287	0.9355	0.8168	0.8829
	SD	0.0376	0.1580	0.0304	0.0229	0.0583	0.2070	0.0064	0.0787	0.0453	0.0663	0.0728	0.1900	0.0243	0.1082	0.0383	0.0656
	Best	0.9318	0.8829	1.0012	0.7759	1.1486	0.8829	0.8305	0.6570	1.0166	0.8213	1.0660	0.9337	1.0481	0.6864	0.8002	0.6570
	Worst	1.0760	1.3641	1.1281	0.8829	1.3922	1.6230	0.8000	0.8829	1.2091	1.1197	1.4619	1.6193	1.1538	1.0184	0.9681	0.8829
Wine	Average	1.0401	0.7070	1.0067	0.5205	1.4455	0.9008	**0.5269**	0.3766	1.1513	0.8529	0.5478	**0.3748**	0.9866	0.4648	0.6231	0.3859
	Median	1.0405	0.7042	1.0208	0.5205	1.4336	0.9173	0.5559	0.3786	1.1407	0.8412	0.5566	0.3731	0.9836	0.4726	0.6124	0.3793
	SD	0.1213	0.1255	0.0802	0.0000	0.1330	0.1967	0.0708	0.0335	0.0745	0.1616	0.0316	0.0254	0.0367	0.0373	0.1313	0.0398
	Best	0.7951	0.5054	0.7941	0.5205	1.2259	0.4947	0.6963	0.3047	1.0241	0.4863	0.4381	0.3047	0.9003	0.4061	0.4382	0.3105
	Worst	1.2187	0.9617	1.1752	0.5205	1.7247	1.3378	0.4379	0.4386	1.3299	1.1736	0.5573	0.4117	1.0457	0.5078	0.8276	0.4558

All text in bold represents the best results obtained by the individual representative algorithms

better performance result in terms of computation cost, with an improved convergence speed, while the hybrid PSODE and IWODE outperformed their single algorithms variants by yielding better quality of clustering solutions. Furthermore, statistical analysis test was conducted to validate the numerical results obtained by the representative algorithms and the results showed that there were statistically significance differences in the performances of the algorithms, with FA outperforming the other methods. Notably, it was observed that the FA improved its convergence speed with the hybridization technique.

For a fairer comparison, the number of function evaluation was adopted and used as a performance metric over the traditional iterative method of evaluation. Different descriptive statistical methods were used to analysis the results of the experiment based on the number function evaluation test. Overall the performance of eight well-known new generation metaheuristic algorithms were compared. The results of the experimental investigation revealed that most of the tested algorithms performed fairly well, however, the FA still showed superior performance over all the other methods. It was also obvious that, except for the BA and SOS algorithms, the remaining six algorithms possessed good qualities of rapid convergence and high stability of results.

Overall, the summarized comparison results revealed that all the representative clustering algorithms considered in this study can effectively determine the most appropriate number of clusters and subsequently provide good clustering partitions. While it has been frequently emphasized in this paper that the FA and its hybrid clustering methods yielded clustering solutions of superior quality to those form other algorithms, it is, nevertheless, equally important to note that DE, PSO, FA and IWO algorithms achieved good clustering performances across the forty-one datasets, which comprised both low and higher dimensional datasets. In addition, the representative algorithms are robust for automatic data clustering problems and easy to implement. However, the first experiment that included the use of GA reported in Table 5, revealed that the GA is not a very stable algorithm for the task of automatic clustering.

In future research, the representative algorithms can also be hybridized with other efficient local search techniques and probably utilized for many different areas of applications. It would also be interesting to consider implementing the current clustering methods using a different clustering validity indexes, besides the DB and CS validity indices that were used for the current study performance analysis. Another interesting future research focus would be to critically analyze the computational time complexities of all the algorithms considered in this paper, in

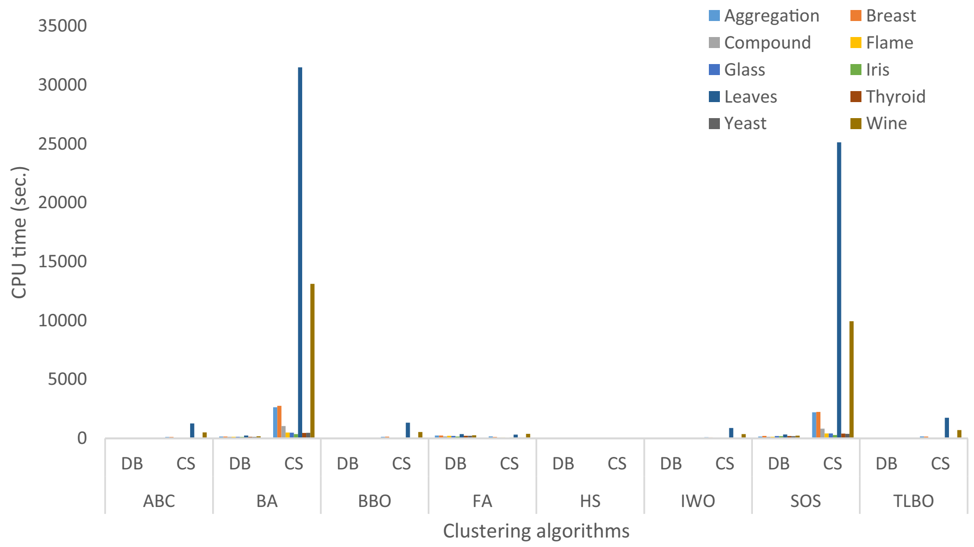


Fig. 20 CPU time consumed by algorithms

terms of their individual capabilities to solve automatic clustering problem.

Compliance with ethical standards

Conflict of interest The author declare that there is no conflict of interest regarding the publication of the paper.

Appendix 1: Representative algorithmic design concepts

Genetic algorithm

Genetic algorithms belong to a class of evolutionary algorithms that are based on an abstraction of Darwin's theory of biological evolution; they were first developed by John Holland [65]. The main characteristics of the GA are three genetic operators, namely crossover, mutation, and selection [79]. Each solution in a population is typically encoded in a binary or real string called a chromosome. The crossover of two strings will produce different offspring by exchanging the genes of the chromosomes. Mutation is performed by flipping some digits in a string which produces new offspring (or solutions). After crossover and mutation, new solutions are generated, and their quality is evaluated based on a fitness function. The fitness function is typically connected to the objective function of the problem to be solved. Based on the fitness of each solution, new solutions are selected and moved to the next generation. The algorithm continues generating new solutions through the crossover, mutation, and selection operators until a specified threshold is reached. The idea of selection is to choose the best chromosome from each generation and pass their genes to the next generation, while the idea of crossover is to combine the best chromosomes in the population and move their genes to the next generation. The idea of mutation is to alter some genes in each chromosome to ensure diversity in each generation. In general, on the one hand, crossover occurs most frequently with a probability range of between 0.6 and 0.95, while, on the other hand, the mutation rate is often lower, ranging from 0.001 to 0.05. GA's quality of convergence is usually aided by its ability to use the crossover to exploit and enhance the important characteristics in the population, which allows for better exploitation of the solution search space. Furthermore, the GA uses the mutation operator to enhance diversity in the search space through exploration by allowing the population to explore the search space better [58]. GA pseudocode is shown in Algorithm listing 1.

Algorithm 1: Genetic Algorithm

```
1: Begin GA
2: Generate initial population randomly, each solution encodes randomly selected $K$ clusters.
3: Compute fitness of generated population based on $DB(K)$ and $CS(K)$ validity indices using equations (3) and (7).
4:        Keep the global best solution
5:        do
6:                Selection
7:                Crossover
8:                Mutation
9:                Evaluate the quality of the new solutions
10:               Update the global best solution
11:       while (termination condition is not met)
12:       return global best solution
13: End GA
14: Output solution with the best minimum objective function
```

Differential evolution

Differential evolution was developed by Storn and Price in [56, 67], respectively. It is a derivative-free algorithm with a self-organizing tendency. Similar to pattern search and GA, DE is a population-based evolutionary algorithm that utilize three operators: selection, crossover and mutation. In fact, DE can be considered as an improved version of GA with explicit updating information, which gives it the capacity to perform theoretical analysis [58]. In DE, encoding and decoding of information is not required because it uses real numbers. Furthermore, in DE, unlike GA, each evolution is performed over each component of a chromosome, and virtually everything is performed in vectors. In mutation, a difference vector of two selected vectors (chosen randomly) is used to perturb an existing vector. This perturbation is more efficient because it is performed over each population vector. Besides, in crossover, chromosomes are exchanged component-wise. Apart from the mutation and crossover operators, DE has explicit updating equations. DE uses the following properties to generate new solutions at different iterations: target vector, trail vector and mutant vector. The target vector carries the solution to the optimization problem, whereas the trail vector is the resultant vector that is produced after the crossover operation has been performed between the target vector and the mutant vector. DE relies on mutation to obtain better solutions. Moreover, it uses the selection operator to drive the search towards regions that have improved solutions [3]. During the iteration $\varphi + 1$, for each vector $X_{i,\varphi} = \left(x_{i1,\varphi}, x_{i2,\varphi}, \dots, x_{in,\varphi}\right), i \in (1, 2, \dots, Pop\_size)$, a mutation vector is computed for every target vectors in the population using Eq. (16) [67]:

$$\mathcal{V}_{i,\varphi+1} = X_{r1,\varphi} + F\left(X_{r2,\varphi} - X_{r3,\varphi}\right) \tag{16}$$

where $r1, r2, r3 \in (1, 2, \dots, Pop\_size)$ are randomly chosen integers, the variable F denotes a scaling vector, between 0 and 1, X_{r1}, X_{r3}, X_{r2} are solution vectors chosen randomly. They satisfy the following conditions:

$$X_{r1}, X_{r3}, X_{r2} | r_1 \neq r_2 \neq r_3 \neq i \tag{17}$$

where i refers to the index of the solution in the present iteration.

In addition, during crossover, a trail vector is generated by merging the parent vector with a mutated vector [56], according to Eq. (18).

$$u_{i,\varphi+1} = \begin{cases} v_{i,\varphi+1} & if\ rand_j \leq CR \\ x_{i,\varphi} & if\ rand_j > CR \end{cases} \tag{18}$$

where CR refers to a constant value for crossover, $rand_j$ is a uniform random number in the interval [0,1], and j refers to each number randomly generated in the resulting array. The pseudocode of the modified DE clustering algorithm is shown in Algorithm listing 2.

Algorithm 2: Differential Evolution

```
1: Begin DE
2: Randomly generate initial solution, each solution encodes randomly selected $K$ clusters.
3: Evaluate quality of generated solution using $DB(K)$ and $CS(K)$ validity indices according to equations (3) and
   (7).
4:   Keep the global best solution
5:   While (termination condition is not met)
6:     For Each $x_i$ in the solution space
7:       Produce new solution $s_i$ using DE mutation, crossover and selection operators
8:       If Quality ($s_i$) >= Quality ($x_i$)
9:         Keep $s_i$ in the solution space
10:      Else
11:        Keep $x_i$ in solution space
12:      End If
13:    End
14:    Test the quality of the new solutions
15:    Update the global best solution
16:  End While
17:  Return the global best solution
18: End DE
19: Output solution with the best minimum objective function
```

Particle swarm optimization

Particle swarm optimization was developed by Kennedy and Eberhert [66]. It is inspired by the flocking behaviour of social organisms that move in groups, such as birds, ant colonies and school of fish. The algorithm mimics the interaction between members of a colony to share information and so it is a typical example of a swarm intelligence algorithm. The PSO can be combined with other algorithms for solving optimization problems of feature selection or parameter optimization, and so on. It uses particles (also known as, agents) to search for best solutions. The trajectory of particles is adjusted by a deterministic and stochastic component. Each particle in the population has the tendency to move randomly, and they are controlled by their best solution and their group best solution. A particle consists of a position vector $P_i = \left(p_{i1}, p_{i2}, \dots, p_{in}\right)$, and a velocity vector $V_i = \left(v_{i1}, v_{i2}, \dots, v_{in}\right)$. At every iteration, the position of each particle is changed according to Eqs. (19) and (20) [80, 81].

$$V_{id}^{t+1} = V_{id}^{t} + \tau_1 * rand(0,1) * \left(P_{id}^{t} - X_{id}^{t}\right) + \tau_2 * rand(0,1) * \left(P_{gd}^{t} - X_{id}^{t}\right) \tag{19}$$

$$X_{id}^{t+1} = X_{id}^{t} + V_{id}^{t+1} \tag{20}$$

where X_{id}^{t} and V_{id}^{t} refers to the position and velocity of each particle in the solution search space. The parameter d refers to the dimension of the problem, i represent the index of each particle and t represents the number of iterations. In addition, τ_1 and τ_2 are learning factors that guide the speed of each particle when moving towards the global optimum. The parameter P_i represents the current best position of the i^{th} particle and P_g represents the current best position identified by the neighbours of P_i. The function $rand(0,1)$ refers to randomly generated numbers, and P_{id}^{t} represents the current best position of the i^{th} particle at t iteration, and P_{gd}^{t} represents the global best particle of the entire population. Interested readers are referred to [82] for the basic PSO algorithm, while the pseudocode for the modified PSO clustering algorithm is shown in Algorithm listing 3.

Algorithm 3: Particle Swarm Optimization

```
1: Begin PSO
2: Generate initial population of particles (solution), each solution encodes randomly selected $K$ clusters.
3: Evaluate quality of generated solution using $DB(K)$ and $CS(K)$ validity indices according to equations (3) and (7).
4:   Keep the global best solution, $g_{best}$
5:   While (termination condition is not reached)
6:      For Each particle $x$ with position $p_i$
7:            Compute the fitness value of particle $x$
8:            If the fitness of $x$ is greater than the current best solution ($p_{best}$)
9:               Set the fitness of $x$ to be $p_{best}$
10:           End If
11:     End For
12:     Select the overall best particle and set it as $g_{best}$
13:     For Each particle $x$
14:           Compute the velocity of particle $x$
15:           Update the position of particle $x$
16:     End For
17:   End While
18: End PSO
19: Output the particle $i$ with best minimum objective function
```

Firefly algorithm

The firefly algorithm is a population-based swarm intelligence optimization algorithm that mimics the flashing light behaviour of fireflies. The FA was developed by Yang [68]. There are diverse species of fireflies and most of them generate quick flashes of light (flashlights) at consistent intervals. These flashlights are produced to lure other fireflies and to send cautionary warnings to potential targets. FA is typically used to handle difficult NP-hard problems [32, 83]. The light intensity of a flashlight decreases for increase in distance, as shown in Eq. (23). This is because light is released into the atmosphere, and is dissipated with the square of the distance.

$$I \propto {}^{1}/_{r^2} \tag{21}$$

Flashlight is typically expressed in such a way that it is proportional to the fitness function to be optimized. FA was designed based on following rules [83]:

1. All the species of firefly belong to the same sex.
2. The attractiveness value of a firefly is related to its light intensity, implying that fireflies with low flashlight will be attracted to fireflies with high flashlight.
3. The flashlight intensity of a firefly is controlled by the landscape of the fitness function to be optimized.

Light intensity and attractiveness are two important issues that should be carefully considered when using FA. The light intensity produced at a point, changes based on the distance and the brightness of light released into the atmosphere, as shown in Eq. (23). Generally, the light intensity I produced at a specific point y is directly proportional to the fitness value produced by the objective function, as expressed in Eq. (22).

$$I(y) \propto F(y) \tag{22}$$

$$I(r) = I_0 e^{-\gamma r^2} \tag{23}$$

where I_0 represents the initial light intensity when $r = 0, \gamma$ refers to the light absorption coefficient, and r represents distance. As revealed in Eq. (23), by merging the result of the inverse square law and absorption, the singularity at $r=0$ is evaded in the expression $1/r^2$, as discussed in [83]. Moreover, the singularity is also circumvented by approximating it in Gaussian form, as shown in Eq. (24). Besides, the attractiveness of a firefly (β) is related to the light intensity of the firefly, as shown in Eq. (24).

$$\beta(r) = \beta_0 e^{-\gamma r^2} \tag{24}$$

where β_0 represent the attractiveness when $r=0$.

The Euclidian distance between two fireflies x_i and x_j is computed using Eq. (25).

$$r_{ij} = \left\| x_i - x_j \right\| = \sqrt{\sum_{k=1}^{d} \left(x_{i,k} - x_{j,k} \right)^2} \tag{25}$$

where d represent the dimension of the problem. The movement of a firefly from position i to position j is given in Eq. (26):

$$x_i = x_i + \beta_0 e^{-\gamma r_{ij}^2}\left(x_j - x_i\right) + \alpha\epsilon_i \tag{26}$$

where $\alpha \in [0, 1]$, $\gamma \in [0, \infty)$, ϵ_i is a random number from a Gaussian distribution and ϵ_i can be substituted with *rand* − 0.5, where *rand* ∈ [0, 1]. The term ($\alpha\epsilon_i$) in Eq. (26) reveals the movement pattern of a firefly from one position to another, regarding their attractiveness. The pseudocode for the modified firefly clustering algorithm is shown in Algorithm listing 4.

Algorithm 4: Firefly Algorithm

```
1: Begin FA
2: Define the initial parameters, including $\beta_o$, $\alpha$, and $\gamma$
3: Define the objective function $f(x_i)$, $x_i = (x_{i1}, x_{i2} \dots x_{ik})^t$, with an initial population of $N$ fireflies (solutions)
   within $k$ dimensional search space $x_{ik}$, $i = 1, 2 \dots, N, k = 1,2,\dots,K$. Each solution encodes randomly selected $K$
   number of clusters.
4: Evaluate the $DB(K)$ and $CS(K)$ validity indices for each solution according to equations (3) and (7).
5: Determine the fitness function of each firefly to get their various light intensities ($L_i$)
6: While (termination condition is not reached)
7:     For i = 1 to number_of_firefles
8:         For j = 1 to number_of_firefles
9:             If ($L_i < L_j$)
10:                Move firefly i towards firefly j
11:            End If
12:            Compute the attractiveness variance with distance r using $exp(-\gamma r)$
13:            Compute the fitness values for all the fireflies
14:            Update the light intensity for each firefly $L_i$
15:        End For
16:    End For
17: End While
18: End FA
17: Output the firefly i with the minim $f(x_i)$
```

Invasive weed optimization

Weeds are plants that grows in unwanted territories. Although, the growth of colonizing weeds poses a threat to cultivated plants, weeds are quite robust and adaptive to change in the environment. Their colonizing behaviour, robustness and adaptive capacity inspired the design of the invasive weed optimisation (IWO) algorithm [69]. The algorithm is divided into four stages: initialization, reproduction, spatial dispersal, and competitive exclusion. It starts by initializing a population, which is achieved by generating a specific number of seeds and randomly distributing them to a search area (initialization). Furthermore, the dispersed seeds grow to become flowering plants which then reproduce by producing seeds, depending on their fitness value (reproduction). Furthermore, the reproductive seeds are then dispersed over the search area, and they in turn grow to become new plants (spatial dispersal). The process continues until the specified number of plants is reached. At this point, the competitive plants (plants with low fitness) are kept and the others are excluded from the population (competitive exclusion).

The exclusion process is carried out using the mechanism outlined below.

- All the weeds in the population can reproduce seeds according to their fitness value.
- Furthermore, the reproduced seeds are randomly spread to different locations in the solution space. The dispersed seeds then grow to become new plants.
- The fitness of each new plant is compared to the fitness of their respective parents. Plants with high fitness are retained in the population and allowed to reproduce, while plants with low fitness are removed. The formulation for weeds producing seeds is given in Eq. (27).

$$weed_n = \frac{f - f_{min}}{f_{max} - f_{min}}\left(w_{max} - w_{min}\right) + w_{min} \tag{27}$$

where f is the current weed's fitness, the variables f_{max} and f_{min}, respectively, represent the maximum and the minimum fitness of the current population, while w_{max} and w_{min}, respectively, represent the maximum and the least value of a weed.

The exclusion mechanism allows plants of low fitness value to reproduce if they contain better information than plants of higher fitness value. The pseudocode for the modified IWO clustering algorithm is given below in Algorithm listing 5.

Note that randomness and adaptation in IWO is provided by the spatial dispersion techniques employed by weeds for their dynamic reproduction. More so, the generated seeds are being randomly distributed over the *d* dimensional search space by normally distributed random numbers with mean equal to zero, but with varying variance. The process is computed by the IWO algorithm using Eq. (28).

$$\sigma_{iter} = \frac{\left(iter_{max} - iter\right)^n}{\left(iter_{max}\right)^n}\left(\sigma_{max} - \sigma_{min}\right) + \sigma_{min} \tag{28}$$

where $iter_{max}$ is the maximum number of iterations, σ_{iter} is the standard deviation at the present time step and n is the nonlinear modulation index [69].

Appendix 2: Clustering result of algorithms on selected dataset

See Figs. 21, 22, 23 and 24.

Algorithm 5: Invasive Weed Optimisation Algorithm

```
1: Begin IWO
2: Define the algorithm parameters, including highest number of weeds $h$ and lowest number of seeds $l$
3: Initialize population of weed matrix consisting of $N$ solutions, each solution encodes randomly selected $K$ clusters.
4: Evaluate the $DB(K)$ and $CS(K)$ validity indices for each solution according to equations (3) and (7).
5: Estimate the fitness function for each plant (solution) in the initial population
6:      Keep the global best plant
7:      While (maximum number of plants is not reached)
8:          Disperse some seeds over a solution space
9:          Evaluate fitness of each plant in the solution space
10:         Perform reproduction using the following: fitness value of the solution, $l$ and $h$
11:         Randomly spread the reproduced plants over the search space using normal distribution
12:         If (Total number of seeds is reached)
13:             Maintain the maximum number of seeds by computing N weeds for elimination
14:             Each weed can reproduce new seeds (or new solution)
15:             Test the fitness of the reproduced solution
16:             Compare the fitness of the reproduced seeds to the fitness of their parent seed
17:             Remove N weeds from the population. That is, weeds with low fitness value
18:         End If
19:         Update the global best plant
20:     End While
```

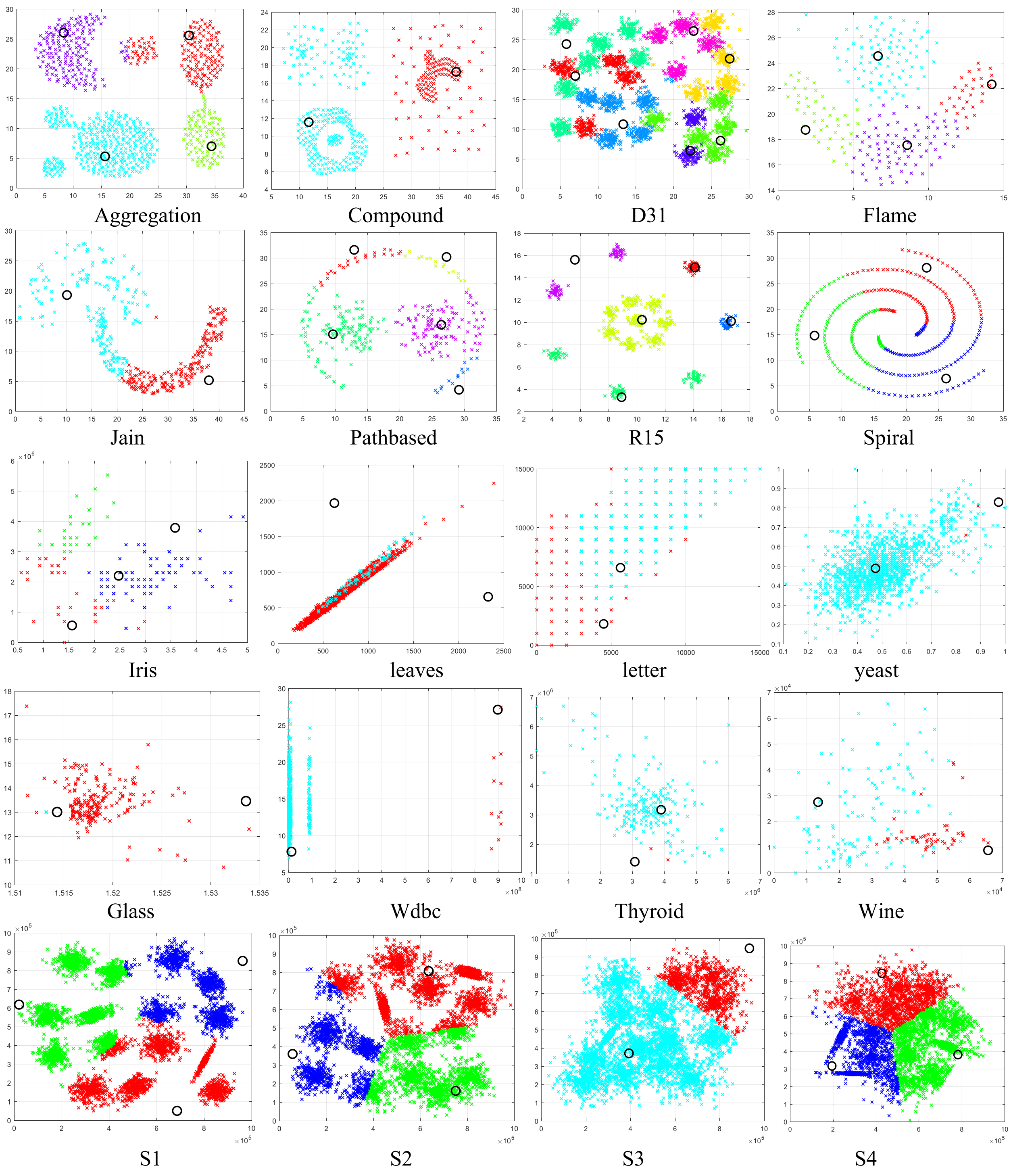


Fig. 21 Two-dimensional clustering plots obtained using DE algorithm

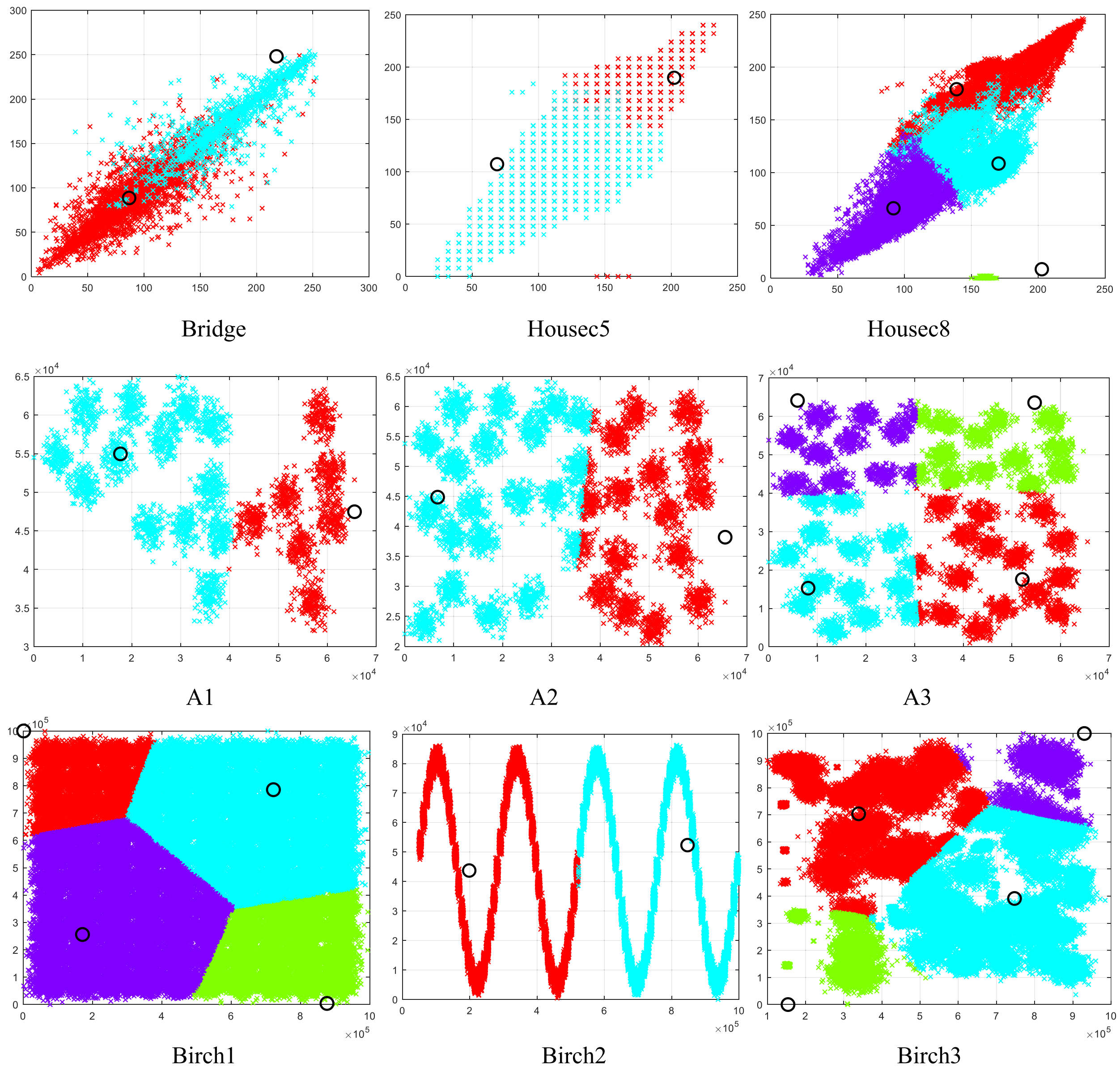


Fig. 21 (continued)

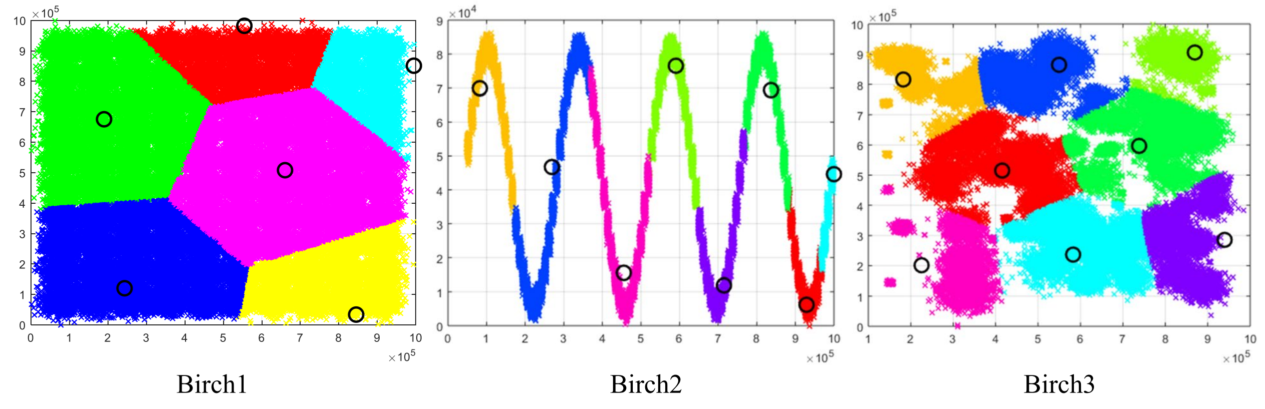


Fig. 22 Two-dimensional clustering plots obtained using PSO algorithm for 150 population size

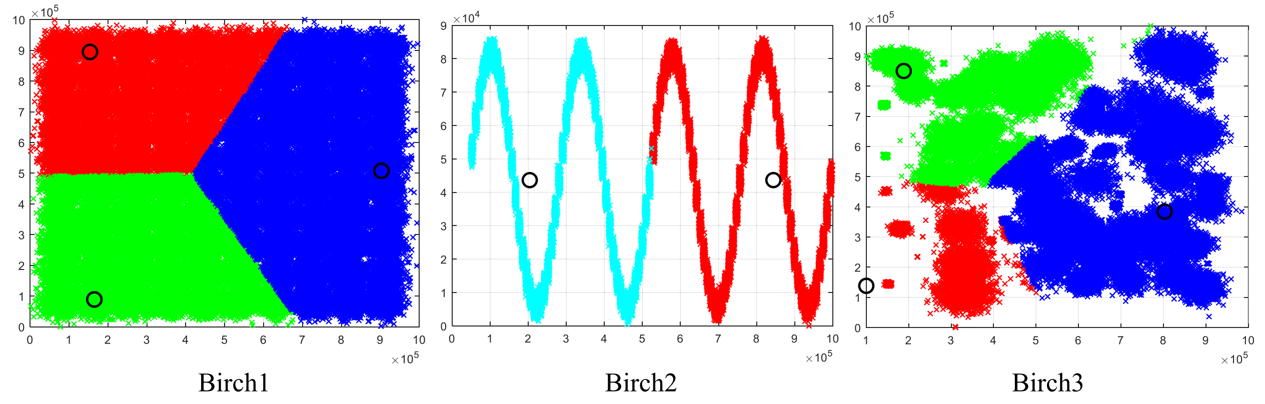


Fig. 23 Two-dimensional plot for Birch-sets obtained using FA algorithm for 150 population size

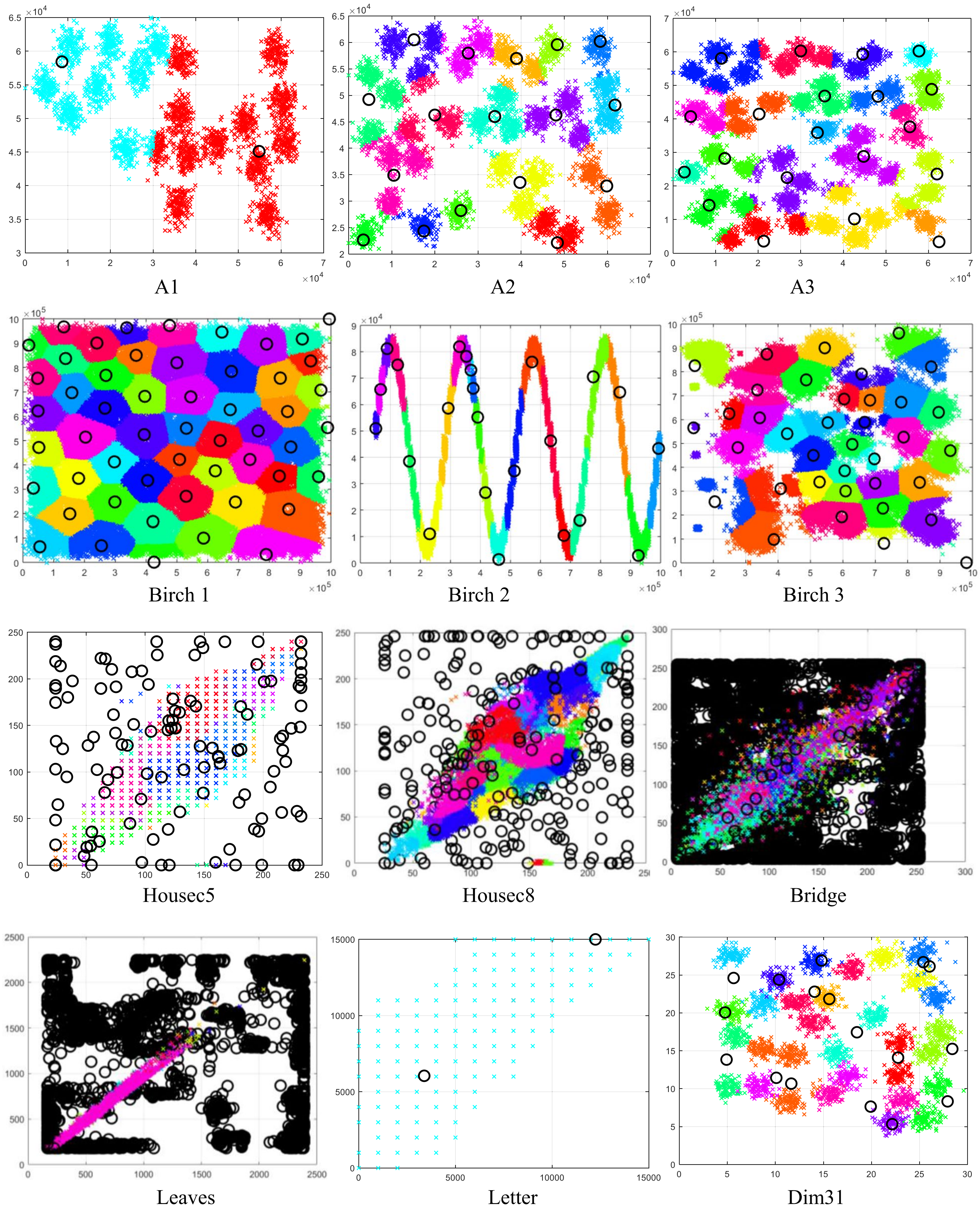


Fig. 24 Two-dimensional plots obtained by FA algorithm using high density data set with high classification features

References

1. Jain AK, Murty MN, Flynn PJ (1999) Data clustering: a review. ACM Comput Surv (CSUR) 31(3):264–323
2. Roberts SJ (1997) Parametric and non-parametric unsupervised cluster analysis. Pattern Recognit 30(2):261–272
3. Gan G, Ma C, Wu J (2007) Data clustering: theory, algorithms, and applications, vol 20. Siam, Philadelphia
4. Madhulatha TS (2012) An overview on clustering methods. arXiv preprint arXiv:1205.1117
5. Pearson K (1894) Contributions to the mathematical theory of evolution. Philos Trans R Soc Lond A 185:71–110
6. Jain AK, Dubes RC (1988) Algorithms for clustering data, vol 6. Prentice Hall, Englewood Cliffs
7. Rokach L, Maimon O (2005) Clustering methods. In: Data mining and knowledge discovery handbook. Springer, Boston, MA, pp 321–352. http://www.ise.bgu.ac.il/faculty/liorr/hbchap15.pdf. Accessed 20 Sept 2019
8. Aliniya Z, Mirroshandel SA (2019) A novel combinatorial merge-split approach for automatic clustering using imperialist competitive algorithm. Expert Syst Appl 117:243–266
9. Agrawal R, Gehrke J, Gunopulos D, Raghavan P (2005) Automatic subspace clustering of high dimensional data. Data Min Knowl Disc 11(1):5–33
10. José-García A, Gómez-Flores W (2016) Automatic clustering using nature-in-spired metaheuristics: a survey. Appl Soft Comput 41:192–213
11. He H, Tan Y (2012) A two-stage genetic algorithm for automatic clustering. Neurocomputing 81:49–59
12. Doval D, Mancoridis S, Mitchell BS (1999) Automatic clustering of software systems using a genetic algorithm. In: STEP'99. Proceedings ninth international workshop software technology and engineering practice. IEEE, pp 73–81
13. Das S, Abraham A, Konar A (2008) Automatic clustering using an improved differential evolution algorithm. IEEE Trans Syst Man Cybern Part A Syst Hum 38(1):218–237
14. Izakian Z, Mesgari MS, Abraham A (2016) Automated clustering of trajectory data using a particle swarm optimization. Comput Environ Urban Syst 55:55–65
15. Das S, Abraham A, Konar A (2008) Automatic kernel clustering with a multi-elitist particle swarm optimization algorithm. Pattern Recognit Lett 29(5):688–699
16. Kumar V, Chhabra JK, Kumar D (2014) Automatic cluster evolution using gravitational search algorithm and its application on image segmentation. Eng Appl Artif Intell 29:93–103
17. Zhou Y, Wu H, Luo Q, Abdel-Baset M (2019) Automatic data clustering using nature-inspired symbiotic organism search algorithm. Knowl Based Syst 163:546–557
18. Kuo RJ, Huang YD, Lin CC, Wu YH, Zulvia FE (2014) Automatic kernel clustering with bee colony optimization algorithm. Inf Sci 283:107–122
19. Su ZG, Wang PH, Shen J, Li YG, Zhang YF, Hu EJ (2012) Automatic fuzzy partitioning approach using variable string length artificial bee colony (VABC) algorithm. Appl Soft Comput 12(11):3421–3441
20. Chowdhury A, Bose S, Das S (2011) Automatic clustering based on invasive weed optimization algorithm. In: International conference on swarm, evolutionary, and memetic computing. Springer, Berlin, pp 105–112
21. Das S, Chowdhury A, Abraham A (2009) A bacterial evolutionary algorithm for automatic data clustering. In: 2009 IEEE congress on evolutionary computation. IEEE, pp 2403–2410
22. Jaramillo JH, Bhadury J, Batta R (2002) On the use of genetic algorithms to solve location problems. Comput Oper Res 29(6):761–779
23. Chen CL, Vempati VS, Aljaber N (1995) An application of genetic algorithms for flow shop problems. Eur J Oper Res 80(2):389–396
24. Neri F, Tirronen V (2010) Recent advances in differential evolution: a survey and experimental analysis. Artif Intell Rev 33(1–2):61–106
25. Yildiz AR (2013) A new hybrid differential evolution algorithm for the selection of optimal machining parameters in milling operations. Appl Soft Comput 13(3):1561–1566
26. Gong W, Cai Z, Zhu L (2009) An efficient multiobjective differential evolution algorithm for engineering design. Struct Multidiscip Optim 38(2):137–157
27. Paterlini S, Krink T (2006) Differential evolution and particle swarm optimisation in partitional clustering. Comput Stat Data Anal 50(5):1220–1247
28. Suresh K, Kundu D, Ghosh S, Das S, Abraham A (2009) Data clustering using multi-objective differential evolution algorithms. Fundamenta Informaticae 97(4):381–403
29. Van der Merwe DW, Engelbrecht AP (2003) Data clustering using particle swarm optimization. In: The 2003 congress on evolutionary computation, 2003. CEC'03, vol 1. IEEE, pp 215–220
30. Rana S, Jasola S, Kumar R (2011) A review on particle swarm optimization algorithms and their applications to data clustering. Artif Intell Rev 35(3):211–222
31. Gandomi AH, Yang XS, Alavi AH (2011) Mixed variable structural optimization using firefly algorithm. Comput Struct 89(23–24):2325–2336
32. Fister I, Fister I Jr, Yang XS, Brest J (2013) A comprehensive review of firefly algorithms. Swarm Evol Comput 13:34–46
33. Senthilnath J, Omkar SN, Mani V (2011) Clustering using firefly algorithm: performance study. Swarm Evol Comput 1(3):164–171
34. Ezugwu AE, Akutsah F (2018) An improved firefly algorithm for the unrelated parallel machines scheduling problem with sequence-dependent setup times. IEEE Access 6:54459–54478
35. Zhou Y, Luo Q, Chen H, He A, Wu J (2015) A discrete invasive weed optimization algorithm for solving traveling salesman problem. Neurocomputing 151:1227–1236
36. Mallahzadeh ARR, Oraizi H, Davoodi-Rad Z (2008) Application of the invasive weed optimization technique for antenna configurations. Prog Electromagn Res 79:137–150
37. Chowdhury Aritra, Das Swagatam (2012) Automatic shape independent clustering inspired by ant dynamics. Swarm Evol Comput 3:33–45
38. Bandyopadhyay S, Maulik U (2002) Genetic clustering for automatic evolution of clusters and application to image classification. Pattern Recognit 35(6):1197–1208
39. Omran M, Salman A, Engelbrecht A (2005) Dynamic clustering using particle swarm optimization with application in unsupervised image classification. In: Fifth world enformatika conference (ICCI 2005), Prague, Czech Republic, pp 199–204
40. Das Swagatam, Konar Amit (2009) Automatic image pixel clustering with an improved differential evolution. Appl Soft Comput 9(1):226–236
41. Liu R, Zhu B, Bian R, Ma Y, Jiao L (2015) Dynamic local search based immune automatic clustering algorithm and its applications. Appl Soft Comput 27:250–268
42. Bandyopadhyay S, Maulik U (2001) Nonparametric genetic clustering: comparison of validity indices. IEEE Trans Syst Man Cybern Part C Appl Rev 31:120–125

43. Bandyopadhyay S, Maulik U (2002) Genetic clustering for automatic evolution of clusters and application to image classification. Pattern Recognit 35:1197–1208
44. Lai CC (2005) A novel clustering approach using hierarchical genetic algorithms. Intell Autom Soft Comput 11:143–153
45. Lin HJ, Yang FW, Kao YT (2005) An efficient GA-based clustering technique. Tamkang J Sci Eng 8:113–122
46. Kundu D, Suresh K, Ghosh S, Das S, Abraham A, Badr Y (2009) Automatic clustering using a synergy of genetic algorithm and multi-objective differential evolution. In: International conference on hybrid artificial intelligence systems. Springer, Berlin, pp 177–186
47. Talbi EG (2009) Metaheuristics from design to implementation. Wiley, New York
48. Zhou Y, Chen H, Zhou G (2014) Invasive weed optimization algorithm for optimization no-idle flow shop scheduling problem. Neurocomputing 137:285–292
49. Rad HS, Lucas C (2007) A recommender system based on invasive weed optimization algorithm. In: 2007 IEEE congress on evolutionary computation. IEEE, pp 4297–4304
50. Karimkashi S, Kishk AA (2010) Invasive weed optimization and its features in electromagnetics. IEEE Trans Antennas Propag 58(4):1269–1278
51. Ezugwu AE, Adeleke OJ, Akinyelu AA, Viriri S (2019) A conceptual comparison of several metaheuristic algorithms on continuous optimisation problems. Neural Comput Appl. https://doi.org/10.1007/s00521-019-04132-w
52. Davies DL, Bouldin DW (1979) A cluster separation measure. IEEE Trans Pattern Anal Mach Intell 2:224–227
53. Dunn JC (1974) Well-separated clusters and optimal fuzzy partitions. J Cybern 4(1):95–104
54. Caliński T, Harabasz J (1974) A dendrite method for cluster analysis. Commun Stat Theory Methods 3(1):1–27
55. Pakhira MK, Bandyopadhyay S, Maulik U (2004) Validity index for crisp and fuzzy clusters. Pattern Recognit 37(3):487–501
56. Das S, Suganthan PN (2010) Differential evolution: a survey of the state-of-the-art. IEEE Trans Evol Comput 15(1):4–31
57. Xu R, Xu J, Wunsch DC (2012) A comparison study of validity indices on swarm-intelligence-based clustering. IEEE Trans Syst Man Cybern Part B (Cybern) 42(4):1243–1256
58. Yang XS (ed) (2018) Mathematical analysis of nature-inspired algorithms. In: Nature-inspired algorithms and applied optimization. Studies in computational intelligence, vol 744. Springer, Cham. https://doi.org/10.1007/978-3-319-67669-2_1
59. Suzuki J (1995) A Markov chain analysis on simple genetic algorithms. IEEE Trans Syst Man Cybern 25(4):655–659
60. Clerc M, Kennedy J (2002) The particle swarm-explosion, stability, and convergence in a multidimensional complex space. IEEE Trans Evol Comput 6(1):58–73
61. Zhang Q, Chen D, Qin X, Gao Q (2010) Convergence analysis of invasive weed optimization algorithm and its application in engineering. J Tongji Univ (Nat Sci) 11. http://en.cnki.com.cn/Article_en/CJFDTotal-TJDZ201011025.htm. Accessed 12 Oct 2019
62. Yang XS, He XS (2018) Why the firefly algorithm works? In: Yang XS (ed) Nature-inspired algorithms and applied optimization. Studies in computational intelligence, vol 744. Springer, Cham
63. Yang XS (2014) Swarm intelligence based algorithms: a critical analysis. Evol Intell 7(1):17–28
64. Ghosh S, Das S, Vasilakos AV, Suresh K (2011) On convergence of differential evolution over a class of continuous functions with unique global optimum. IEEE Trans Syst Man Cybern Part B (Cybern) 42(1):107–124
65. Holland JH (1992) Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT Press, Cambridge
66. Kennedy J, Eberhart RC (1995) Particle swarm optimization. In: Proceedings of IEEE international conference on neural networks, Piscataway, NJ, pp 1942–1948
67. Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11(4):341–359
68. Yang XS (2010) Firefly algorithm, stochastic test functions and design optimisation. Int J Bio Inspired Comput 2(2):78–84
69. Mehrabian AR, Lucas C (2006) A novel numerical optimization algorithm inspired from weed colonization. Ecol Inform 1(4):355–366
70. Nwankwor E, Nagar AK, Reid DC (2013) Hybrid differential evolution and particle swarm optimization for optimal well placement. Comput Geosci 17(2):249–268
71. Blum C, Puchinger J, Raidl GR, Roli A (2011) Hybrid metaheuristics in combinatorial optimization: a survey. Appl Soft Comput 11(6):4135–4151
72. Wu YC, Lee WP, Chien CW (2011) Modified the performance of differential evolution algorithm with dual evolution strategy. In: International conference on machine learning and computing, vol 3, pp 57–63
73. Karaboga D, Basturk B (2007) A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J Glob Optim 39(3):459–471
74. Geem ZW, Kim JH, Loganathan GV (2001) A new heuristic optimization algorithm: harmony search. Simulation 76(2):60–68
75. Simon D (2008) Biogeography-based optimization. IEEE Trans Evol Comput 12(6):702–713
76. Rao RV, Savsani VJ, Vakharia DP (2011) Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. Comput Aided Des 43(3):303–315
77. Pham DT, Ghanbarzadeh A, Koç E, Otri S, Rahim S, Zaidi M (2006) The bees algorithm—a novel tool for complex optimisation problems. In: Intelligent production machines and systems. Elsevier, pp 454–459
78. Ezugwu AE, Prayogo D (2019) Symbiotic organisms search algorithm: theory, recent advances and applications. Expert Syst Appl 119:184–209
79. Goldberg DE, Holland JH (1988) Genetic algorithms and machine learning. Mach Learn 3(2):95–99
80. Robles G, Fresno J, Martínez-Tarifa J, Ardila-Rey J, Parrado-Hernández E (2018) Partial discharge spectral characterization in HF, VHF and UHF bands using particle swarm optimization. Sensors 18(3):746
81. Del Valle Y, Venayagamoorthy GK, Mohagheghi S, Hernandez JC, Harley RG (2008) Particle swarm optimization: basic concepts, variants and applications in power systems. IEEE Trans Evol Comput 12(2):171–195
82. Goldbarg EF, Goldbarg MC, de Souza GR (2008) Particle swarm optimization algorithm for the traveling salesman problem. In: Traveling salesman problem. Rijeka, pp 75–96
83. Yang XS (2010) Nature-inspired metaheuristic algorithms. Luniver Press, Bristol
84. Omran MG, Salman A, Engelbrecht AP (2006) Dynamic clustering using particle swarm optimization with application in image segmentation. Pattern Anal Appl 8(4):332
85. Masoud H, Jalili S, Hasheminejad SMH (2013) Dynamic clustering using combinatorial particle swarm optimization. Appl Intell 38(3):289–314

86. Ling HL, Wu JS, Zhou Y, Zheng WS (2016) How many clusters? A robust PSO-based local density model. Neurocomputing 207:264–275
87. Kuo R, Zulvia F (2013) Automatic clustering using an improved particle swarm optimization. J Ind Intell Inf 1(1):46–51. https://doi.org/10.12720/jiii.1.1.46-51
88. Nanda SJ, Panda G (2013) Automatic clustering algorithm based on multi-objective immunized PSO to classify actions of 3D human models. Eng Appl Artif Intell 26(5–6):1429–1441
89. Kao Y, Chen CC (2014) Automatic clustering for generalised cell formation using a hybrid particle swarm optimisation. Int J Prod Res 52(12):3466–3484
90. Abubaker A, Baharum A, Alrefaei M (2015) Automatic clustering using multi-objective particle swarm and simulated annealing. PLoS ONE 10(7):e0130995
91. Lee WP, Chen SW (2010) Automatic clustering with differential evolution using cluster number oscillation method. In: 2010 2nd international workshop on intelligent systems and applications. IEEE, pp 1–4
92. Saha I, Maulik U, Bandyopadhyay S (2009) A new differential evolution based fuzzy clustering for automatic cluster evolution. In: 2009 IEEE international advance computing conference. IEEE, pp 706–711
93. Maulik U, Saha I (2010) Automatic fuzzy clustering using modified differential evolution for image classification. IEEE Trans Geosci Remote Sens 48(9):3503–3510
94. Zhong Y, Zhang S, Zhang L (2013) Automatic fuzzy clustering based on adaptive multi-objective differential evolution for remote sensing imagery. IEEE J Sel Top Appl Earth Obs Remote Sens 6(5):2290–2301
95. Liu Y, Wu X, Shen Y (2011) Automatic clustering using genetic algorithms. Appl Math Comput 218(4):1267–1279
96. Rahman MA, Islam MZ (2014) A hybrid clustering technique combining a novel genetic algorithm with K-means. Knowl Based Syst 71:345–365
97. Ozturk C, Hancer E, Karaboga D (2015) Dynamic clustering with improved binary artificial bee colony algorithm. Appl Soft Comput 28:69–80
98. Kuo RJ, Zulvia FE (2018) Automatic clustering using an improved artificial bee colony optimization for customer segmentation. Knowl Inf Syst 57(2):331–357
99. Murty MR, Naik A, Murthy JVR, Reddy PP, Satapathy SC, Parvathi K (2014) Automatic clustering using teaching learning based optimization. Appl Math 5(08):1202
100. Peng H, Wang J, Shi P, Riscos-Núñez A, Pérez-Jiménez MJ (2015) An automatic clustering algorithm inspired by membrane computing. Pattern Recognit Lett 68:34–40
101. Kumar V, Chhabra JK, Kumar D (2016) Automatic data clustering using parameter adaptive harmony search algorithm and its application to image segmentation. J Intell Syst 25(4):595–610
102. Kapoor S, Zeya I, Singhal C, Nanda SJ (2017) A grey wolf optimizer based automatic clustering algorithm for satellite image segmentation. Procedia Comput Sci 115:415–422
103. Anari B, Torkestani JA, Rahmani AM (2017) Automatic data clustering using continuous action-set learning automata and its application in segmentation of images. Appl Soft Comput 51:253–265
104. Pacheco TM, Gonçalves LB, Ströele V, Soares SSR (2018) An ant colony optimization for automatic data clustering problem. In: 2018 IEEE congress on evolutionary computation (CEC). IEEE, pp 1–8
105. Elaziz MA, Nabil NEGGAZ, Ewees AA, Lu S (2019) Automatic data clustering based on hybrid atom search optimization and sine–cosine algorithm. In: 2019 IEEE congress on evolutionary computation (CEC). IEEE, pp 2315–2322
106. Sheng W, Chen S, Sheng M, Xiao G, Mao J, Zheng Y (2016) Adaptive multi-subpopulation competition and multi-niche crowding-based memetic algorithm for automatic data clustering. IEEE Trans Evol Comput 20(6):838–858
107. Zhou X, Gu J, Shen S, Ma H, Miao F, Zhang H, Gong H (2017) An automatic k-means clustering algorithm of GPS data combining a novel niche genetic algorithm with noise and density. ISPRS Int J Geo Inf 6(12):392
108. Agbaje MB, Ezugwu AE, Els R (2019) Automatic data clustering using hybrid firefly particle swarm optimization algorithm. IEEE Access 7:184963–184984. https://doi.org/10.1109/ACCESS.2019.2960925