applied sciences

MDPI

Article

Improved SOSK-Means Automatic Clustering Algorithm with a Three-Part Mutualism Phase and Random Weighted Reflection Coefficient for High-Dimensional Datasets

Abiodun M. Ikotun [1] and Absalom E. Ezugwu [1,2,*]

[1] School of Computer Science, University of KwaZulu-Natal, Pietermaritzburg Campus, Pietermaritzburg 3201, South Africa
[2] Unit for Data Science and Computing, North-West University, 11 Hoffman Street, Potchefstroom 2520, South Africa
* Correspondence: ezugwua@ukzn.ac.za or ezugwuabsalom79@gmail.com

Abstract: Automatic clustering problems require clustering algorithms to automatically estimate the number of clusters in a dataset. However, the classical K-means requires the specification of the required number of clusters a priori. To address this problem, metaheuristic algorithms are hybridized with K-means to extend the capacity of K-means in handling automatic clustering problems. In this study, we proposed an improved version of an existing hybridization of the classical symbiotic organisms search algorithm with the classical K-means algorithm to provide robust and optimum data clustering performance in automatic clustering problems. Moreover, the classical K-means algorithm is sensitive to noisy data and outliers; therefore, we proposed the exclusion of outliers from the centroid update's procedure, using a global threshold of point-to-centroid distance distribution for automatic outlier detection, and subsequent exclusion, in the calculation of new centroids in the K-means phase. Furthermore, a self-adaptive benefit factor with a three-part mutualism phase is incorporated into the symbiotic organism search phase to enhance the performance of the hybrid algorithm. A population size of $40 + 2g$ was used for the symbiotic organism search (SOS) algorithm for a well distributed initial solution sample, based on the central limit theorem that the selection of the right sample size produces a sample mean that approximates the true centroid on Gaussian distribution. The effectiveness and robustness of the improved hybrid algorithm were evaluated on 42 datasets. The results were compared with the existing hybrid algorithm, the standard SOS and K-means algorithms, and other hybrid and non-hybrid metaheuristic algorithms. Finally, statistical and convergence analysis tests were conducted to measure the effectiveness of the improved algorithm. The results of the extensive computational experiments showed that the proposed improved hybrid algorithm outperformed the existing SOSK-means algorithm and demonstrated superior performance compared to some of the competing hybrid and non-hybrid metaheuristic algorithms.

Keywords: symbiotic organism search; K-means; clustering algorithms; hybrid metaheuristics; automatic clustering; outliers

check for updates

Citation: Ikotun, A.M.; Ezugwu, A.E. Improved SOSK-Means Automatic Clustering Algorithm with a Three-Part Mutualism Phase and Random Weighted Reflection Coefficient for High-Dimensional Datasets. *Appl. Sci.* **2022**, *12*, 13019. https://doi.org/10.3390/app122413019

Academic Editor: Gaetano Zizzo

Received: 30 October 2022
Accepted: 15 December 2022
Published: 19 December 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Data clustering is an aspect of data mining where knowledge discovery from data requires that the data reveals the existing groups within itself. In cluster analysis, objects are grouped such that the intra-cluster distances among data objects are minimized while the inter-cluster distances are maximized. K-means is one of the most popular traditional clustering algorithms used for cluster analysis, due to its efficiency and simplicity. The k-means algorithm randomly selects a specified k number of initial cluster centroids as a representative center of each group. It then assigns data objects to their nearest cluster, based on the sum of squares point to nearest centroid distance. The mean of each cluster is calculated to generate an initial cluster with an updated cluster centroid. These data object

assignment and centroid update processes are iteratively repeated until an optimum cluster solution is obtained. The need to specify the number of clusters a priori makes the K-means algorithm unsuitable for automatic clustering. According to [1], estimating the optimal number of clusters in a dataset is a fundamental problem in cluster analysis, referred to as 'automatic clustering'. In most cases, K-means is hybridized with metaheuristic algorithms for automatic clustering [2].

The standard K-means algorithm is sensitive to noisy data and outliers [3,4]. Intuitively, the data points that are far away from their nearest neighbors are described as outliers [5]. The sensitivity of K-means to noise and outliers comes from the least square optimization procedure, normally employed for cluster analysis. K-means' reliance on the outlier-sensitive statistics (mean) for cluster updates compromises the clustering results when outliers are present in datasets. Several proposals have been reported in the literature that add a separate module for detecting and removing outliers as a data pre-processing step embedded in the K-means algorithm, before the actual data clustering procedure. A separate module for outlier detection significantly compromises the efficiency of the K-means algorithm. Chawla and Gionis [6] proposed an algorithm that simultaneously detected and removed outliers from the dataset during the clustering process, eliminating the need for a separate module. Their algorithm incorporated outlier detection into the K-means algorithm without complicated modification to the standard algorithm. However, their algorithm required specifying the desired number of outliers within the dataset, like how the desired number of clusters had to be specified. Olukanmi et al. [7] proposed a K-means variant for automatic outlier detection with the automatic specification of the number of clusters via Chebyshev-type inequalities. This paper incorporated this automatic detection and exclusion of outliers in the centroid update process of the standard K-means section of SOSK-means for a more robust hybridized algorithm.

The symbiotic organisms search (SOS) algorithm is a nature-inspired metaheuristic algorithm proposed by [8]. The algorithm has no basic control algorithm-specific parameter in its initialization stage. It works similarly to other well-known metaheuristic algorithms, such as the genetic algorithm (GA), the particle swarm optimization (PSO) algorithm, and the firefly algorithm (FA). The algorithm is inspired by the symbiotic relationships among organisms in a natural habitat, necessary to their survival in the ecosystem. The three symbiotic relationships (mutualism, commensalism, and parasitism) are modeled to find optimal solutions to optimization problems. As a population-based algorithm, the first two phases (mutualism and commensalism) use the best solution to exploit the population information in searching for potential solutions. The parasitism phase creates new solutions by modifying the existing solution, while, at the same time, removing inferior solutions [9]. The SOS algorithm is rated among the most competitive swarm intelligence-based metaheuristic algorithms for solving optimization algorithms, based on its simplicity and parameter-less attributes. SOS has been used to find the solution to many different real-world problems [10–16]. Tejani et al. [17] introduced a parameter setting of the beneficial factor, based on the normalized value of organisms, to create an adaptive SOS algorithm for structural optimization problems. A discrete version of SOS was designed and implemented by [18] for task scheduling in a cloud computing environment. Cheng, Prayogo and Tran [11] introduced another discrete SOS for multiple resources leveling optimization. Panda and Pani [10] introduced multi-objective SOS for handling multi-objective optimization problems. Kawambwa et al. [12] proposed a cloud-based model SOS using cloud-based theory to generate random number operators in the mutualism phase for power system distributed generators. Mohammadzadeh and Gharehchopogh [19] proposed three variants of SOS to solve the feature selection challenge, and Cheng, Cao, and Herianto [14] proposed an SOS optimized neural network–long short-term memory for obtaining hyperparameters of the neural network and long short-term memory for the establishment of a robust hybridization model for cash flow forecasting.

SOS has also been employed in solving automatic clustering problems [20–25]. Boushaki, Bendjeghaba, and Kamel [20] proposed a biomedical document clustering solution using

accelerated symbiotic organisms search, which required no parameter tuning. Yang and Sustrino [25] proposed a clustering-based solution for high-dimensional optimization problems using SOS with only one control parameter. Chen, Zhang, and Ning proposed an adaptive clustering-based algorithm using SOS for automatic path planning of heterogeneous UAVs [22]. Zainal and Zamil [23] proposed a novel solution for software module clustering problems using a modified symbiotic organism search with levy flights. The effectiveness of the SOS algorithm in solving automatic clustering problems was demonstrated by [21], where the SOS algorithm was used in clustering different UCI datasets. SOS has been hybridized with other metaheuristic algorithms for performance enhancement in cluster analysis. Rajah and Ezugwu [24] combined the SOS algorithm with four different metaheuristic algorithms to solve the automatic clustering problem and compared the clustering performances of each hybrid algorithm. Our earlier work combined the SOS algorithm with K-means to boost the traditional clustering algorithm's performance for automatic clustering [26].

However, SOS exhibits some limitations, such as slow convergence rate and high computational complexity [17,26–28]. Modifications have been made to the standard SOS to improve its performance. Tejani, Savsani and Patel [17] introduced adaptive benefit factors to the classical SOS algorithm, proposing three modified variants of SOS for improved efficiency. Adaptive benefit factors and benefit factors were effectively combined to achieve a good exploration–exploitation balance in the search space. Nama, Saha, and Ghosh [27] introduced a random weighted reflective parameter, to enhance searchability within the additional predation phase, to the classical SOS algorithm to improve the solving of multiple complex global optimization problems and improve the algorithm's performance. Secui [29] sought to improve the algorithm's capacity for the timely identification of stable and high-quality solutions by introducing new relations for solution updates at the mutualism and commensalism phases. A logistic map-generated chaotic component for finding promising zones was also added for the enhancement of the algorithm's exploration capacity. The removal of the parasitism phase reduced the computational load. Nama, Saha, and Ghosh [30] combined SOS with simple quadratic interpolation (SQI) for handling large-scale and real-world problems. Although their proposed hybrid SOS algorithm increased the algorithms' complexity, it provided an efficient and effective quality solution, with an improved convergence rate. Ezugwu and Adewumi [31] improved and extended the classical SOS using three mutation-based local search operators to improve population reconstruction, exploration, and exploitation capability and to accelerate the convergence speed. Other improvements reported in the literature include [32–38]. According to Chakraborty, Nama, and Saha [9], most of the proposed improvements could not guarantee finding the best optimal solution. As such, they proposed using a non-linear benefit factor where the mutual vector was calculated, based on the weights of two organisms for a broader and more thorough search. They also altered the parasitism phase to reduce the computational burden of the algorithm.

This paper proposes an improved version of the hybrid SOSK-means algorithm [24], called ISOSK-means, to address the common limitations of the individual classical algorithms and their hybridization variants for a more robust, efficient, and stable hybrid algorithm. The algorithm adopted a population size of $40 + 2g$ for a well distributed initial population sample, based on the assumption that selecting the right sample size produces a sample mean that approximates the true centroid on Gaussian distribution [39]. This ensured a substantially large population size at each iteration for early convergence in the SOS phase of the hybrid algorithm. The modified algorithm also employed a global threshold of point-to-centroid distance distribution in the K-means algorithm phase to detect outliers for subsequent exclusion in calculating the new centroids [3]. The detection and exclusion of outliers in the K-means phase assisted in improving the cluster result. The performance of the SOS algorithm was enhanced by introducing self-adaptive benefit factors, with a three-part mutualism phase, as suggested in [28]. These techniques enhanced the search for the optimum centroids in the solution space and improved the convergence

rate of the hybrid algorithm. The performance of the proposed algorithm was evaluated on 42 datasets with varying sizes and dimensions. Davies Bouldin (DB) index [40] and cluster separation (CS) index [41] were used as cluster validity indices for evaluating the performance of the algorithm. The choice of the two clustering validity indices was based on the duo's cluster validity approach of minimizing the intra-cluster similarity and maximizing the inter-cluster similarity, which is the same as the primary objective of data clustering. The CS index was only applied to 12 datasets for comparison with the classical version of the improved hybrid algorithm. The two indices, based on their validity approach, are credited with the ability to produce clusters that are well separated, with the minimum intra-cluster distance and maximum inter-cluster distance [42–44]. The DB index can guide the clustering algorithm using a partitioning approach without the need to specify the number of clusters, and the CS index is known for its excellent performance in the identification of clusters with varying densities and sizes [40,41,43]. The enhanced outcomes were compared with the existing hybrid SOSK-means algorithm and other comparative non-hybrid and hybrid metaheuristic algorithms on large dimensional datasets. The performance of the proposed improved hybrid algorithm was further validated using the nonparametric Friedman mean rank tests and Wilcoxon signed-rank statistical tests. The results of the extensive computational experiments showed that the proposed improved hybrid algorithm (ISOSK-means) outperformed the existing SOSK-means algorithm, and demonstrated superior performance over some of the competing hybrid and non-hybrid metaheuristic algorithms. The main contributions of this paper are as follows:

- Integration of a robust module in the proposed SOS-based K-means hybrid algorithm using outlier detection and the exemption technique in the K-means phase for a more effective cluster analysis with compact clusters.
- Integration of a three-part mutualism phase with a random weighted reflection coefficient into the SOS phase, for a more productive search in the solution space for early convergence and reduced computational time.

Adopting a population size of $40 + 2g$ for a well-distributed initial population in the SOS phase, allowing for a large enough solution space per iteration, aimed to ensure early convergence of the proposed clustering algorithm.

The remaining sections of this paper are organized as follows. Section 2 presents related work on hybrid algorithms involving SOS and K-means. It also includes a brief introduction to the standard K-means algorithm, standard SOS algorithm and hybrid SOSK-means. Section 3 describes the improvements integrated into the K-means and SOS algorithms and presents the description of the proposed ISOSK-means. The simulation and comparison of results of the proposed ISOSK-means are presented in Section 4, while Section 5 presents the concluding remarks and suggestions for future directions.

2. Related Work

There are existing works in the literature that reported hybridizations involving either K-means or symbiotic organism search or both. However, it is worth stating here that hybridizations for clustering problems are few. In [45], the SOS algorithm was hybridized with the improved opposition-based learning firefly algorithm (IOFA) to improve the exploitation and exploration of IOFA. SOS was combined with differential evolution (DE) in [46] to improve the convergence speed and optimal solution quality of shape and size truss structure optimization in engineering optimization problems. The conventional butterfly optimization algorithm (BOA) was combined with the first two phases of the SOS for enhancement of global and local search behavior of the BOA. In [47], a modified strategy of SOS was hybridized with the mutation strategy of the DE to ensure the preservation to the SOS local search capability, while maintaining its ability to conduct global search. Other reported hybrids involving SOS include the following: Ref. [48] combined GA, PSO and SOS algorithms for continuous optimization problems; Ref. [24] reported four hybrid algorithms for automatic clustering, combining SOS with FA, for teaching–learning based

optimization (TLBO), DE, and PSO algorithms. Other hybridization algorithms involving SOS can be found in the review work on SOS algorithms conducted by [49].

Several hybridizations involving K-means with other metaheuristic algorithms were reported in the review work presented by [2]. Recent hybridizations involving K-means include the following: Ref. [50] combined GA with K-means and support vector machine (SVM) for automatic selection of optimized cluster centroid and hyperparameters tunning; Ref. [51] combined multi-objective individually directional evolutionary algorithm (IDEA) with K-means for multi-dimensional medical data modeling using fuzzy cognitive maps; Ref. [52] presented an hybrid of K-means with PSO for semantic segmentation of agricultural products; Ref. [53], combined K-means algorithm with particle swarm optimization for customer segmentation.

For hybridization involving SOS and K-means, as mentioned earlier, the reported literature works are very minimal showing that research in this aspect is still shallow. In [25], a clustering-based SOS for high-dimensional optimization problems was proposed, combining an automatic K-means with symbiotic organism search for efficient computation and better searching quality. The automatic K-means was used to generate subpopulations for the SOS algorithm to create a sub-ecosystem that made the combination of global and local searches possible. The mutualism and commensalism phases formed the local search, where solutions were allowed to interact within each cluster. For the global search, only best solutions from each cluster were allowed to interact across the clusters under the parasitism phase. In this case, the K-means algorithm was used as a pre-processing phase for the SOS algorithm to enhance its performance in finding solutions to high dimensional optimization problems. In [26], the standard algorithms SOS and K-means were combined to solve automatic clustering problems. The SOS phase resolved the initialization challenge for the K-means algorithm by automatically determining the optimum number of clusters and generating the corresponding initial cluster centers. This ensured the K-means algorithm avoided the possibility of local optimum convergence while improving the cluster analysis capability of the algorithm. However, the problem of low convergence persisted in the hybrid algorithm.

In this work, we hoped to further improve on the clustering performance of the previous SOS-based K-means hybrid algorithm in [26] for automatic clustering of high-dimensional datasets, by incorporating some improvements into each of the standard algorithms, SOS and K-means, so as to achieve better convergence and more compact clusters.

2.1. K-Means Algorithm

The K-means algorithm is a partitional clustering algorithm that iteratively groups a given dataset X in R^d into k number of clusters $C_1, C_2, C_3, \ldots, C_k$ such that:

$$C_i \neq \varnothing; \tag{1}$$

$$\cup_1^k C_i = X \tag{2}$$

$$C_i \cap C_j \neq \varnothing \; \forall i, \; j \in 1, 2, \ldots, k \; and \; i \neq j \tag{3}$$

based on a specified fitness function. The K-means algorithm handles the partitioning process as an optimization problem to minimize the within-cluster variance σ:

$$\sigma = \sum_{x \in X} \min_{cc \in C} ||x - cc||^2 \tag{4}$$

with cluster center cc_i uniquely defining each cluster as:

$$cc_i = \frac{1}{|C_i|} \sum_{x \in C_i} x \tag{5}$$

with the set of k centres

$$CC = \{cc_1, \ldots, cc_k\} \tag{6}$$

representing the solution to the K-means algorithm [54].

The standard K-means algorithm involves three major phases: the initialization phase, the assignment phase, and the centroid update phase. During the initialization phase, the initial cluster centers are randomly selected to represent each cluster. This is followed by the data object assignment phase, where each data point in the dataset is then assigned to the nearest cluster, based on the shortest centroid-point distance. Each cluster centroid is then re-evaluated during the centroid update phase. The last two phases are repeated until the centroid value remains constant in consecutive iterations [44].

The K-means clustering is an NP-hard optimization problem [55], with the algorithm having a time complexity of $O(nkt)$, where n represents the number of data points in the dataset, k represents the number of clusters, and t denotes the number of iterations required for convergence. The computational complexity is a function of the size of the dataset, hence, clustering large real-world, or dynamic, datasets using the K-means algorithm incurs a sizeable computational time overhead. Moreover, the number of clusters k is required to be a user-specified parameter for the K-means algorithm. In most real-world datasets, the number of clusters is not known a priori; therefore, specifying the correct number of clusters in such a dataset is arduous. Furthermore, the random selection of initial cluster centroids incurs the possibility of the algorithm getting stuck in the local optimum [54]. Based on these challenges, many variants of K-means have been proposed in the literature [2,4] to improve the performance of the K-means algorithm. One of the areas being exploited for improving the standard K-means algorithm is hybridizing K-means with metaheuristic algorithms [2].

2.2. SOS Algorithm

Cheng and Prayogo [8] proposed the concept of SOS simulating the interactive relationships between organisms in an ecosystem. In an ecosystem, most organisms do not live in isolation because they need to interact with other organisms for their survival. These relationships between organisms, known as symbiotic relationships, are, thus, defined using three possible major interactions: mutualism, commensalism, and parasitism. Mutualism represents a symbiotic relationship where the participating organisms benefit from each other. For instance, an oxpecker feeds on the parasites living on the body of a zebra or rhinoceros, while the symbiont, in turn, enjoys the benefit of pest control. Commensalism describes a relationship where only one of the participating organisms benefits from the association while the other organism does not benefit from the relationship, although it is not negatively affected either. An example of this is orchids, which grow on branches and trunks of trees to access sunlight and obtain nutrients from the branches. As a slim, tender plant, their existence does not harm the tree on which they grow. The relationship between humans and mosquitos provides a perfect scenario of the parasitism symbiotic relationship, in which one of the organisms (mosquito) benefits from the association while simultaneously causing harm to the symbiont (human). These three relationships are captured in the SOS algorithm [8]. In most cases, the type of relationship increases the fitness of benefiting organisms, giving them a long-term survival advantage.

In the SOS algorithm, the three-symbiosis relationship is simulated as the three phases of the algorithm. At the initial stage, a population of candidate solutions are randomly generated to the search space for solution representations for optimal global solution search. This set of solutions forms the initial ecosystem. where each individual candidate solution represents an organism in the ecosystem. A fitness value is associated with each organism to determine the measure of its adaptation capability with respect to the desired objective. Subsequently, each candidate solution is further updated using the three simulated symbiotic relationship phases to generate a new solution. For each phase, a new solution is only accepted if its fitness value is better than the previous one. Otherwise, the initial solution is retained. This iterative optimization process is performed until the termination criteria are met. The three simulated symbiotic relationship phases of the SOS algorithm are described below.

Given two organisms x_i and x_j co-existing in an ecosystem such that i and j represent the iterative values of the optimization, ranging from 1 to d, where d is the problem dimension and $i \neq j$, an organism, x_j, is randomly selected during the mutualism phase to participate in a mutual relationship with x_i such that the two organisms enjoy a common benefit from their interaction for survival. Their interaction yields new solutions x_{inew} and x_{jnew} based on Equations (7) and (8), respectively, with Equation (9) representing the mutual benefit x_{mutual} enjoyed by both organisms.

$$x_{inew} = x_i + rand(0,1) \times (x_{best} - x_{mutual} \times BF_1) \tag{7}$$

$$x_{jnew} = x_j + rand(0,1) \times (x_{best} - x_{mutual} \times BF_2) \tag{8}$$

$$x_{mutual} = \frac{x_i + x_j}{2} \tag{9}$$

The expression $(x_{best} - x_{mutual} \times BF_1)$ as shown in Equations (7) and (8), represents the mutual survival efforts exhibited by each organism to remain in the ecosystem. The highest degree of adaptation achieved in the ecosystem is represented by the x_{best} which acts as the target point for increasing the fitness of the two interacting organisms. Each organism's benefit level of value 1 or 2 is randomly determined to indicate the organism's benefit from the relationship, which can either be full or partial. This benefit level is denoted as BF (benefit factor) in the equations. The BF value is generated using Equation (10):

$$BF = 1 + round[rand(0,1)] \tag{10}$$

As stated earlier, the newly generated solutions are accepted as a replacement for the existing solution if, and only if, they are better. In other words, new solutions are rejected if the existing solutions are better. Equations (11) and (12) incorporate this:

$$x_{inew} = x_i + rand(0,1) \times (x_{best} - x_{mutual} \times BF_1)\ if\ f(x_{inew}) < f(x_i) \tag{11}$$

$$x_{jnew} = x_j + rand(0,1) \times (x_{best} - x_{mutual} \times BF_2)\ if\ f(x_{jnew}) < f(x_j) \tag{12}$$

In simulating the commensalism phase, an organism x_j is randomly selected for interaction with organism x_i exhibiting the characteristic of the commensalism symbiosis relationship where only one of the two organisms x_i derives a benefit from the relationship. This relationship is simulated using Equation (13):

$$x_{inew} = x_i + rand(-1,1) \times (x_{best} - x_j)\ if\ f(x_{inew}) > f(x_i) \tag{13}$$

Thus, only the benefiting organisms generate a new solution, as reflected by Equation (10). The new solution is accepted only if it is better, in terms of fitness, than the previous solution before the interaction.

The parasitism phase is the last phase, which is simulated using Equation (14). In simulating the parasitism symbiotic relationship in the SOS algorithm, a duplicate copy of the x_i organism is created as a parasite vector with some of its selected dimensions modified using a random number. An organism x_j is then randomly selected from the ecosystem to play host to the parasite vector $x_{parasite}$. If the fitness value of $x_{parasite}$ is better than that of x_j, then $x_{parasite}$ replaces x_j in the ecosystem. If the opposite happens, and x_j builds immunity against $x_{parasite}$, then $x_{parasite}$ is removed from the ecosystem:

$$x_{parasite} = rand(0,1) \times (UB - LB) + LB \tag{14}$$

2.3. Description of Data Clustering Problem

In data clustering problems, data objects sharing similar characteristics are grouped together into a cluster, such that data objects in one cluster are different from data objects in other clusters. In most cases, the number of clusters is specified, while in some cases,

especially in high dimensional datasets, predetermining the number of clusters is not feasible. In optimization terms, data objects are clustered, based on the similarity or dissimilarity between them, such that the inter-cluster similarity is minimized (maximizing inter-cluster dissimilarity), while maximizing the intra-cluster similarity (or minimizing intra-cluster dissimilarity). The description of a clustering problem as an optimization problem is given below.

Given a dataset $X = (x_1,\ x_2 \ldots, x_n)$ of dimension d which represents the number of attributes or features of the data objects in the datasets such that $x_i\ (i = 1, 2, \ldots, n)$ where n is the number of data objects in the dataset. Each data object $x_i = (x_{i1},\ x_{i2},\ \ldots, x_{id})$, where $x_{i1},\ x_{i2},\ \ldots, x_{id}$ represents all the features for data object x_i. The dataset X can be represented in a matrix form as shown in Equation (15):

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_i \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} x_{1,1} & x_{1,2} & \ldots & x_{1,d} \\ x_{2,1} & x_{2,2} & \ldots & x_{2,d} \\ \vdots & \vdots & \vdots & \vdots \\ x_{i,1} & x_{i,2} & x_{i,j} & x_{i,d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \ldots & x_{n,d} \end{bmatrix} \tag{15}$$

X is required to be grouped into k number of clusters to satisfy Equations (1)–(3), using Equation (4) as the objective function, with each cluster having a cluster center defined in Equations (5) and (6). For an automatic clustering, the number of clusters is not defined. Therefore, finding the optimum number of clusters represented by equation (6) becomes the optimization problem that seeks to optimize the function $f(CC,\ D)$ over all possible clustering of X where function f represents the global validity index for obtaining the best quality clustering solution and D represents the distance metric measure, stated in Equation (4).

2.4. Cluster Validity Indices

The cluster validity indices are qualitative methods, like the statistical mathematical functions, for evaluating the clustering quality of a clustering algorithm. A cluster validity index has the capacity to accurately determine the cluster number in a dataset, as well as find the proper structure of each cluster in the dataset [56]. The principal concerns of most validity indices in clustering are to determine clustering compactness, separation, and cohesion. The cluster validity indices are presented as the fitness function during the optimization process of the clustering algorithms. The DB index and CS index are used as the cluster validity indices to evaluate the quality of the clustering results.

The DB index determines the quality of a clustering result by using the average inter-cluster similarity between any two clusters and the intra-cluster similarity between data objects within a cluster. The average intra-cluster similarity value is evaluated against the average inter-cluster similarity value using Equations (16) and (17). In the DB index, the fitness function is minimized during the data clustering. This implies that a smaller index value indicates better compactness or separation, and vice versa:

$$f_{dbi} = \frac{1}{K} \sum_{K}^{1} d\left(cc_i, cc_j\right) \tag{16}$$

$$d\left(cc_i, cc_j\right) = max\left\{ \frac{wc_{dist(i)} - wc_{dist(j)}}{ic_{dist(ij)}}\ |1 \le i, j \le K,\ i \ne j \right\} \tag{17}$$

where $wc_{dist(i)}$ and $wc_{dist(j)}$ represent the within-cluster distance for clusters i and j, $ic_{dist(ij)}$ represents the inter-cluster distance between the two clusters i and j. The $d\left(cc_i, cc_j\right)$ is the inter-cluster distance between the two cluster centroids cc_i and cc_j of the respective clusters i and j.

The CS index estimates the quality of a clustering result by finding the ratio of the sum of the intra-cluster scatter to the inter-cluster separation using Equation (18):

$$f_{csi} = \frac{\sum_{i=1}^{K}\left[\frac{1}{|C_i|}\sum_{wc_{scat(i)}i\in C_i} max_{bc_{sep(j)}j\in C_i}\left\{d\left(wc_{scat(i)}, bc_{sep(j)}\right)\right\}\right]}{\frac{1}{K}\sum_{i=1}^{K}\left[min_{j\in K,\, j\neq i}\left\{V\left(cc_i,\, cc_j\right)\right\}\right]} \quad (18)$$

where $wc_{scat(i)}$ and $bc_{sep(j)}$ represents within-cluster scatter and between-cluster separation with the distance measure given as $d\left(wc_{scat(i)}, bc_{sep(j)}\right)$. The CS index is rated as being more computationally intensive but more efficient, compared with the DB index, and gives a more quality solution than the DB index. In the CS index, the fitness function is also minimized, therefore a lower validity index implies better separation or compactness, while a higher index value implies weak separation or compactness.

2.5. Hybrid SOSK-Means

A hybridization of SOS and K-means proposed by [26] found the solution to the automatic clustering algorithm. The proposed algorithm employed the standard SOS algorithm to globally search for the optimum initial cluster centroids for the K-means algorithm. This resolved the problems associated with the random generation of the initial cluster centroid without initial specification of the value of k. The problem of multiple parameter controls required in most nature-inspired population-based metaheuristic algorithms (e.g., GA, PSO, FA) was also avoided, since the SOS required only the basic control parameters for a metaheuristic algorithm, such as the number of iterations and population size, with no algorithm-specific parameters. The SOS as a global search algorithm ensured that K-means returned a global optimum solution to the clustering problem, canceling the possibility of getting stuck in the local optimum. According to [26], the SOSK-means algorithm combined the local exploitation capability of the standard K-means with less parameter tuning and global exploration, as provided by the SOS algorithm with implementation simplicity common to the two algorithms, to produce a powerful, efficient, and effective automatic clustering algorithm. The resulting hybrid algorithm was credited with better cluster solutions than the results from the standard SOS and K-means algorithms executed separately.

The SOSK-means algorithm commences by initializing the population of n organisms representing the ecosystem, randomly generated using Equation (19), and the fitness value of each organism is calculated based on the fitness function for the optimization process. The initial organisms are generated by the expression $rand(1, K \times m)$ in the equation, representing random and uniformly distributed points within the ecosystem, the solution search space is bounded between specified lower and upper limits a and b, respectively, for the clustering problem:

$$x_i = rand(1, K \times m) \times (b - a) + a \quad \ni \; i = 1, 2, \ldots, n \quad (19)$$

New candidate solutions are subsequently generated using the three phases of the SOS algorithm, described under the SOS algorithm. The optimum result from the phases of the SOS algorithm is passed as the optimum cluster centroids for initializing the K-means algorithm. These processes are iteratively performed until the stopping criterion is achieved. The details on the design and performance of hybrid SOSK-means can be found in [26]. Algorithm 1 presents the pseudocode for the hybrid SOSK-means algorithm. The flowchart for Algorithm 1 can be found in [26].

Algorithm 1: Hybrid SOSK-means clustering pseudocode [26]

Input: *Eco_size:* population size — *ULSS*: upper limit for search space
Max_iter: maximum number of iterations — *LLSS*: lower limit for search space
PDim: problem dimension — *ObjFn(X):* fitness (objective) function
Output: Optimal Solution

Create an initial population of organisms $X = (X_1, X_2, \ldots, X_{ecosize})$
Calculate the fitness of each organism
Keep the initial population's best solution *BestX*
while $iter \leq Max\_iter$
  for $i = 1$ *to Eco_size* **do**
   // 1st Phase: Mutualism //
    Select index j $(1 \leq j \leq Eco\_size; j \neq i)$ randomly
     $BF_1 = (1 + round(rand(0,1)))$
     $BF_2 = (1 + round(rand(0,1)))$
      $X_{mutual} = \left(\frac{X_i + X_j}{2}\right)$
      for $k = 1$ *to PDim* ***do***
        $X_{inew} = X_i + rand(0,1) * (BestX - BF_1 * X_{mutual})$
        $X_{jnew} = X_j + rand(0,1) * (BestX - BF_2 * X_{mutual})$
      end for
      if $(ObjFn(X_{inew}) < ObjFn(X_i)$
        $X_i = X_{inew}$
      end if
      if $(ObjFn\left(X_{jnew}\right) < ObjFn\left(X_j\right)$
        $X_j = X_{jnew}$
      end if
      // 2nd Phase: Commensalism //
      Select index j $(1 \leq j \leq Eco\_size; j \neq i)$ randomly
      for $k = 1$ *to PDim* **do**
        $X_{inew} = X_i + rand(-1,1) * \left(BestX - X_j\right)$
      end for
      if $ObjFn(X_{inew}) < ObjFn(X_i)$
        $X_i = X_{inew}$
      end if
      //3rd Phase: Parasitism //
      Select index j $(1 \leq j \leq Eco\_size; j \neq i)$ randomly
      for $k = 1$ *to PDim* **do**
        if $rand(0,1) < rand(0,1)$
          $X_{parasite} = X_i$
        else
          $X_{parasite} = rand(0,1) * (ULSS[K] - LLSS) + LLSS$
        end if
      end for
      if $ObjFn\left(X_{parasite}\right) < ObjFn\left(X_j\right)$
        $X_j = X_{parasite}$
      end if
        Update current population's best solution *BestX*
        //K-means Clustering Section//
        K-means' initialization using the position of the *BestX*
        Execute K-means clustering
    end for
    iter = *iter* + 1
  end while

3. Modified SOSK-Means

The adoption of standard SOS and K-means algorithms in the hybrid SOSK-means combines the benefits of the two classical algorithms to produce an efficient algorithm. However, some of the individual challenges of the two algorithms remain. The SOSK-means algorithm still suffers from the low convergence rate peculiar to the classical SOS algorithm. The computational time is affected by the dataset size as it is with the classical K-means clustering. The proposed ISOSK-means follows the structure of the original SOSK-means algorithm, described in Section 2.2, with a few modifications incorporated into the two classical algorithms to further enhance the performance of the hybrid algorithm. The summary flowchart for ISOSK-means is shown in Figure 1.

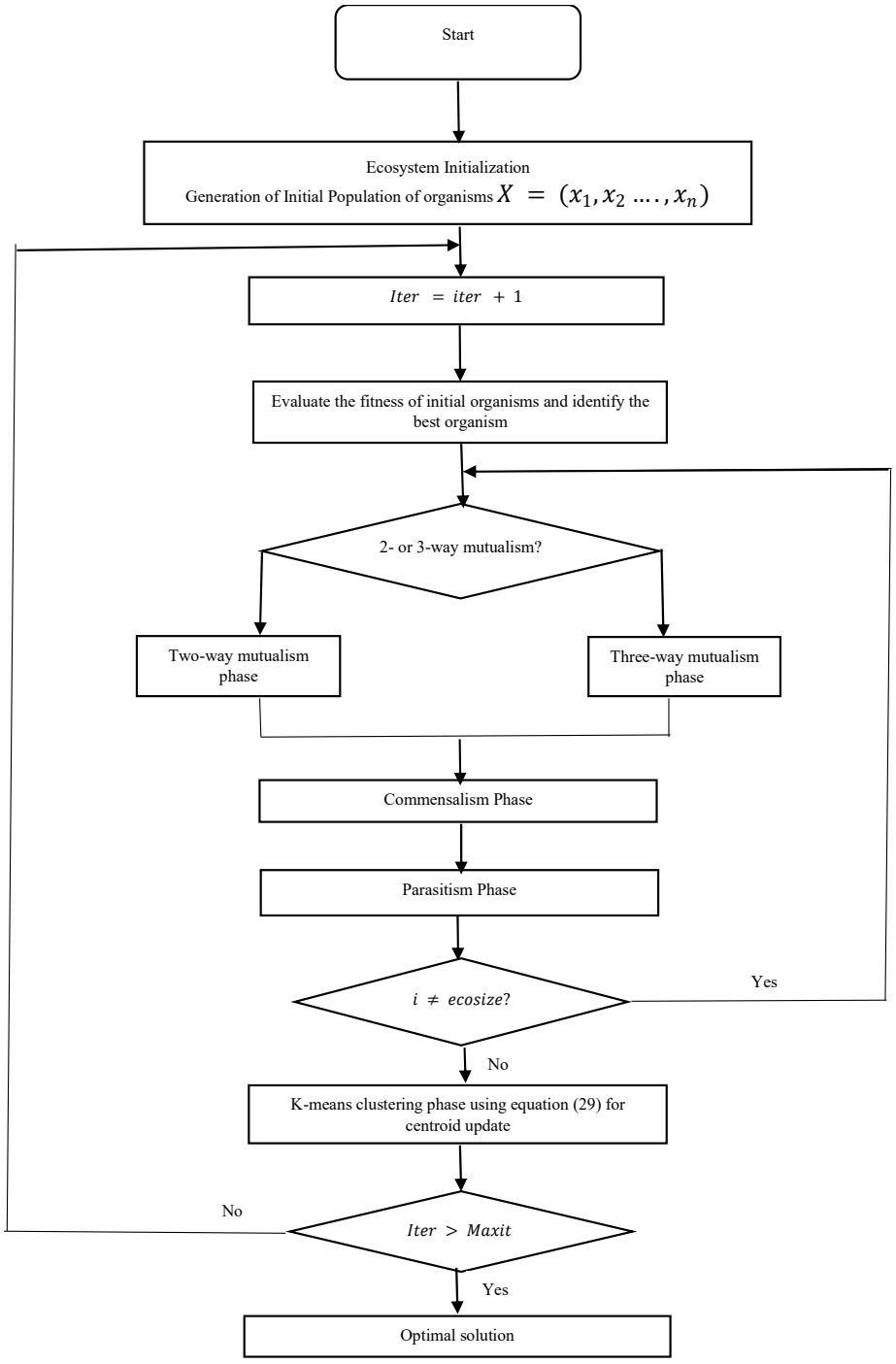


Figure 1. Flowchart for ISOSK-means clustering algorithm.

3.1. Modification in the SOS Phase

The modifications in the SOS phase affect three major parts of the standard algorithm, the initialization phase, the mutualism phase, and the commensalism phase. During the initialization phase, the population size is required to be specified as an input parameter. This is common to most metaheuristic algorithms, with no specific rule for determining the best population size for optimum algorithm performance. In the improved SOSK-means, the rule for determining the population size suggested by [39] was adopted. The population size was constructed as $40 + 2g$ where $g > 1$, representing the minimum number of possible groupings of the datasets for an optimum search in the metaheuristic algorithm to achieve a well-distributed space for initial population of solutions that scales well with the data size. In this work, the value for g was given as 2. The idea was coined from generalizing the central limit theorem (CLT) for a mixture distribution that infers sample population parameters. The central limit theorem states [57] that the mean (average) of a random sample follows a normal distribution as the mean, μ and variance, σ of the population from which it is selected According to [39], the selection of a data sample having a size that is sufficiently large enough to be well above the total number of clusters has a high possibility of containing data objects that belong to each of the clusters. At each iteration, new population samples are randomly selected as initial populations. The use of the population size of $40 + 2g$ ensured that the initial selection of candidate solutions was well spread across the solution search space for the generation of optimum values in the SOS phase for a faster search for the optimum cluster centroids, which, subsequently, served as the K-means algorithm's initial centroids [39]. The different initial population samples at each iteration in the optimization process represented the varied samples of the initial solution space. Selecting such a sufficiently large population size has a high probability of containing the cluster centroids for all the clusters within the dataset. This invariably increased the convergence rate of the SOS algorithm, and the computational time was, consequently, reduced.

To upgrade the SOS algorithm's performance, a three-part mutualism phase, with the random weighted reflection coefficient introduced by [28], was incorporated. Three-part mutualism reflects the possibility of three organisms interacting mutually, each deriving benefits that sustain their existence in the ecosystem. A three-part mutual relationship can be observed in the interaction between sloths, algae, and moths. A three-part mutualism is incorporated alongside the use of two interacting organisms in the mutualism phase. In three-part mutualism, three organisms are chosen randomly in the ecosystem to interact for the generation of newer organisms. Each organism is simulated using Equations (20)–(22), respectively. Their mutual benefits are simulated using Equation (23), involving contributions from the three organisms. At the commencement of the mutualism phase for the organism's update, a random probability is used to determine whether an organism will be engaged in three-part mutualism or not. Thus, a choice is required between the normal dual mutualism interaction and the three-part mutualism. The flowchart depicting the three-part mutualism phase is shown in Figure 2.

$$x_{inew} = x_i + rand(0,1) \times (x_{best} - x_{mutual} \times MBF_1) \tag{20}$$

$$x_{jnew} = x_j + rand(0,1) \times (x_{best} - x_{mutual} \times MBF_2) \tag{21}$$

$$x_{knew} = x_k + rand(0,1) \times (x_{best} - x_{mutual} \times MBF_3) \tag{22}$$

$$x_{mutual} = \frac{x_i + x_j + x_k}{3} \tag{23}$$

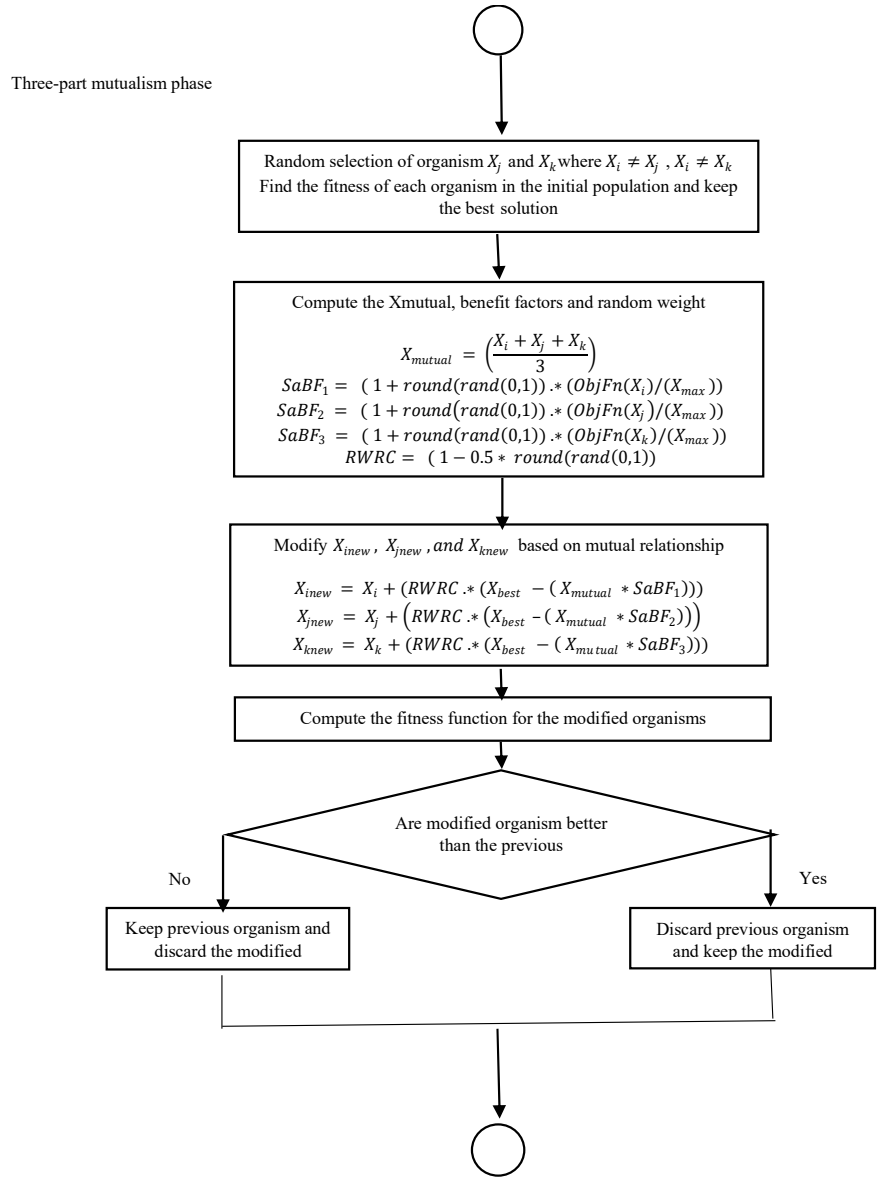


Figure 2. Flowchart for three-part mutualism phase.

The BF is also modified using the modified benefit factor (*MBF*) in [28] to increase the quality of the solution. In the *MBF*, the relativity of the fitness value of each organism under consideration, with respect to the maximum fitness value, is incorporated to achieve a benefit factor that is self-adaptable. The *MBF* for each organism was simulated using Equations (24)–(26). This ensured the automatic maintenance of the values of the benefit factors throughout the search process.

$$MBF_1 = 1 + rand(0,1) \times \frac{x_{1fitness}}{Maxfitness}, \ \ if \ Bestfitness \ \neq 0 \tag{24}$$

$$MBF_2 = 1 + rand(0,1) \times \frac{x_{2fitness}}{Maxfitness}, \ \ if \ Bestfitness \ \neq 0 \tag{25}$$

$$MBF_3 = 1 + rand(0,1) \times \frac{x_{3fitness}}{Maxfitness}, \ \ if \ Bestfitness \ \neq 0 \tag{26}$$

The modified benefit factor (*MBF*) incorporated the possibility of variations in the actual benefits derived from the interactions, instead of the static 1 or 2 implemented in the standard SOS. As stated earlier, a benefit factor of 1 caused slow convergence by reducing

the search step, while a benefit factor of 2 reduced the search ability by speeding up the search process. In the case of the *MBF*, the benefit accrued to an organism was a factor of its fitness with respect to the best fitness in the ecosystem.

Moreover, a random weight ($SimRW$), suggested by [28], for the SOS algorithm's performance improvement was added to each dimension of the organism. The weight was simulated using Equation (27):

$$SimRW_j = 1 - 0.5 \times \left(1 + r_j^1\right),\ r^i\ \in [0,1]\ where\ j = 1,2,3\ldots,D \tag{27}$$

3.2. Modification in the K-Means Phase

The modification effected in the K-means phase of the improved SOSK-means algorithm addressed the misleading effects of outliers in the dataset, which usually affect the standard K-means algorithm at the centroids update stage. The use of the 'average' as the statistic for calculating the new cluster centroid was sensitive to outliers [3]. According to [58,59], K-means assume Gaussian data distribution, being an instance of the Gaussian mixture model. As such, about 99.73% of the data points within a cluster record point-to-centroid distances of three standard deviations (σ) from the cluster centroid. Therefore, any point with a point-to-centroid distance that is outside this range with respect to its cluster is considered an outlier. Subsequently, it is excluded in the computation of the centroid update.

According to [59], a standard deviation (σ) was taken to be 1.4826 of the median absolute deviation (*MAD*) for population distribution. The median absolute deviation gave a robust statistical dispersion measure and was more resilient to outliers than the standard deviation [60]. Given a point-to-centroid distance threshold T_{ptc}:

$$\sigma = 1.4826 MAD_{ci} \tag{28}$$

$$MAD_{ci} = median|ci - median(ci)| \tag{29}$$

$$T_{ptc} = 3\sigma$$

$$\therefore\ T_{ptc} = 3(1.4826 MAD_{ci})$$

$$\therefore\ T_{ptc} = 4.4478 MAD_{ci} \tag{30}$$

The centroid update was calculated using Equation (31) in the classical K-means:

$$\mu_i^{(it+1)} = \frac{1}{\left|C_i^{(it)}\right|} \sum_{x_j \epsilon C_i} x_j \tag{31}$$

However, in the improved SOSK-means, T_{ptc} was introduced for outlier detection and exclusion in the new centroid update using Equation (32):

$$\mu_i^{(it+1)} = \frac{1}{\left|C_i^{(it)}\right| - \left|C_{io}^{(it)}\right|} \sum_{x_j \epsilon C_i} x_j\ : x_j = 0\ if\ ||x - \mu_i||\ > T_{ptc} \tag{32}$$

where $C_{io}^{(it)}\ \epsilon\ C_i^{(it)}$ represents the sets of points x_j assigned to cluster $C_i^{(it)}$ which has a point-to-centroid distance of $||x - \mu_i|| > T_{ptc}$. The main difference between Equations (31) and (32) is that in the latter equation, the sets of points assigned to a cluster with a point-to-centroid distance greater than thrice the standard deviation (3σ) were excluded from the centroid update calculation. This excluded the outliers from contributing to the mean square error that was being minimized. Algorithm 2 presents the pseudocode for the proposed improved SOSK-means algorithm.

Algorithm 2: The proposed improved SOSK-means Pseudocode

Eco_size: population size *ULSS*: Upper limit for search space
Max_iter: maximum number of iterations *LLSS*: Lower limit for search space
PDim: problem dimension *ObjFn(X)*: fitness (objective) function
Optimal Solution
1: Create an initial population of organisms $X = (X_1, X_2, \ldots, X_{ecosize})$
2: Calculate the fitness of each organism
3: Keep the initial population's best solution $BestX$
4: Keep the initial population's maximum solution X_{max}
5: **while** *iter* $\leq$ Max_*iter*
6: **for** $i = 1$ *to Eco_size* **do**
7: // 1st Phase: Mutualism //
8: Select index j $(1 \leq j \leq Eco\_size; j \neq i)$ randomly
9: Select index k $(1 \leq k \leq Eco\_size; k \neq i; k \neq j)$ randomly
10: $rand_1 = (1 + round(rand(0,1)))$
11: $rand_2 = (1 + round(rand(0,1)))$
12: **if** $rand_1 < rand_2$
13: $X_{mutual} = \left(\frac{X_i + X_j}{2}\right)$
14: $SaBF_1 = (1 + round(rand(0,1)) .* (ObjFn(X_i)/(X_{max}))$
15: $SaBF_2 = \left(1 + round\left(rand(0,1)) .* (ObjFn(X_j)\right)/(X_{max})\right)$
16: $RWRC = (1 - 0.5 * round(rand(0,1))$
17: **for** $n = 1$ *to PDim* ***do***
18: $X_{inew} = X_i + (RWRC .* (X_{best} - (X_{mutual} * SaBF_1)))$
19: $X_{jnew} = X_j + (RWRC .* (X_{best} - (X_{mutual} * SaBF_2)))$
20: **end for**
21: **if** $(ObjFn(X_{inew}) < ObjFn(X_i)$
22: $X_i = X_{inew}$
23: **end if**
24: **if** $(ObjFn\left(X_{jnew}\right) < ObjFn\left(X_j\right)$
25: $X_j = X_{jnew}$
26: **end if**
27: **else**
28: $X_{mutual} = \left(\frac{X_i + X_j + X_k}{3}\right)$
29: $SaBF_1 = (1 + round(rand(0,1)) .* (ObjFn(X_i)/(X_{max}))$
30: $SaBF_2 = \left(1 + round\left(rand(0,1)) .* (ObjFn(X_j)\right)/(X_{max})\right)$
31: $SaBF_3 = (1 + round(rand(0,1)) .* (ObjFn(X_k)/(X_{max}))$
32: $RWRC = (1 - 0.5 * round(rand(0,1))$
33: **for** $n = 1$ *to PDim* ***do***
34: $X_{inew} = X_i + (RWRC .* (X_{best} - (X_{mutual} * SaBF_1)))$
35: $X_{knew} = X_k + (RWRC .* (X_{best} - (X_{mutual} * SaBF_2)))$
36: **end for**
37: **if** $(ObjFn(X_{inew}) < ObjFn(X_i)$
38: $X_i = X_{inew}$
39: **end if**
40: **if** $(ObjFn\left(X_{jnew}\right) < ObjFn\left(X_j\right)$
41: $X_j = X_{jnew}$
42: **end if**
43: **if** $(ObjFn(X_{knew}) < ObjFn(X_k)$
44: $X_k = X_{knew}$
45: **end if**
46: // 2nd Phase: Commensalism //
47: Select index j $(1 \leq j \leq Eco\_size; j \neq i)$ randomly
48: $RWRC = (1 - 0.5 * round(rand(0,1))$
49: **for** $k = 1$ *to PDim* **do**
50: $X_{inew} = X_i + RWRC * \left(BestX - X_j\right)$

```
        end for
        if ObjFn(X_inew) < ObjFn(X_i)
          X_i = X_inew
        end if
        //3rd Phase: Parasitism //
        Select index j (1 ≤ j ≤ Eco_size; j ≠ i) randomly
        for k = 1 to PDim do
          if rand(0,1) < rand(0,1)
            X_parasite = X_i
          else
            X_parasite = rand(0,1) * (ULSS[K] − LLSS) + LLSS
          end if
        end for
        if ObjFn(X_parasite) < ObjFn(X_j)
          X_j = X_parasite
        end if
        Update the current population's best solution of the BestX
        //K-means Clustering Phase//
        K-means initialization using the position of the BestX
        Execute K-means clustering using equation (32) for cluster update
      end for
      iter = iter + 1
end while
```

4. Performance Evaluation of Improved SOSK-Means

Forty-two datasets were considered for validating the proposed ISOSK-means algorithm for automatic clustering, of which 24 were real-life datasets and 18 were artificial datasets. All the algorithms were programmed using MATLAB R2018b, running on Windows 10 operating system, installed on a 3.60 GHz Intel® Core i7-7700 processor computer system with 16 GB memory size. For the study, the *eco_size* was set at $40 + 2g$. The algorithms were executed 40 times with 200 iterations in each run. The performance results are presented using the minimum, maximum, average, and standard deviation of the fitness values, as well as the average computational time. The improved SOSK-means algorithm's effectiveness was measured and compared using the following criteria:

- The average best fitness value measured the algorithm's quality of the clustering solutions.
- The performance speed used the algorithm's computational cost and convergence curve.
- The statistical significance difference was over 40 replications.

The results were compared with the standard K-means, the standard SOS algorithm, the existing SOSK-means and other state-of-the-art hybrid and non-hybrid metaheuristic algorithms. The best results among the compared algorithms are presented in bold format. The values of the common control parameters were the same for the K-means, standard SOS algorithm and the SOSK-means algorithms. The control parameters for the competing metaheuristic algorithms obtained from literature in [61] are shown in Table 1.

Table 1. The control parameters for competing metaheuristic algorithms.

DE		FA		IWO		PSO	
Parameter	**Value**	**Parameter**	**Value**	**Parameter**	**Value**	**Parameter**	**Value**
Pop_size	25, 50, 100, 150	*Pop_size*	25, 50, 100, 150	*Pop_size*	25, 50, 100, 150	*Pop_size*	25, 50, 100, 150
Kmin	2	*Kmin*	2	*Kmax*	256	*Kmin*	2
Kmax	256	*Kmax*	256	*MaxGen*	200	*Kmax*	256
MaxIt	200	*MaxGen*	200	s	5	*MaxIt*	200
F	0.8	β_0	2	*e*	2	W_1	1.00
CRmax	1	γ	1	$\sum_1$	0.5	W_2	0.99
CRmin	0.2			$\sum_2$	0.001	τ_1	1.50
						τ_2	2.00

Pop_size: Population Size; *Kmin*: minimum number of clusters; *Kmax*: maximum number of clusters; *MaxIt*: number of iterations; *MaxGen*: Maximum number of Generations; *CR*: Crossover rate; *e*: variance reduction exponent; *s*: maximum number of seeds; W_1: Inertial weight; W_2: Inertial weight damping ratio; β_0: attractiveness; γ : light absorption; τ_1 : personal learning coefficient. τ_2 : global learning coefficient. $\sum_1$: initial value of standard deviation ; $\sum_2$: final value of standard deviation.

4.1. Datasets

During the experiment, 42 datasets were considered. The datasets were grouped into two groups: 18 synthetically generated datasets and 24 real-life datasets. The characteristics of the two categories of datasets are presented in Tables 2 and 3, respectively.

Table 2. The characteristics of the 18 synthetically generated datasets.

Datasets	Number of Data Objects	Dimension of Data	Number of Clusters	Dataset Types	References
A1	3000	2	20	Synthetically generated	[62–64]
A2	5250	2	35	Synthetically generated	[62–64]
A3	7500	2	50	Synthetically generated	[62–64]
Birch1	100,000	2	100	Synthetically generated	[62,64,65]
Birch2	100,000	2	100	Synthetically generated	[62,64,65]
Birch3	100,000	2	100	Synthetically generated	[62,64,65]
Dim002	1351–10,126	2–15	9	Synthetically generated	[62,64,66]
Dim016	1024	16	16	High-dimensional	[62,64,67]
Dim032	1024	32	16	High-dimensional	[62,64,67]
Dim064	1024	64	16	High-dimensional	[62,64,67]
Dim128	1024	128	16	High-dimensional	[62,64,67]
Dim256	1024	256	16	High-dimensional	[62,64,67]
Dim512	1024	512	16	High-dimensional	[62,64,67]
Dim1024	1024	1024	16	High-dimensional	[62,64,67]
S1	5000	2	15	Synthetically generated	[62,64,68]
S2	5000	2	15	Synthetically generated	[62,64,68]
S3	5000	2	15	Synthetically generated	[62,64,68]
S4	5000	2	15	Synthetically generated	[62,64,68]

Table 3. The characteristics of the 24 real-life datasets.

Datasets	Number of Data Objects	Dimension of Data	Number of Clusters	Dataset Types	References
Aggregation	788	2	7	Shape sets	[64,69]
Breast	699	9	2	UCI dataset	[64,70]
Bridge	4096	16	256	Grey-scale image blocks	[64,71]
Compound	399	2	6	Shape sets	[64,72]
D31	3100	2	31	Shape sets	[64,73]
Flame	240	2	2	Shape sets	[64,74]
Glass	214	9	7	UCI dataset	[64,70]
Housec5	34,112	3	256	RGB Image	[64,71]
Housec8	34,112	3	256	RGB Image	[64,71]
Iris	150	4	3	UCI dataset	[64,70]
Jain	373	2	2	Shape sets	[64,75]
Leaves	1600	64	100	UCI dataset	[64,70]
Letter	20,000	16	26	UCI dataset	[64,70]
Joensuu	6014	2	4	Mopsi locations	[64,76]
Finland	13,467	2	4	Mopsi locations	[64,76]
Path-based	300	2	3	Shape sets	[1,64]
R15	600	2	15	Shape sets	[64,77]
Spiral	312	2	3	Shape sets	[64,78]
Thyroid	215	5	2	UCI dataset	[64,70]
T4.8k	8000	2	3	Miscellaneous	[64,79]
Two moons	10,000	2	2	Miscellaneous	[80,81]
Wdbc	569	32	2	UCI dataset	[64,70]
Wine	178	13	3	UCI dataset	[64,70]
Yeast	1484	8	10	UCI dataset	[64,70]

4.1.1. The Synthetic Datasets

The synthetically generated datasets consisted of the A-datasets (three sets), Birch datasets (three sets), DIM (one set of a low dimensional dataset and seven sets of high dimensional datasets), and the S-generated datasets (four sets). The A datasets had an increasing number of clusters, while the Birch datasets had the number of clusters fixed at 100. The DIM datasets were characterized by well-separated clusters, with DIM002 having 9 predetermined clusters (with varying dimensions ranging from 2 to 15), while others had 16 clusters with dimensions ranging from 16 to 1024. The S1–S4 were two-dimensional datasets, characterized by varying spatial data distribution complexity with a uniform cluster of 15.

4.1.2. The Real-Life Datasets

The real-life datasets consisted of the following: eight shape sets (Aggregation, Compound, D31, Flame, Jain, Path-based, R15 and Spiral datasets); ten datasets from the UCI repository (Breast, Glass, Iris, Jain, Leaves, Letter, Thyroid, Wdbc, Wine and Yeast); two Mopsi locations datasets (Joensuu and Finland); two RGB images datasets (Housec5 and Housec8); one Gray-scale image blocks; and two miscellaneous datasets (T4.8k and Two moons). The real-life datasets had varying clusters, ranging from 2 to 256, their di-

mensions ranged between 2 and 64, and the number of data objects in the datasets ranged between 150 and 34,112.

4.2. Experiment 1

In the first set of experiments conducted, the improved SOSK-means was run on 12 datasets, initially used for the SOSK-means. These were Breast, Compound, Flame, Glass, Iris, Jain, Path-based, Spiral, Thyroid, Two moons, Wine, and Yeast datasets. The experimental results are summarized in Table 4, showing the values obtained by the ISOSK-means for each of the 12 datasets. The four decimal place values represented the minimum, the maximum, the mean value over 40 simulations and the standard deviations, which measured the range of values the algorithm converged. From the results obtained, the ISOSK-means returned the best mean values for four of the datasets (Compound, Jain, Thyroid, and two-moons) under the DB index, and eight of the datasets (Breast, Flame, Glass, Iris, Path-based, Spiral, Wine and Yeast) had the best mean values under the CS index. The average computational time for achieving convergence for each dataset by the improved algorithm is shown in Figure 3. Even though the CS index returned the best overall average results for the 12 datasets, this was at the expense of higher computational time.

Table 4. Computational results for improved SOSK-means on 12 real-life datasets.

	DB Index				CS Index			
Datasets	**Min**	**Max**	**Mean**	**Std Dev**	**Min**	**Max**	**Mean**	**Std Dev**
Breast	0.8121	0.8121	0.8121	0.0000	0.5996	0.7209	**0.6639**	0.0391
Compound	0.4947	0.5033	**0.4985**	0.0021	0.4995	0.5032	0.5031	0.0006
Flame	0.7748	0.7770	0.7760	0.0006	0.3846	0.3846	**0.3846**	0.0000
Glass	0.3612	0.7821	0.6547	0.1727	0.0608	0.0608	**0.0608**	0.0000
Iris	0.5913	0.6475	0.6165	0.0132	0.5311	0.5821	**0.5510**	0.0118
Jain	0.6495	0.6522	**0.6507**	0.0008	0.6546	0.6546	0.6546	0.0000
Path-based	0.6533	0.6718	0.6669	0.0050	0.5588	0.6494	**0.6289**	0.0299
Spiral	0.7320	0.7458	0.7386	0.0033	0.5861	0.6862	**0.6618**	0.0329
Thyroid	0.5692	0.6439	**0.6039**	0.0182	0.6409	0.6409	0.6409	0.0000
Two moons	0.6008	0.6033	**0.6021**	0.0006	0.7176	0.7482	0.7243	0.0113
Wine	0.9414	1.0437	1.0053	0.0228	0.6570	0.8625	**0.7614**	0.0571
Yeast	0.3560	0.9902	0.8019	0.1407	0.3897	0.5110	**0.4777**	0.0252
Average	0.6280	0.7394	0.7023	0.0317	**0.5234**	0.5837	**0.5594**	0.0173

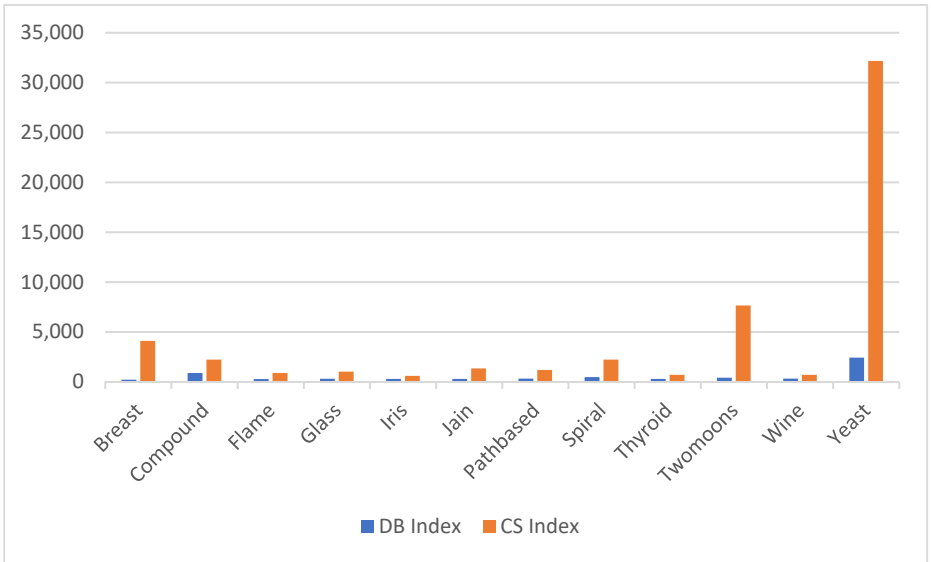


Figure 3. Computational time for the ISOSK-means.

Table 5 summarizes the simulated results for each of the four competing algorithms for the automatic clustering of the 12 real-life datasets. The results present the mean and the standard deviation for each algorithm. The values were obtained for the two cluster validity indices, the DB index, and the CS index.

Table 5. Computation results for the four competing algorithms on 12 real-life datasets.

Datasets	Statistical Measure	DB Index				CS Index			
		SOS	K-Means	SOSK-means	ISOSK-Means	SOS	K-Means	SOSK-Means	ISOSK-Means
Breast	Mean	1.3520	0.8121	0.8121	0.8121	0.9946	1.1019	0.7606	**0.6639**
	Std Dev	0.2858	0.0000	0.0000	0.0000	0.2667	0.0000	0.1217	0.0391
Compound	Mean	0.6924	0.9716	0.5046	**0.4985**	0.5670	1.2887	0.5072	**0.5031**
	Std Dev	0.1481	0.0748	0.0044	0.0021	0.1225	0.1486	0.0155	0.0006
Flame	Mean	0.8234	1.2306	0.7770	**0.7760**	1.2707	1.5806	0.3846	0.3846
	Std Dev	0.0180	0.0059	0.0008	0.0006	0.1006	0.0263	0.0000	0.0000
Glass	Mean	0.8164	1.2208	0.7113	**0.6547**	0.2200	1.4894	0.0608	0.0608
	Std Dev	0.1174	0.1570	0.1217	0.1727	0.2563	0.1904	0.0000	0.0000
Iris	Mean	0.8602	0.9167	0.6346	**0.6165**	0.8585	1.2404	0.5743	**0.5510**
	Std Dev	0.1809	0.0033	0.0188	0.0132	0.1922	0.0092	0.0237	0.0118
Jain	Mean	0.7007	0.8587	0.6518	**0.6507**	0.8196	1.0668	0.6546	0.6546
	Std Dev	0.0274	0.0001	0.0009	0.0008	0.0212	0.0003	0.0000	0.0000
Path-based	Mean	0.7578	0.7696	0.6708	**0.6669**	1.0021	0.9893	0.6511	**0.6289**
	Std Dev	0.0686	0.0066	0.0031	0.0050	0.1708	0.0086	0.0120	0.0299
Spiral	Mean	0.8013	0.9589	0.7437	**0.7386**	1.0818	1.1896	0.6812	**0.6618**
	Std Dev	0.0447	0.0109	0.0045	0.0033	0.2107	0.0053	0.0115	0.0329
Thyroid	Mean	1.0232	1.0298	0.6321	**0.6039**	0.6446	1.7863	0.6409	0.6409
	Std Dev	0.1479	0.2042	0.0316	0.0182	0.0238	0.3602	0.0000	0.0000
Two-moons	Mean	0.6128	0.7948	0.6032	**0.6021**	0.7701	0.9385	0.7498	**0.7243**
	Std Dev	0.0179	0.0000	0.0010	0.0006	0.0281	0.0000	0.0162	0.0113
Wine	Mean	1.1488	1.3053	1.0460	**1.0053**	1.1938	1.4425	0.8422	**0.7614**
	Std Dev	0.1394	0.0022	0.0207	0.0228	0.3318	0.0128	0.0527	0.0571
Yeast	Mean	1.2144	1.7176	0.8496	**0.8019**	0.5594	2.6417	0.5242	**0.4777**
	Std Dev	0.2911	0.1875	0.1588	0.1407	0.2847	0.5950	0.0437	0.0252

As shown in Table 5, the ISOSK-means had the best mean score under the DB validity index in 11 of the 12 datasets, with a tie on the Breast dataset for three of the algorithms: ISOSK-means, SOSK-means and K-means. In the same vein, the ISOSK-means recorded the best mean scores under the CS validity index for the ISOSK-means in eight datasets with ties for two of the algorithms: SOSK-means and ISOSK-means, in four of the datasets (Flame, Glass, Jain, and Thyroid). The performance analysis of the four competing algorithms is shown in Figures 4 and 5 for each cluster validity index. These results indicated that the ISOSK-means algorithm performed better than the other three competing algorithms.

The average execution times of the SOS, SOSK-means and ISOSK-means algorithms for 200 generations were presented separately for each validity index, as shown in Figures 6 and 7. As expected, being a non-hybrid algorithm, the SOS recorded the lowest computational time under the two validity indices. Under the DB validity index, the ISOSK-means recorded a shorter average execution time than the SOSK-means in nine datasets. This indicated that the ISOSK-means required less execution time to achieve convergence. Under the CS validity index, the ISOSK-means recorded lower execution time in six datasets. Out of the remaining six, only three datasets showed a considerable difference in execution time for the two algorithms in favor of the SOSK-means. Since the CS index usually requires higher computational time, this result showed that the ISOSK-means substantially outperformed SOSK-means in terms of computational cost.

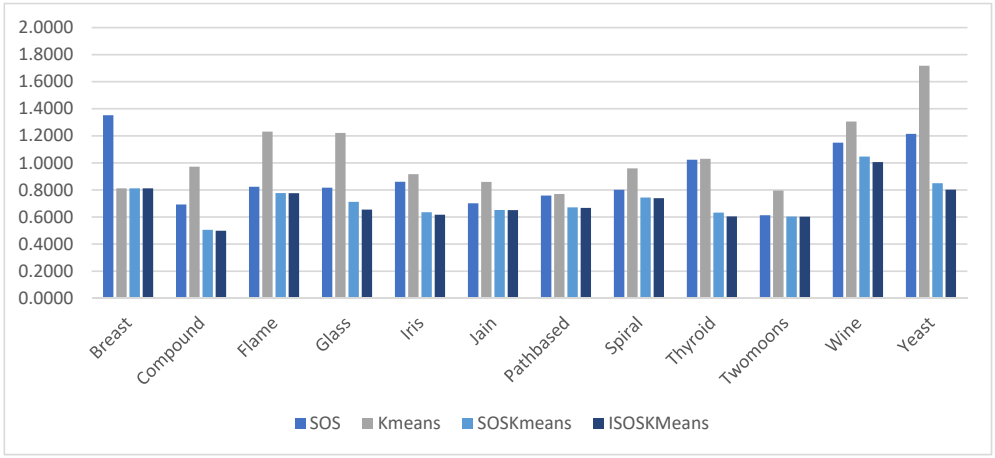


Figure 4. Performance analysis of four competing algorithms, ISOSK-means, SOSK-means, SOS and K-means, under the DB index.

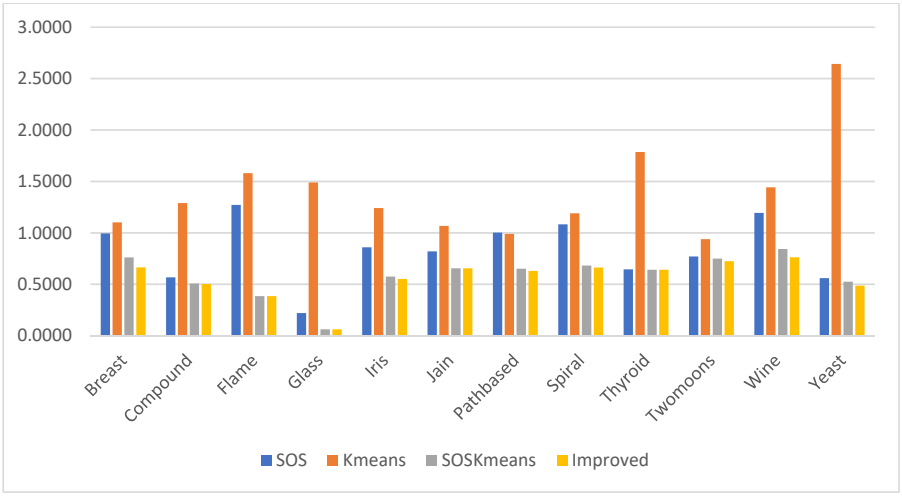


Figure 5. Performance analysis of four competing algorithms, ISOSK-means, SOSK-means, SOS and K-means, under the CS index.

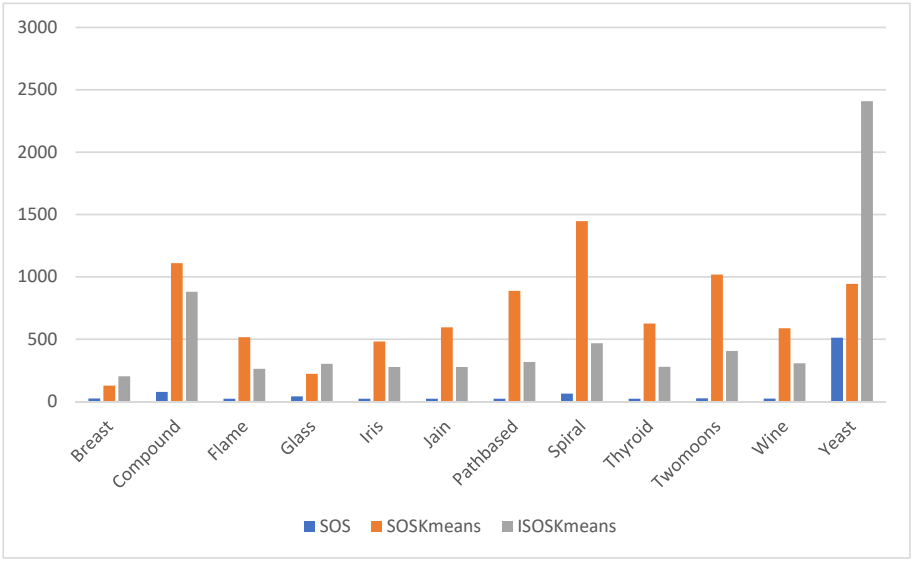


Figure 6. Analysis of computational times of SOS, SOSK-means and ISOSK-means under the DB index.

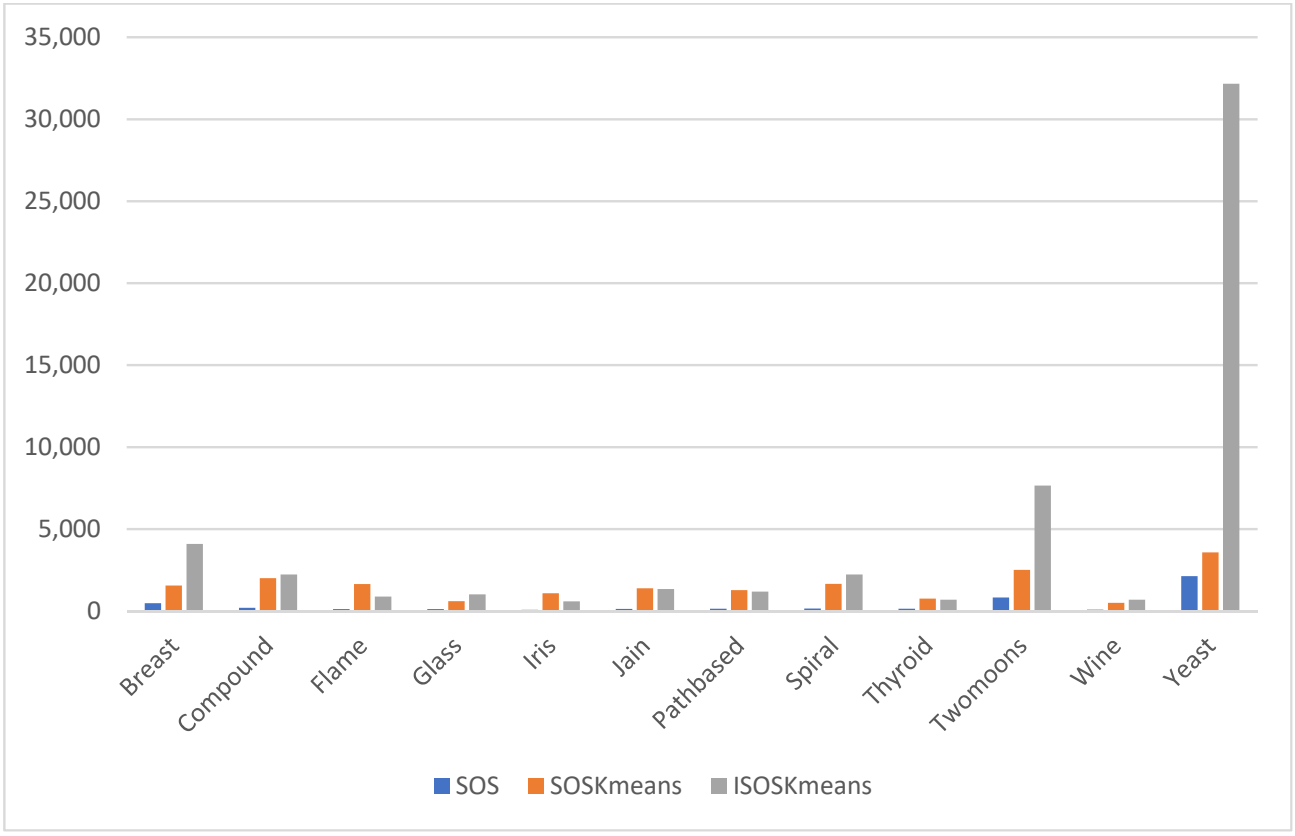


Figure 7. Analysis of computational times of SOS, SOSK-means and ISOSK-means under the CS index.

The convergence curve for the three algorithms, ISOSK-means, SOSK-means and SOS, are illustrated in Figures 8 and 9 for the DB index and CS index, respectively. A rapid and smooth declining curve reflected a superior performance. In all 12 datasets, the ISOSK-means recorded the best performance with rapidly declining smooth curves under the two validity indices.

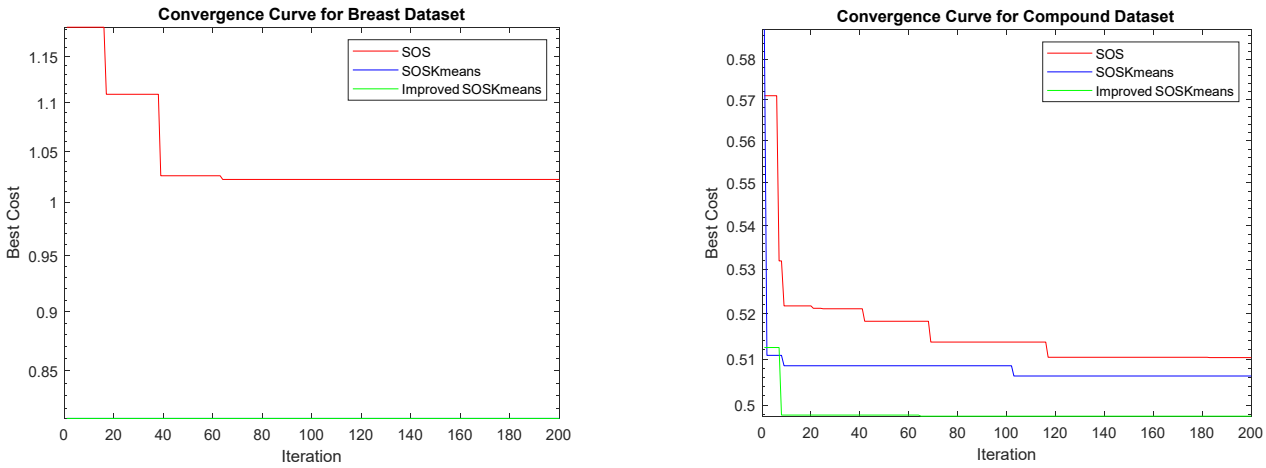
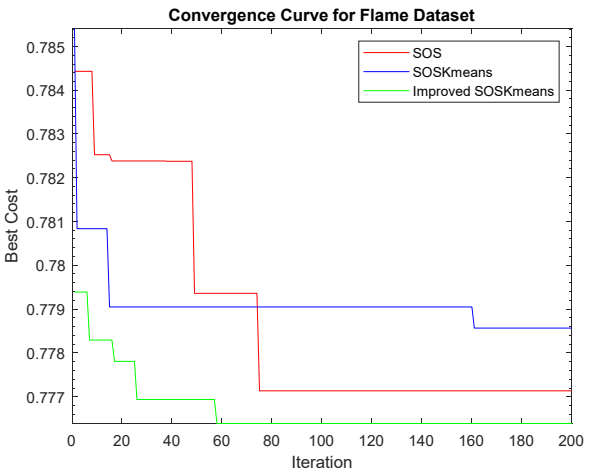
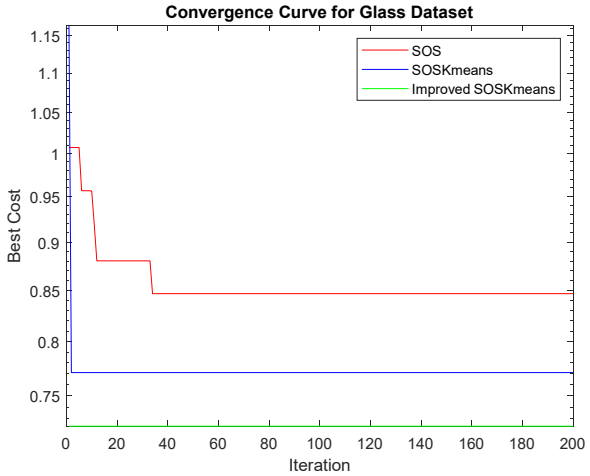
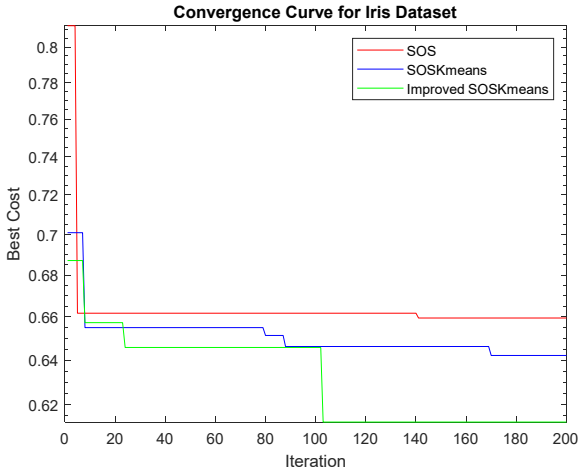
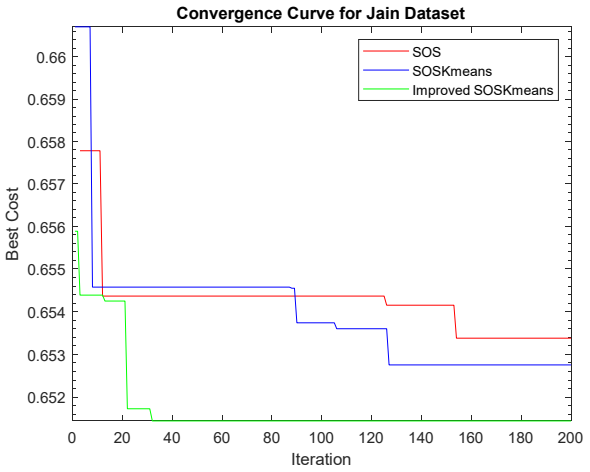


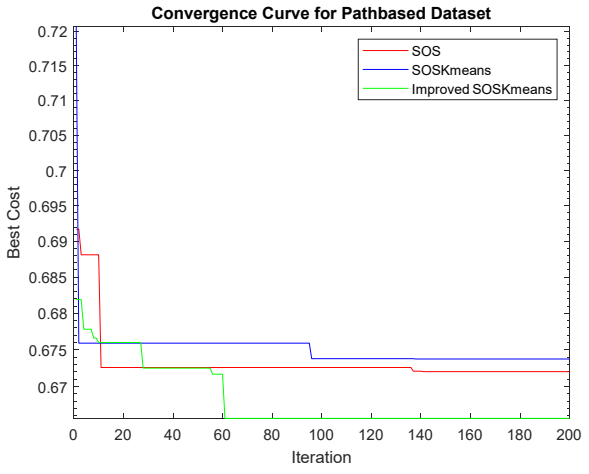
Figure 8. *Cont.*











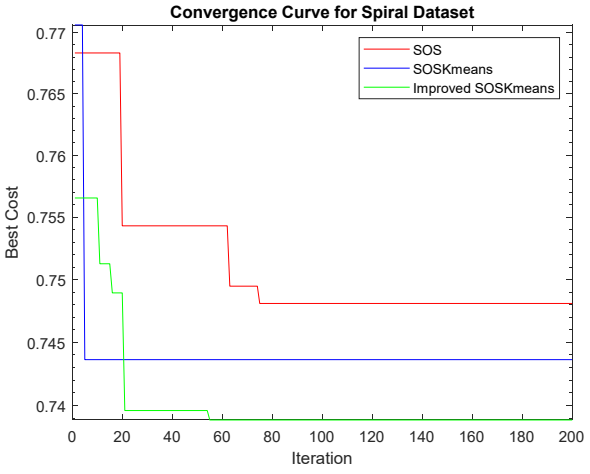


Figure 8. *Cont.*

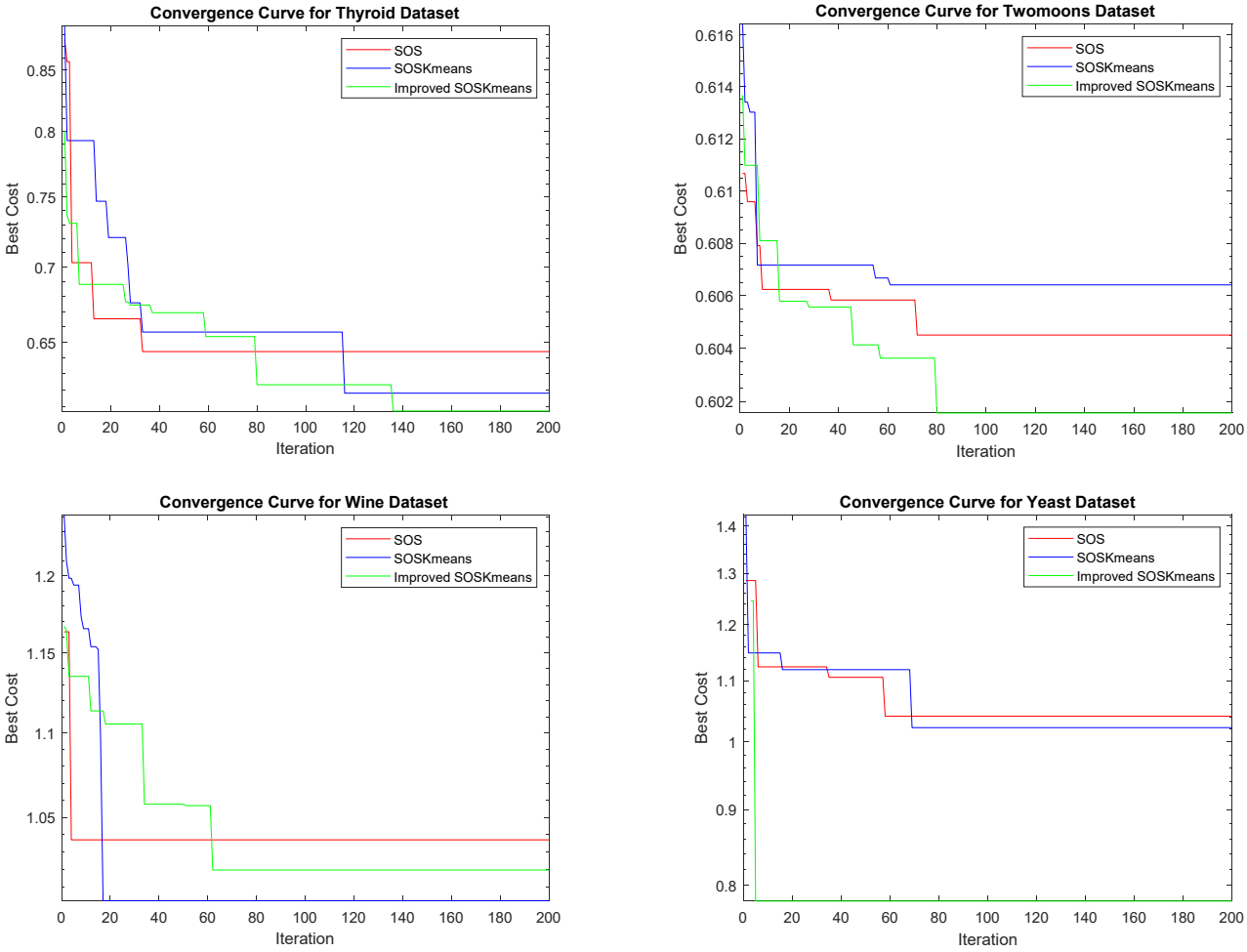


Figure 8. Convergence curves for the 12 datasets under DB index.

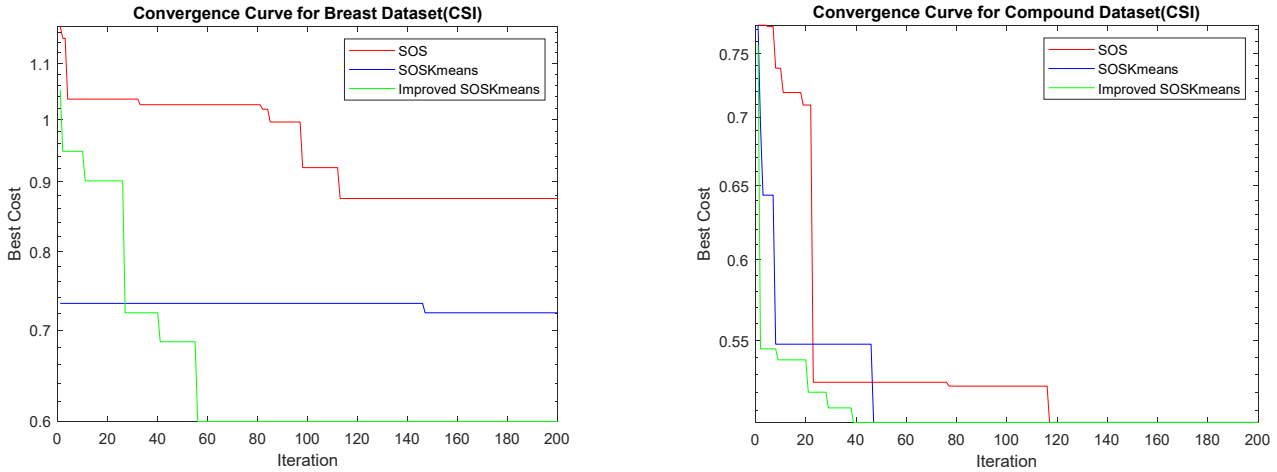


Figure 9. *Cont.*

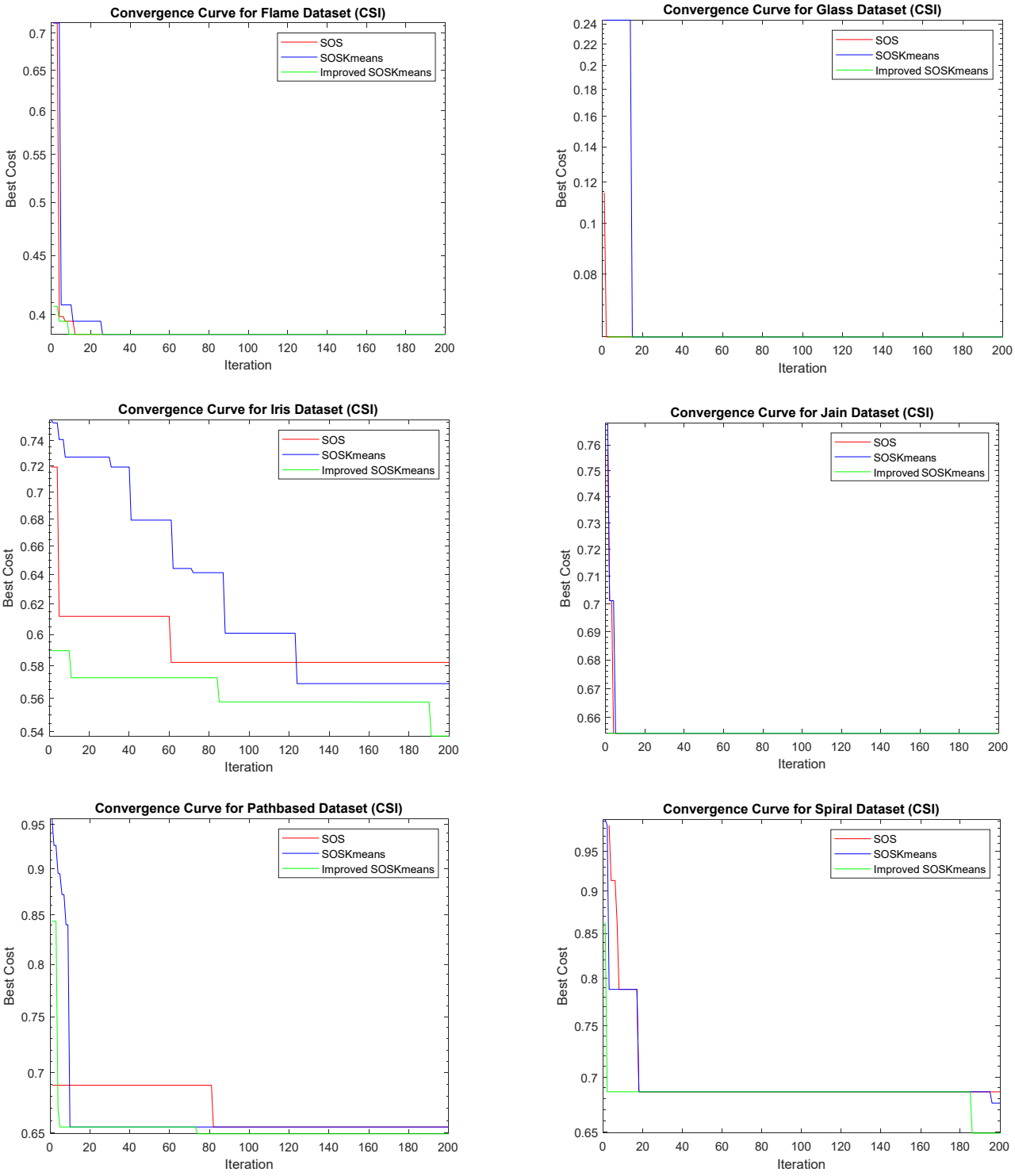


Figure 9. *Cont.*

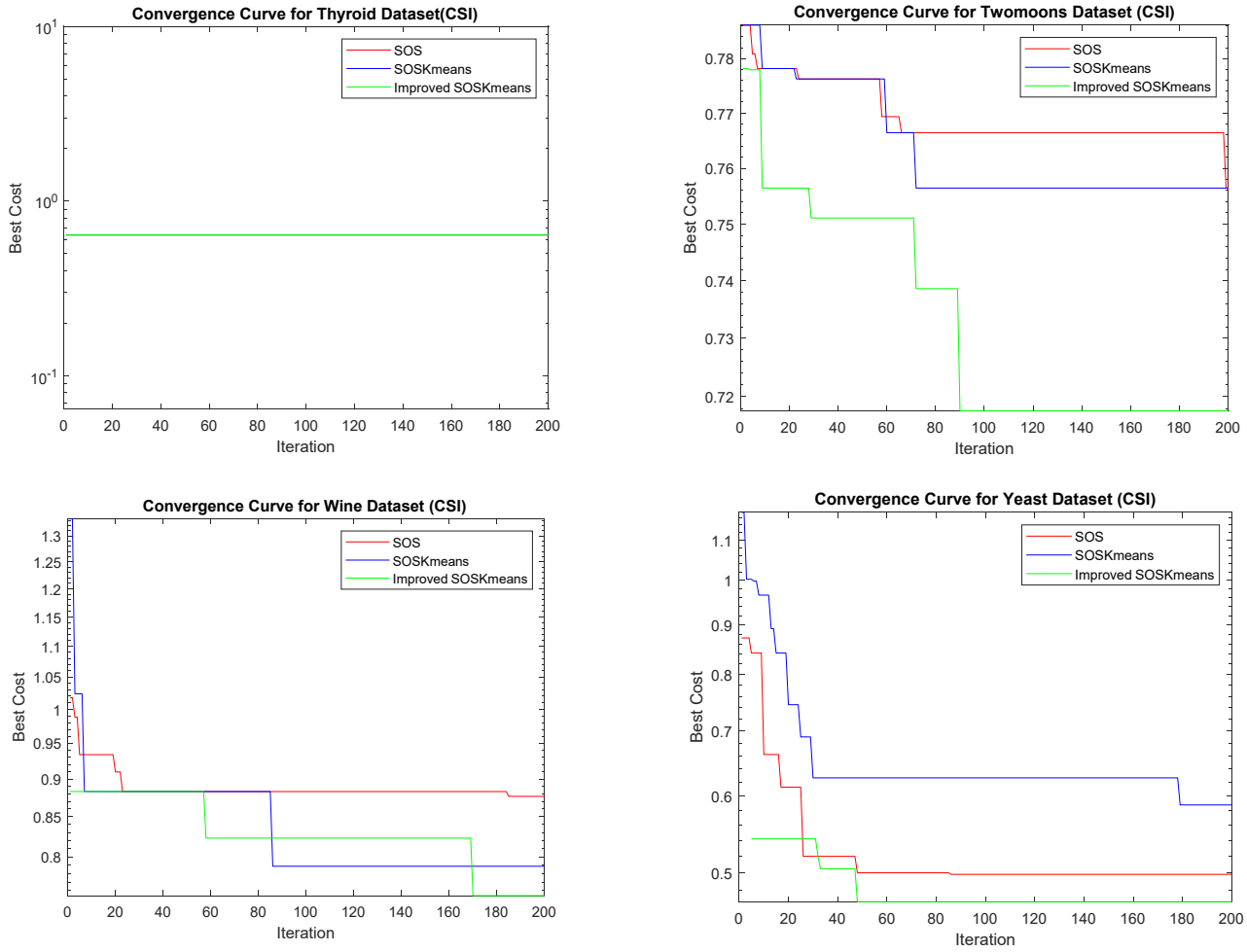


Figure 9. Convergence curves for the 12 datasets under the CS index.

4.3. Experiment 2

In Experiment 2, an extensive performance evaluation of the ISOSK-means algorithm on 18 synthetic datasets was demonstrated. Table 6 shows the computational results from the experiment under the DB validity index, showing the best, the worst, the mean, and the standard deviation scores. The results of the ISOSK-means were compared with other existing hybrid algorithms from the literature [61] for the same problem in Table 7. The ISOSK-means recorded a better overall mean value for the 18 synthetic datasets, compared with the other hybrid metaheuristic algorithms. This showed that the mean performance of the improved hybrid algorithm was superior, compared with the competing hybrid metaheuristic algorithms, in solving the automatic clustering problem.

Table 6. Computational results of ISOSK-means on 18 synthetic datasets.

	ISOSK-Means (DB Index)			
Datasets	**Min**	**Max**	**Mean**	**Std Dev**
A1	0.5905	0.5918	0.5911	0.0003
A2	0.6777	0.6786	0.6781	0.0002
A3	0.7921	0.7965	0.7945	0.0011
Birch1	0.8020	0.8042	0.8030	0.0006
Birch2	0.5070	0.5073	0.5071	0.0001
Birch3	0.7161	0.7179	0.7168	0.0004
Dim002	0.5873	0.6676	0.6384	0.0222
Dim016	0.7201	1.2869	1.1069	0.1870
Dim032	0.8293	1.4705	1.2402	0.2463
Dim064	0.8474	1.5901	1.0390	0.2524
Dim128	0.8507	1.7100	1.0045	0.2008
Dim256	0.9056	1.8192	1.0785	0.3065
Dim512	0.2054	1.0135	0.9247	1.0135
Dim1024	0.9060	1.9655	1.1332	0.3795
S1	0.7752	0.7785	0.7770	0.0009
S2	0.7394	0.7430	0.7412	0.0008
S3	0.7119	0.7139	0.7126	0.0005
S4	0.7705	0.7741	0.7723	0.0009
Average	0.7186	1.0350	0.8477	0.1452

Table 7. Performance comparison of ISOSK-means with other hybrid metaheuristic algorithms from literature on 18 synthetic datasets.

	ISOSK-Means		PSODE		FADE		IWODE	
	Mean	**Std Dev**	**Mean**	**Std Dev**	**Mean**	**Std Dev**	**Mean**	**Std Dev**
A1	**0.5911**	0.0003	0.5949	0.0086	0.6171	0.0347	0.6525	0.0621
A2	**0.6781**	0.0002	0.6912	0.0161	0.6976	0.0215	0.7296	0.0391
A3	0.7945	0.0011	0.7106	0.0176	**0.7085**	0.0332	0.7527	0.0319
Birch1	0.8030	0.0006	0.7256	0.0276	**0.7232**	0.0257	0.76923	0.0279
Birch2	**0.5071**	0.0001	**0.507**	0.0002	0.5155	0.0235	0.5176	0.0084
Birch3	0.7168	0.0004	0.7074	0.0151	**0.7012**	0.0191	0.757	0.0247
Dim002	0.6384	0.0222	**0.5975**	0.0445	0.628	0.0607	0.6705	0.0432
Dim016	1.1069	0.1870	1.0183	0.1285	1.0413	0.0512	1.4336	0.0456
Dim032	1.2402	0.2463	1.1142	0.0915	**1.0727**	0.0894	1.5731	0.0574
Dim064	**1.0390**	0.2524	1.1918	0.1769	1.1003	0.0925	1.7015	0.0415
Dim128	**1.0045**	0.2008	1.3363	0.1425	1.198	0.1173	1.7773	0.0477
Dim256	**1.0785**	0.3065	1.5051	0.174	1.2938	0.1308	1.8603	0.0309
Dim512	**0.9247**	1.0135	1.6827	0.0633	1.3529	0.1463	1.9311	0.0424
Dim1024	**1.1332**	0.3795	1.7644	0.0112	1.4759	0.12	1.9654	0.0261
S1	0.7770	0.0009	**0.6739**	0.0351	0.6756	0.027	0.7501	0.028
S2	0.7412	0.0008	**0.6844**	0.028	0.6939	0.0345	0.7556	0.019
S3	0.7126	0.0005	0.7106	0.0199	**0.7072**	0.0181	0.7559	0.0317
S4	0.7723	0.0009	**0.7299**	0.0162	0.7356	0.0226	0.7896	0.0303
Average	**0.8477**	0.1452	0.9414	0.0565	0.8855	0.0593	1.1190	0.0354

4.4. Experiment 3

In Experiment 3, the performance of the ISOSK-means algorithm on 23 real life datasets were evaluated. The computational results from the experiment under the DB validity index are shown in Table 8, showing the minimum, the maximum, the average, and the standard deviation scores for the ISOSK-means algorithm. The results were compared with other existing hybrid algorithms from the literature [61] for the same problem, as presented in Table 9. The ISOSK-means recorded a better mean value for the 23 real-life datasets than the other hybrid metaheuristic algorithms. This indicated that the improved hybrid algorithm performed better, on average, in solving the automatic clustering problem, compared with

the competing hybrid metaheuristic algorithms. However, it is worth noting that FADE had the best clustering result in nine of the datasets. Nevertheless, the ISOSK-means recorded the least standard deviation in 18 datasets, with the least mean standard deviation score in all, showing that it produced more compact clusters than the competing hybrid metaheuristic algorithms.

Table 8. Computational results of ISOSK-means on 23 real-life datasets.

	ISOSK-Means (DB Index)			
Datasets	**Min**	**Max**	**Mean**	**Std Dev**
Aggregation	0.7229	0.7297	0.7262	0.0014
Breast	0.8121	0.8121	0.8121	0.0000
Bridge	0.6455	0.6474	0.6464	0.0009
Compound	0.4947	0.5033	0.4985	0.0021
D31	0.8016	0.8263	0.8125	0.0070
Flame	0.7748	0.7770	0.7760	0.0006
Glass	0.3612	0.7821	0.6547	0.1727
Housec5	0.5158	0.5650	0.5377	0.0123
Housec8	0.4919	0.5305	0.5158	0.0107
Iris	0.5913	0.6475	0.6165	0.0132
Jain	0.6495	0.6522	0.6507	0.0008
Leaves	0.7207	0.9591	0.7618	0.0745
Letter	0.9683	1.0545	1.0242	0.0188
Joensuu	0.5011	0.5147	0.5073	0.0038
Finland	0.4409	0.4453	0.4427	0.0011
Path-based	0.6533	0.6718	0.6669	0.0050
R15	0.7106	0.7771	0.7400	0.0171
Spiral	0.7320	0.7458	0.7386	0.0033
T4.8k	0.0227	0.0227	0.0227	0.0000
Thyroid	0.5692	0.6439	0.6039	0.0182
Wdbc	0.0508	0.0508	0.0508	0.0000
Wine	0.9414	1.0437	1.0053	0.0228
Yeast	0.7084	0.8976	0.7489	0.0682
Average	0.6035	0.6652	0.6331	0.0198

Table 9. Performance comparison of ISOSK-means with other hybrid metaheuristic algorithms on 23 real-life datasets.

	ISOSK-Means		PSODE		FADE		IWODE	
	Mean	**Std Dev**	**Mean**	**Std Dev**	**Mean**	**Std Dev**	**Mean**	**Std Dev**
Aggregation	0.7262	0.0014	0.5958	0.0391	**0.5948**	0.0424	0.6015	0.051
Breast	0.8121	0.0000	0.7256	0.1115	0.9378	0.2051	0.7965	0.1177
Bridge	0.6464	0.0009	0.7141	0.1007	**0.6405**	0.0709	1.1397	0.0666
Compound	0.4985	0.0021	**0.4983**	0.0092	0.5273	0.0576	0.50004	0.0312
D31	0.8125	0.0070	0.8021	0.0407	**0.7788**	0.0376	0.7972	0.0514
Flame	0.7760	0.0006	4.4069	4.6725	0.6795	0.0477	**0.6749**	0.0391
Glass	0.6547	0.1727	0.5966	0.0672	0.5971	0.0996	**0.5562**	0.1384
Housec5	**0.5377**	0.0123	2.2408	4.0861	0.5467	0.0287	0.6865	0.0229
Housec8	0.4919	0.5305	0.5022	0.0315	**0.4707**	0.0383	0.6344	0.0408
Iris	0.6165	0.0132	**0.5811**	0.021	0.585	0.035	0.8384	0.0916
Jain	0.6507	0.0008	**0.6399**	0.015	0.6451	0.0062	0.65	0.0058
Leaves	0.7618	0.0745	0.758	0.1514	**0.7483**	0.1539	1.5116	0.0396
Letter	0.9683	1.0545	0.9121	0.0628	**0.8665**	0.0663	1.2057	0.0571
Joensuu	0.5073	0.0038	0.5094	0.0098	0.51	0.0327	**0.4972**	0.0041
Finland	**0.4427**	0.0011	0.4465	0.006	0.4686	0.0547	0.4864	0.0371
Path-based	0.6669	0.0050	**0.6526**	0.0166	0.6561	0.0178	0.6567	0.0156
R15	0.7400	0.0171	0.6423	0.0722	0.6399	0.0984	**0.5996**	0.0894
Spiral	0.7386	0.0033	0.7441	0.0136	**0.7373**	0.0156	0.7758	0.0143
T4.8k	**0.0227**	0.0000	**0.0227**	0	0.04234	0.0882	0.0928	0.0515
Thyroid	0.6039	0.0182	0.5024	0.0196	**0.4955**	0.0241	0.9893	0.1077
Wdbc	**0.0508**	0.0000	**0.0508**	0	**0.0508**	0	0.0814	0.0263
Wine	1.0053	0.0228	0.8891	0.0601	**0.8648**	0.0973	1.2312	0.0525
Yeast	0.7489	0.0682	0.7193	0.0677	0.6375	0.1344	**0.5949**	0.1144
Average	**0.6331**	0.0198	0.8805	0.2608	0.7234	0.0615	0.8961	0.0464

4.5. Experiment 4

In this experiment, the ISOSK-means results were compared with other existing non-hybrid algorithms (DE, PSO, FA, IWO) from the literature [60] for the same problem, and the results are presented in Table 10. The ISOSK-means recorded the best mean scores in four datasets: A1, A2, Birch2 and Housec5. The FA recorded the best mean performance, followed by the proposed ISOSK-means algorithm. FA and ISOSK-means recorded better average performance scores than DE, PSO and IWO. This showed that the improved hybrid algorithm demonstrated a high competing capability with other metaheuristic algorithms in automatic clustering of high dimensional datasets.

Table 10. Performance comparison of ISOSK-means with other non-hybrid metaheuristic algorithms on high-dimensional datasets.

High-Dimensional Dataset	ISOSK-Means		DE		PSO		FA		IWO	
	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
A1	**0.5911**	0.0003	0.6016	0.0116	0.6662	0.0042	0.6089	0.0341	0.6308	0.0265
A2	**0.6781**	0.0002	0.7081	0.0348	0.7134	0.0284	0.6842	0.0143	0.7483	0.0468
A3	0.7945	0.0011	0.7349	0.0184	0.7279	0.0287	**0.6844**	0.0241	0.7545	0.0339
Birch 1	0.8030	0.0006	0.748	0.0171	0.7528	0.0169	**0.7091**	0.0203	0.7644	0.0211
Birch 2	**0.5071**	0.0001	0.5086	0.0016	0.5876	0.0358	0.5087	0.0078	0.5168	0.0052
Birch 3	0.7168	0.0004	0.7488	0.0297	0.7213	0.0306	**0.6974**	0.0127	0.7568	0.0248
D31	0.8125	0.0070	0.8757	0.0302	**0.763**	0.0514	0.7808	0.0436	0.7896	0.0312
Housec5	**0.5377**	0.0123	0.619	0.0412	0.7456	0.0679	0.5642	0.029	0.7034	0.043
Housec8	0.4919	0.5305	0.5245	0.0139	0.6418	0.0858	**0.4584**	0.0111	0.6206	0.0546
Leaves	0.7618	0.0745	1.1345	0.0798	1.0335	0.2854	**0.6036**	0.0587	1.5132	0.049
Letter	0.9683	1.0545	0.9852	0.0303	1.0354	0.1084	**0.8071**	0.0246	1.2245	0.0416
Average	0.6966	0.1529	0.7444	0.0281	0.7626	0.0676	**0.6461**	0.0255	0.8203	0.0343

4.6. Statistical Analysis

The statistical analysis experiment involved using a nonparametric statistical analysis technique to further validate the algorithms' computational results. The Friedman's nonparametric test was conducted to draw a statistically verified meaningful conclusion from the reported performances. The reports for the computed Friedman's mean rank on the four competing algorithms (the ISOSK-means, SOSK-means, SOS, and K-means) are presented in Table 11. The results showed that the ISOSK-means had the best performance, recording 1.13 as its minimum rank value and 2.00 as its maximum rank value, under the DB index, with a 1.15 minimum rank value and 1.99 maximum rank value under the CS index.

Table 11. Friedman mean rank for ISOSK-means, SOSK-means, SOS, and K-means.

Datasets	DB Index				CS Index			
	SOS	K-Means	SOSK-Means	ISOSK-Means	SOS	K-Means	SOSK-Means	ISOSK-Means
Breast	4	2	2	2	3.13	3.45	2.03	1.4
Compound	2.9	3.95	1.98	1.18	2.34	4	1.91	1.75
Flame	3	4	1.83	1.18	3	4	1.5	1.5
Glass	2.42	4	1.94	1.64	2.68	4	1.6	1.6
Iris	3.3	3.48	1.98	1.25	2.85	4	1.98	1.18
Jain	3	4	1.88	1.13	3	4	1.5	1.5
Path-based	3.18	3.45	2.08	1.3	3.63	3.23	1.94	1.21
Spiral	2.8	4	1.9	1.3	3.18	3.58	1.79	1.46
Thyroid	3.4	3.53	1.9	1.18	2.03	4	1.99	1.99
Two-moons	2.61	4	2.14	1.25	2.63	4	2.16	1.21
Wine	3	3.7	2.17	1.13	3.09	3.63	2.14	1.15
Yeast	3	3.88	1.73	1.4	1.63	4	2.69	1.69

For further verification of the specific significant differences among the competing algorithms, a post hoc test on the Friedman mean-rank test results, using the Wilcoxon signed-rank test, was conducted. The Wilcoxon signed-rank test presents a set of p values which statistically measures whether there is a significant difference between the competing algorithms at a significance level of 0.05.

Table 12 shows the significant differences between ISOSK-means and the other competing algorithms under the DB and CS cluster validity indices. Out of 12 datasets, ISOSK-means recorded about 92%, 100%, and 83% significant differences in its performance from K-means, SOS, and SOSK-means under the DB index, and with a significant difference of 100%, 83% and 58%, respectively, under the CS index. This implied that the modification made to the existing SOSK-means algorithm contributed to the performance enhancement of the existing hybrid algorithm in solving cluster analysis problems.

Table 12. Wilcoxon signed-rank test for ISOSK-means, SOSK-means, SOS, and K-means.

Cluster Validity Index	DB Index			CS Index		
Datasets	**ISOSK-Means vs. SOS**	**ISOSK-Means vs. K-Means**	**ISOSK-Means vs. SOSK-Means**	**ISOSK-Means vs. SOS**	**ISOSK-Means vs. K-Means**	**ISOSK-Means vs. SOSK-Means**
Breast	0.0000	1.0000	1.0000	0.0000	0.0000	0.0010
Compound	0.0000	0.0000	0.0000	0.0010	0.0000	0.0430
Flame	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000
Glass	0.0000	0.0000	0.0740	0.0000	0.0000	1.0000
Iris	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Jain	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000
Pathbased	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Spiral	0.0000	0.0000	0.0000	0.0000	0.0000	0.0040
Thyroid	0.0000	0.0000	0.0000	0.3170	0.0000	1.0000
Two moons	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Wine	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Yeast	0.0000	0.0000	0.0000	0.2760	0.0000	0.0000

5. Conclusions and Future Directions

In this study, an improved hybrid ISOSK-means metaheuristic algorithm is presented. Several improvements were incorporated into each of the two classical algorithms combined in the hybridization. In the initialization phase of the SOS algorithm, the population size was constructed using $40 + 2g$ for a well-distributed initial population of solutions that scaled well with the data size. This resulted in a convergence rate increase and lower computational time. For performance upgrade, a three-part mutualism phase, with a random weighted reflection coefficient, was also integrated into the SOS algorithm with a random probability for determining whether an organism would be engaged in the three-part mutualism. To improve the quality of the clustering solution, the benefit factor was modified, by incorporating consideration for fitness value relativity with respect to the maximum fitness value.

The misleading effects of outliers in the dataset were addressed by the improvement incorporated into the K-means phase of the improved hybrid algorithm. A method for detecting and excluding putative outliers during the centroid update phase of the classical algorithm was added. The algorithm uses a point-to-centroid distance threshold for the centroid update, instead of using the means of data points. The point-to-centroid distance threshold uses the median absolute deviation, which is considered to be a robust measure of statistical dispersion and is known to be more resilient to outliers. This ensured that outliers were excluded from contributing to minimizing the mean square error in the K-means. This resulted in a more compact cluster output.

The improved hybrid algorithm was evaluated on 42 datasets (18 synthetic and 24 real-life) with varying characteristics, such as being high dimensional datasets, low dimensional datasets, synthetically generated datasets, image datasets, shape datasets, and location datasets, with varied dataset sizes and clusters. The performance of the improved

hybrid algorithm was compared with the standard hybrid and non-hybrid algorithms. The performance was also compared with the two standard algorithms SOS and K-means on 12 real life datasets. A Friedman means rank test was applied to analyze the significant difference between the ISOSK-means and the competing algorithms. A pairwise post hoc Wilcoxon signed-rank sum test was also performed to highlight the performance of the ISOSK-means in comparison with the competing algorithms. The ISOSK-means algorithm outperformed the three algorithms with lower computational time and higher convergence rate, as reflected in the convergence curve for the competing algorithms.

The ISOSK-means clustering results were compared with four non-hybrid metaheuristic algorithms and three hybrid metaheuristic algorithms from the literature. The ISOSK-means had fair competitiveness, in terms of clustering performance measured using the DB validity index, on 42 datasets. The ISOSK-means recorded the lowest standard deviation score for most datasets, compared with the competing algorithms.

For future research, the K-means phase of the hybridized algorithm could be improved to efficiently manage large datasets and, thereby, reduce the algorithm's computational complexity. Other suggested improvements to the SOS algorithm from literature could also be incorporated into the algorithm, or be used as a replacement to the current SOS phase of the ISOSK-means, for further performance enhancement, while reducing the algorithm run time. The real-life application of the proposed improved hybrid algorithm is another area of research that could be exploited. Using other cluster validity indices in implementing this algorithm would also be an interesting area for future research.

Author Contributions: All authors contributed to the conception and design of the research work. Material preparation, experiments, and analysis were performed by A.M.I. and A.E.E. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare that they have no known competing financial interest or personal relationship that could have appeared to influence the work reported in this article.

References

1. José-García, A.; Gómez-Flores, W. Automatic clustering using nature-inspired metaheuristics: A survey. *Appl. Soft Comput.* **2016**, *41*, 192–213. [CrossRef]
2. Ikotun, A.M.; Almutari, M.S.; Ezugwu, A.E. K-Means-Based Nature-Inspired Metaheuristic Algorithms for Automatic Data Clustering Problems: Recent Advances and Future Directions. *Appl. Sci.* **2021**, *11*, 11246. [CrossRef]
3. Olukanmi, P.O.; Twala, B. K-means-sharp: Modified centroid update for outlier-robust k-means clustering. In Proceedings of the 2017 Pattern Recognition Association of South Africa and Robotics and Mechatronics International Conference (PRASA-RobMech), Bloemfontein, South Africa, 29 November–1 December 2017; pp. 14–19. [CrossRef]
4. Ikotun, A.M.; Ezugwu, A.E.; Abualigah, L.; Abuhaija, B.; Heming, J. K-means Clustering Algorithms: A Comprehensive Review, Variants Analysis, and Advances in the Era of Big Data. *Inf. Sci.* **2022**, *622*, 178–210. [CrossRef]
5. Knorr, E.M.; Ng, R.T.; Tucakov, V. Distance-based outliers: Algorithms and applications. *VLDB J.* **2000**, *8*, 237–253. [CrossRef]
6. Chawla, S.; Gionis, A. k-means–: A unified approach to clustering and outlier detection. In Proceedings of the 2013 SIAM International Conference on Data Mining, Austin, TX, USA, 2–4 May 2013; pp. 189–197. [CrossRef]
7. Olukanmi, P.; Nelwamondo, F.; Marwala, T.; Twala, B. Automatic detection of outliers and the number of clusters in k-means clustering via Chebyshev-type inequalities. *Neural Comput. Appl.* **2022**, *34*, 5939–5958. [CrossRef]
8. Cheng, M.-Y.; Prayogo, D. Symbiotic Organisms Search: A new metaheuristic optimization algorithm. *Comput. Struct.* **2014**, *139*, 98–112. [CrossRef]
9. Chakraborty, S.; Nama, S.; Saha, A.K. An improved symbiotic organisms search algorithm for higher dimensional optimization problems. *Knowl.-Based Syst.* **2021**, *236*, 107779. [CrossRef]
10. Panda, A.; Pani, S. A symbiotic organisms search algorithm with adaptive penalty function to solve multi-objective constrained optimization problems. *Appl. Soft Comput.* **2016**, *46*, 344–360. [CrossRef]

11. Cheng, M.-Y.; Prayogo, D.; Tran, D.-H. Optimizing Multiple-Resources Leveling in Multiple Projects Using Discrete Symbiotic Organisms Search. *J. Comput. Civ. Eng.* **2016**, *30*, 04015036. [CrossRef]
12. Kawambwa, S.; Hamisi, N.; Mafole, P.; Kundaeli, H. A cloud model based symbiotic organism search algorithm for DG allocation in radial distribution network. *Evol. Intell.* **2021**, *15*, 545–562. [CrossRef]
13. Liu, D.; Li, H.; Wang, H.; Qi, C.; Rose, T. Discrete symbiotic organisms search method for solving large-scale time-cost trade-off problem in construction scheduling. *Expert Syst. Appl.* **2020**, *148*, 113230. [CrossRef]
14. Cheng, M.-Y.; Cao, M.-T.; Herianto, J.G. Symbiotic organisms search-optimized deep learning technique for mapping construction cash flow considering complexity of project. *Chaos Solitons Fractals* **2020**, *138*, 109869. [CrossRef]
15. Abdullahi, M.; Ngadi, A.; Abdulhamid, S.M. Symbiotic organism search optimization based task scheduling in cloud computing environment. *Future Gener. Comput. Syst.* **2016**, *56*, 640–650. [CrossRef]
16. Ezugwu, A.E.-S.; Adewumi, A.O.; Frîncu, M.E. Simulated annealing based symbiotic organisms search optimization algorithm for traveling salesman problem. *Expert Syst. Appl.* **2017**, *77*, 189–210. [CrossRef]
17. Tejani, G.; Savsani, V.J.; Patel, V. Adaptive symbiotic organisms search (SOS) algorithm for structural design optimization. *J. Comput. Des. Eng.* **2016**, *3*, 226–249. [CrossRef]
18. Abdullahi, M.; Ngadi, A. Hybrid symbiotic organisms search optimization algorithm for scheduling of tasks on cloud computing environment. *PLoS ONE* **2016**, *11*, e0158229. [CrossRef]
19. Mohammadzadeh, H.; Gharehchopogh, F.S. Feature selection with binary symbiotic organisms search algorithm for email spam detection. *Int. J. Inf. Technol. Decis. Mak.* **2021**, *20*, 469–515. [CrossRef]
20. Boushaki, S.I.; Bendjeghaba, O.; Kamel, N. Biomedical document clustering based on accelerated symbiotic organisms search algorithm. *Int. J. Swarm Intell. Res.* **2021**, *12*, 169–185. [CrossRef]
21. Zhou, Y.; Wu, H.; Luo, Q.; Abdel-Baset, M. Automatic data clustering using nature-inspired symbiotic organism search algorithm. *Knowl.-Based Syst.* **2019**, *163*, 546–557. [CrossRef]
22. Chen, J.; Zhang, Y.; Wu, L.; You, T.; Ning, X. An adaptive clustering-based algorithm for automatic path planning of heterogeneous UAVs. *IEEE Trans. Intell. Transp. Syst.* **2021**, *23*, 16842–16853. [CrossRef]
23. Zainal, N.A.; Zamli, K.Z.; Din, F. A modified symbiotic organism search algorithm with lévy flight for software module clustering problem. In Proceedings of the ECCE2019—5th International Conference on Electrical, Control & Computer Engineering, Kuantan, Malaysia, 29 July 2019; pp. 219–229. [CrossRef]
24. Rajah, V.; Ezugwu, A.E. Hybrid Symbiotic Organism Search algorithms for Automatic Data Clustering. In Proceedings of the 2020 Conference on Information Communications Technology and Society (ICTAS), Durban, South Africa, 11–12 March 2020; pp. 1–9. [CrossRef]
25. Yang, C.-L.; Sutrisno, H. A clustering-based symbiotic organisms search algorithm for high-dimensional optimization problems. *Appl. Soft Comput.* **2020**, *97*, 106722. [CrossRef]
26. Ikotun, A.M.; Ezugwu, A.E. Boosting k-means clustering with symbiotic organisms search for automatic clustering problems. *PLoS ONE* **2022**, *17*, e0272861. [CrossRef] [PubMed]
27. Nama, S.; Saha, A.K.; Ghosh, S. Improved symbiotic organisms search algorithm for solving unconstrained function optimization. *Decis. Sci. Lett.* **2016**, *5*, 361–380. [CrossRef]
28. Nama, S.; Saha, A.K.; Sharma, S. A novel improved symbiotic organisms search algorithm. *Comput. Intell.* **2020**, *38*, 947–977. [CrossRef]
29. Secui, D.C. A modified Symbiotic Organisms Search algorithm for large scale economic dispatch problem with valve-point effects. *Energy* **2016**, *113*, 366–384. [CrossRef]
30. Nama, S.; Saha, A.K.; Ghosh, S. A hybrid symbiosis organisms search algorithm and its application to real world problems. *Memetic Comput.* **2016**, *9*, 261–280. [CrossRef]
31. Ezugwu, A.E.-S.; Adewumi, A.O. Discrete symbiotic organisms search algorithm for travelling salesman problem. *Expert Syst. Appl.* **2017**, *87*, 70–78. [CrossRef]
32. Ezugwu, A.E.; Adeleke, O.J.; Viriri, S. Symbiotic organisms search algorithm for the unrelated parallel machines scheduling with sequence-dependent setup times. *PLoS ONE* **2018**, *13*, e0200030. [CrossRef]
33. Tsai, H.-C. A corrected and improved symbiotic organisms search algorithm for continuous optimization. *Expert Syst. Appl.* **2021**, *177*, 114981. [CrossRef]
34. Kumar, S.; Tejani, G.G.; Mirjalili, S. Modified symbiotic organisms search for structural optimization. *Eng. Comput.* **2018**, *35*, 1269–1296. [CrossRef]
35. Miao, F.; Zhou, Y.; Luo, Q. A modified symbiotic organisms search algorithm for unmanned combat aerial vehicle route planning problem. *J. Oper. Res. Soc.* **2018**, *70*, 21–52. [CrossRef]
36. Çelik, E. A powerful variant of symbiotic organisms search algorithm for global optimization. *Eng. Appl. Artif. Intell.* **2019**, *87*, 103294. [CrossRef]
37. Do, D.T.; Lee, J. A modified symbiotic organisms search (mSOS) algorithm for optimization of pin-jointed structures. *Appl. Soft Comput.* **2017**, *61*, 683–699. [CrossRef]
38. Nama, S.; Saha, A.K.; Sharma, S. Performance up-gradation of Symbiotic Organisms Search by Backtracking Search Algorithm. *J. Ambient Intell. Humaniz. Comput.* **2021**, *13*, 5505–5546. [CrossRef] [PubMed]

39. Olukanmi, P.O.; Nelwamondo, F.; Marwala, T. k-Means-Lite: Real time clustering for large datasets. In Proceedings of the 2018 5th International Conference on Soft Computing & Machine Intelligence (ISCMI), Nairobi, Kenya, 21–22 November 2018; pp. 54–59. [CrossRef]
40. Davies, D.L.; Bouldin, D.W. A cluster separation measure. *IEEE Trans. Pattern Anal. Mach. Intell.* **1979**, *2*, 224–227. [CrossRef]
41. Chou, C.-H.; Su, M.-C.; Lai, E. A new cluster validity measure and its application to image compression. *Pattern Anal. Appl.* **2004**, *7*, 205–220. [CrossRef]
42. Arbelaitz, O.; Gurrutxaga, I.; Muguerza, J.; Pérez, J.M.; Perona, I. An extensive comparative study of cluster validity indices. *Pattern Recognit.* **2013**, *46*, 243–256. [CrossRef]
43. Chouikhi, H.; Charrad, M.; Ghazzali, N. A comparison study of clustering validity indices; A comparison study of clustering validity indices. In Proceedings of the 2015 Global Summit on Computer & Information Technology (GSCIT), Sousse, Tunisia, 11–13 June 2015. [CrossRef]
44. Xia, S.; Peng, D.; Meng, D.; Zhang, C.; Wang, G.; Giem, E.; Wei, W.; Chen, Z. Fast adaptive clustering with no bounds. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *44*, 87–99. [CrossRef]
45. Goldanloo, M.J.; Gharehchopogh, F.S. A hybrid OBL-based firefly algorithm with symbiotic organisms search algorithm for solving continuous optimization problems. *J. Supercomput.* **2021**, *78*, 3998–4031. [CrossRef]
46. Nguyen-Van, S.; Nguyen, K.T.; Luong, V.H.; Lee, S.; Lieu, Q.X. A novel hybrid differential evolution and symbiotic organisms search algorithm for size and shape optimization of truss structures under multiple frequency constraints. *Expert Syst. Appl.* **2021**, *184*, 115534. [CrossRef]
47. Huo, L.; Zhu, J.; Li, Z.; Ma, M. A Hybrid Differential Symbiotic Organisms Search Algorithm for UAV Path Planning. *Sensors* **2021**, *21*, 3037. [CrossRef] [PubMed]
48. Farnad, B.; Jafarian, A.; Baleanu, D. A new hybrid algorithm for continuous optimization problem. *Appl. Math. Model.* **2018**, *55*, 652–673. [CrossRef]
49. Gharehchopogh, F.S.; Shayanfar, H.; Gholizadeh, H. A comprehensive survey on symbiotic organisms search algorithms. *Artif. Intell. Rev.* **2019**, *53*, 2265–2312. [CrossRef]
50. Ghezelbash, R.; Maghsoudi, A.; Shamekhi, M.; Pradhan, B.; Daviran, M. Genetic algorithm to optimize the SVM and K-means algorithms for mapping of mineral prospectivity. *Neural Comput. Appl.* **2022**, 1–15. [CrossRef]
51. Yastrebov, A.; Kubuś, Ł.; Poczeta, K. Multiobjective evolutionary algorithm IDEA and k-means clustering for modeling multidimenional medical data based on fuzzy cognitive maps. *Nat. Comput.* **2022**, 1–11. [CrossRef]
52. Zhang, H.; Peng, Q. PSO and K-means-based semantic segmentation toward agricultural products. *Futur. Gener. Comput. Syst.* **2021**, *126*, 82–87. [CrossRef]
53. Li, Y.; Chu, X.; Tian, D.; Feng, J.; Mu, W. Customer segmentation using K-means clustering and the adaptive particle swarm optimization algorithm. *Appl. Soft Comput.* **2021**, *113*, 107924. [CrossRef]
54. Olukanmi, P.O.; Nelwamondo, F.; Marwala, T. k-Means-MIND: An Efficient Alternative to Repetitive k-Means Runs. In Proceedings of the 2020 7th International Conference on Soft Computing & Machine Intelligence (ISCMI), Stockholm, Sweden, 14–15 November 2020; pp. 172–176. [CrossRef]
55. Jain, A.K. Data clustering: 50 years beyond K-means. *Pattern Recognit. Lett.* **2009**, *31*, 651–666. [CrossRef]
56. Das, S.; Suganthan, P.N. Differential Evolution: A Survey of the State-of-the-Art. *IEEE Trans. Evol. Comput.* **2011**, *15*, 4–31. [CrossRef]
57. Kwak, S.G.; Kim, J.H. Central limit theorem: The cornerstone of modern statistics. *Korean J. Anesthesiol.* **2017**, *70*, 2, 144. [CrossRef]
58. Murugavel, P.; Punithavalli, M. Performance Evaluation of Density-Based Outlier Detection on High Dimensional Data. *Int. J. Comput. Sci. Eng.* **2013**, *5*, 62–67.
59. Rousseeuw, P.J.; Croux, C. Alternatives to the median absolute deviation. *J. Am. Stat. Assoc.* **1993**, *88*, 1273–1283. [CrossRef]
60. Leys, C.; Ley, C.; Klein, O.; Bernard, P.; Licata, L. Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *J. Exp. Soc. Psychol.* **2013**, *49*, 764–766. [CrossRef]
61. Ezugwu, A.E. Nature-inspired metaheuristic techniques for automatic clustering: A survey and performance study. *SN Appl. Sci.* **2020**, *2*, 273. [CrossRef]
62. Das, S.; Konar, A. Automatic image pixel clustering with an improved differential evolution. *Appl. Soft Comput.* **2009**, *9*, 226–236. [CrossRef]
63. Bandyopadhyay, S.; Maulik, U. Nonparametric genetic clustering: Comparison of validity indices. *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.* **2001**, *31*, 120–125. [CrossRef]
64. Zhou, X.; Gu, J.; Shen, S.; Ma, H.; Miao, F.; Zhang, H.; Gong, H. An automatic K-Means clustering algorithm of GPS data combining a novel niche genetic algorithm with noise and density. *ISPRS Int. J. Geo-Inf.* **2017**, *6*, 392. [CrossRef]
65. Bandyopadhyay, S.; Maulik, U. Genetic clustering for automatic evolution of clusters and application to image classification. *Pattern Recognit.* **2002**, *35*, 1197–1208. [CrossRef]
66. Lai, C.-C. A novel clustering approach using hierarchical genetic algorithms. *Intell. Autom. Soft Comput.* **2005**, *11*, 143–153. [CrossRef]
67. Lin, H.-J.; Yang, F.-W.; Kao, Y.-T. An Efficient GA-based Clustering Technique. *J. Appl. Sci. Eng.* **2005**, *8*, 113–122.
68. Liu, R.; Zhu, B.; Bian, R.; Ma, Y.; Jiao, L. Dynamic local search based immune automatic clustering algorithm and its applications. *Appl. Soft Comput.* **2014**, *27*, 250–268. [CrossRef]

69. Kundu, D.; Suresh, K.; Ghosh, S.; Das, S.; Abraham, A.; Badr, Y. Automatic Clustering Using a Synergy of Genetic Algorithm and Multi-objective Differential Evolution. In *International Conference on Hybrid Artificial Intelligence Systems*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 177–186. [CrossRef]
70. Kumar, V.; Chhabra, J.K.; Kumar, D. Automatic Data Clustering Using Parameter Adaptive Harmony Search Algorithm and Its Application to Image Segmentation. *J. Intell. Syst.* **2016**, *25*, 595–610. [CrossRef]
71. Anari, B.; Torkestani, J.A.; Rahmani, A. Automatic data clustering using continuous action-set learning automata and its application in segmentation of images. *Appl. Soft Comput.* **2017**, *51*, 253–265. [CrossRef]
72. Kuo, R.; Huang, Y.; Lin, C.-C.; Wu, Y.-H.; Zulvia, F.E. Automatic kernel clustering with bee colony optimization algorithm. *Inf. Sci.* **2014**, *283*, 107–122. [CrossRef]
73. Liu, Y.; Wu, X.; Shen, Y. Automatic clustering using genetic algorithms. *Appl. Math. Comput.* **2011**, *218*, 1267–1279. [CrossRef]
74. Chowdhury, A.; Das, S. Automatic shape independent clustering inspired by ant dynamics. *Swarm Evol. Comput.* **2011**, *3*, 33–45. [CrossRef]
75. Kumar, V.; Chhabra, J.K.; Kumar, D. Automatic cluster evolution using gravitational search algorithm and its application on image segmentation. *Eng. Appl. Artif. Intell.* **2014**, *29*, 93–103. [CrossRef]
76. Sheng, W.; Chen, S.; Sheng, M.; Xiao, G.; Mao, J.; Zheng, Y. Adaptive multisubpopulation competition and multiniche crowding-based memetic algorithm for automatic data clustering. *IEEE Trans. Evol. Comput.* **2016**, *20*, 838–858. [CrossRef]
77. Das, S.; Chowdhury, A.; Abraham, A. A Bacterial Evolutionary Algorithm for Automatic Data Clustering. In Proceedings of the 2009 IEEE Congress on Evolutionary Computation, Trondheim, Norway, 8–21 May 2009.
78. Talbi, E.-G. *Metaheuristics: From Design to Implementation*; John Wiley & Sons: Hoboken, NJ, USA, 2009.
79. Chowdhury, A.; Bose, S.; Das, S. Automatic Clustering Based on Invasive Weed Optimization Algorithm. In Proceedings of the International Conference on Swarm, Evolutionary, and Memetic Computing 2011, Visakhapatnam, India, 19–21 December 2011; pp. 105–112. [CrossRef]
80. Zhang, X.; Li, J.; Yu, H. Local density adaptive similarity measurement for spectral clustering. *Pattern Recognit. Lett.* **2011**, *32*, 352–358. [CrossRef]
81. Agbaje, M.B.; Ezugwu, A.E.; Els, R. Automatic data clustering using hybrid firefly particle swarm optimization algorithm. *IEEE Access* **2019**, *7*, 184963–184984. [CrossRef]