electronics

MDPI

Article

Phase-Angle-Encoded Snake Optimization Algorithm for K-Means Clustering

Dan Xue [1], Sen-Yuan Pang [1], Ning Liu [1], Shang-Kun Liu [1] and Wei-Min Zheng [2,*]

[1] College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China; xuedan@sdust.edu.cn (D.X.); pangsenyuan@sdust.edu.cn (S.-Y.P.); liuning@sdust.edu.cn (N.L.); liushangkun@sdust.edu.cn (S.-K.L.)

[2] College of Artificial Intelligence, Nanjing University of Information Science and Technology, Nanjing 210044, China

* Correspondence: 003872@nuist.edu.cn

Abstract: The rapid development of metaheuristic algorithms proves their advantages in optimization. Data clustering, as an optimization problem, faces challenges for high accuracy. The K-means algorithm is traditaaional but has low clustering accuracy. In this paper, the phase-angle-encoded snake optimization algorithm (θ-SO), based on mapping strategy, is proposed for data clustering. The disadvantages of traditional snake optimization include slow convergence speed and poor optimization accuracy. The improved θ-SO uses phase angles for boundary setting and enables efficient adjustments in the phase angle vector to accelerate convergence, while employing a Gaussian distribution strategy to enhance optimization accuracy. The optimization performance of θ-SO is evaluated by CEC2013 datasets and compared with other metaheuristic algorithms. Additionally, its clustering optimization capabilities are tested on Iris, Wine, Seeds, and CMC datasets, using the classification error rate and sum of intra-cluster distances. Experimental results show θ-SO surpasses other algorithms on over 2/3 of CEC2013 test functions, hitting a 90% high-performance mark across all clustering optimization tasks. The method proposed in this paper effectively addresses the issues of data clustering difficulty and low clustering accuracy.

Keywords: phase-angle-encoded snake optimization; snake optimization; metaheuristic algorithms; K-means clustering

Citation: Xue, D.; Pang, S.-Y.; Liu, N.; Liu, S.-K.; Zheng, W.-M. Phase-Angle-Encoded Snake Optimization Algorithm for K-Means Clustering. *Electronics* **2024**, *13*, 4215. https://doi.org/10.3390/electronics13214215

Academic Editors: Chih-Lung Lin, Bacha Rehman and Amin Amini

Received: 13 September 2024
Revised: 23 October 2024
Accepted: 25 October 2024
Published: 27 October 2024

1. Introduction

Metaheuristics are algorithmic frameworks inspired by natural phenomena, designed to address complex optimization problems for which traditional algorithms are often inefficient [1]. Metaheuristic algorithms (MAs) have proven effective in addressing various optimization challenges [2], including vehicle routing problems [3], electric systems [4], gene selection [5], and industry applications [6]. Broadly speaking, MAs lack a universally accepted classification scheme [7]. Hashim et al. classify MA into four distinct categories: EAs, SI, physical and chemical algorithms, as well as human-based algorithms [8]. Zhao et al. categorize MA into several types: EAs, SI, physics or mathematics, and others [9]. In this paper, MAs are classified into three main categories based on their underlying heuristic principles: evolutionary algorithms, swarm intelligence, and others. Figure 1 shows the classification of the MAs.

(1) Evolutionary Algorithms (EAs). Evolutionary algorithms simulate natural selection and genetic evolution processes to address optimization problems. Guided by evolutionary principles such as inheritance, mutation, selection, and crossover, these algorithms aim to iteratively enhance a population of potential solutions. With each iteration, the overall fitness of the solutions is improved, serving as a metric to quantify their effectiveness in addressing the problem. By applying evolutionary operators, new individuals are generated, gradually evolving into superior versions compared to the previous generation. Ultimately, the goal

is to converge on the optimal solution. As a result, evolutionary algorithms demonstrate a profound capability for global optimization while effectively avoiding local optima [10]. Within the domain of evolutionary computation, the four principal paradigms are comprised of genetic algorithms (GA), evolutionary programming (EP), evolutionary strategies (ES), and genetic programming (GP) [11]. The most emblematic example is the genetic algorithm (GA), originally proposed by John H. Holland [12].

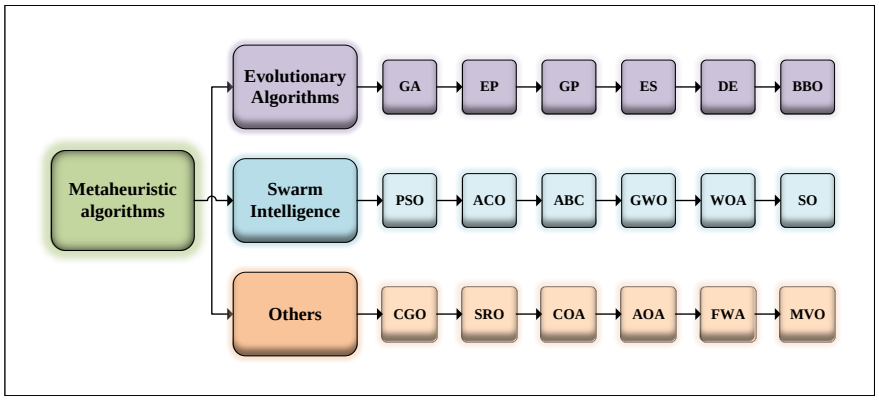

Figure 1. Classification of Metaheuristic Algorithms.

Among the four, GA simulates the principles of natural selection and genetics, as observed in Darwinian evolution. The fundamental procedures within the GA revolve around the concepts of inheritance, where traits are passed down from one generation to the next, and variation, which introduces novelty through mutation and crossover. These mechanisms work in concert to evolve a population of candidate solutions, driving both exploration and exploitation in the search for optimal or near-optimal outcomes [13]. Some other popular evolutionary algorithms are differential evolution (DE) [14] and biogeography-based optimization (BBO) [15]. DE was introduced by Rainer Storn and Kenneth Price in 1995 for the purpose of solving real-parameter optimization problems over continuous spaces [16]. It iteratively improves a population of candidate solutions through three key processes: mutation operation, crossover operation, and selection operation. During the mutation phase, new solution candidates are generated by perturbing the solution space based on differences between existing solutions, thereby fostering diversity. The crossover phase involves recombining these mutated solutions with other solutions to create offspring, which are subsequently evaluated. Finally, the selection mechanism ensures that only the most fit solutions are retained for the next generation [17].

(2) Swarm Intelligence (SI). SI algorithms draw inspiration from the collective behavior and interaction patterns observed in natural swarms, aiming to mimic these natural phenomena to solve complex optimization problems [18]. Inside a swarm, although individuals within the system lack intelligence, the system as a whole can exhibit intelligent behavior, akin to a form of collective intelligence. SI algorithms employ a population of agents, each represented by a solution vector. The system is initially populated with random samples from the search space. These agents are updated in a quasi-deterministic manner, known as algorithmic dynamics, which governs the evolution of the system based on a set of equations. The introduction of randomness serves to disrupt the equilibrium of the system and potentially enables it to escape from local minima. A selection mechanism identifies the best solutions, allowing them to propagate to the next generation. This process often leads to convergence towards a set of optimal solutions as iterations progress [19]. Traditional swarm intelligence algorithms sometimes fall short when tackling complex, real-world multi-objective optimization challenges. However, in recent years, scholars have developed numerous innovative swarm intelligence optimization algorithms that exhibit strong adaptability and have demonstrated impressive experimental results in solving intricate practical problems [20].

The most widely recognized examples of SI algorithms include the particle swarm optimization (PSO) [21], ant colony optimization (ACO) [22], artificial bee colony algorithm

(ABC) [23], grey wolf optimizer (GWO) [24], whale optimization algorithm (WOA) [25], firefly algorithm (FA) [26], seagull optimization algorithm (SOA) [27], ant lion optimizer (ALO) [28], squirrel search algorithm (SSA) [29], cuckoo search (CS) [30], crow search algorithm (CSA) [31], bat algorithm (BA) [32], snake optimization (SO) [8], and fish migration optimization algorithm (FMO) [33]. PSO, initially introduced in 1995, is a stochastic optimization technique that is inspired by the social behaviors of animals, employing population-based principles [34,35].

(3) Others. In addition to the previously mentioned categories, evolutionary algorithms (EAs) and swarm intelligence (SI), the final category includes a diverse range of other metaheuristic methods. These are based on either physics, mathematics, or human-based algorithms, which encompass algorithms inspired by various aspects of human behavior, including both physical and non-physical activities like thinking and social interaction. Such algorithms include chaos game optimization (CGO) [36], search and rescue optimization (SRO) [37], cognitive behavior optimization algorithm (COA) [38], Archimedes optimization algorithm (AOA) [39], fireworks algorithm (FWA) [40], multiverse optimizer (MVO) [41], Lichtenberg algorithm (LA) [42], poor and rich optimization (PRO) [43], atomic orbital search (AOS) [44], sine cosine algorithm (SCA) [45], and human mental search (HMS) [46].

Meanwhile, grouping data into clusters based on common attributes or features is the essence of clustering, a process that organizes information into distinct segments for analysis [47,48]. Benabdella et al. categorized clustering into partitioning-based, hierarchical-based, density-based, grid-based, and model-based methods by matching the considered factors to the 4Vs of big data: volume, variety, velocity, and value [49]. It is generally agreed that clustering algorithms can be broadly categorized into two primary classifications: hierarchical clustering and partitional clustering [50–52]. Due to the varying requirements of different application scenarios for clustering performance, as well as the distinct characteristics of datasets, no single clustering algorithm can excel across all evaluation criteria [53]. Among various clustering methods, one of the most popular used clustering algorithms is the K-means clustering algorithm because of its easiness and effectiveness [54].

The K-means clustering algorithm is an iterative refinement process that comprises two main steps: initialization, where the initial set of K centroids is determined, and an iterative procedure, which is originally known as the Lloyd algorithm [55]. The K-means algorithm is a typical prototype-based clustering method, which usually assumes that the cluster structure can be depicted through a set of prototypes. It is one of the most commonly used methods in clustering. The K-means algorithm takes the mean vector of samples within a cluster as the prototype to depict the cluster structure. It initializes cluster centers with K random samples and adopts a greedy strategy to perform clustering through iterative optimization. This clustering process is simple, efficient, easy to understand and implement, and has good scalability; especially when the data are close to normal distribution, the effect is better.

However, within the framework of the K-means, the initial placement of centroids holds a pivotal role in shaping the ultimate clustering outcomes. Different initial cluster centroids often result in varied ultimate clustering configurations, thus resulting in a relatively poor stability of the K-means. Furthermore, the choice of cluster centroids can cause the clustering results to fall into a local optimum. It is one of the key challenges in the K-means that lies in the assignment of data points to centroids [56]. Therefore, many studies have improved the classic K-means algorithm.

The K-medoids algorithm is specifically designed to identify medoids, which are central representative points within a cluster. Compared to the K-means algorithm, K-medoids exhibits greater robustness due to its approach of selecting k representative objects that minimize the total dissimilarity among data points within the clusters. In contrast, K-means utilizes the sum of squared Euclidean distances between data points as the distance metric, which can be influenced by outliers and noise. By minimizing the sum of dissimilarities rather than squared distances, K-medoids is able to mitigate the effects of outliers and noise in the datasets, resulting in potentially more accurate and robust clustering outcomes [57].

K-means++ is a clustering algorithm specifically designed to enhance the process of selecting initial cluster centers in the K-means algorithm. Unlike K-means, which randomly

selects the initial cluster centers, K-means++ employs a more strategic approach by selecting new cluster centers based on a probability proportional to the squared distance of each data point to the closest existing cluster center. This strategy aims to reduce the similarity among the initial cluster centers, potentially accelerating the convergence of the algorithm and improving the quality of the clustering results. However, despite its improvements in clustering effectiveness, when K-means++ runs in sequential mode, the speed of the clustering process can still be limited due to the sequential processing of all data points in each iteration, especially when dealing with large-scale datasets [58].

Generally speaking, MAs are a powerful tool in optimization due to their global exploration capabilities, which involve the use of randomization to explore multiple regions within the solution space and avoid becoming trapped in local optima. Additionally, these algorithms possess local exploitation abilities, allowing them to conduct thorough searches within specific areas to fine-tune and find better or optimal solutions. From these empirical studies, many studies have focused on the parameters of algorithms, while few have delved into the basic mechanisms. In this paper, a new variant of SO, named phase-angle-encoded snake optimization, is proposed that aims to enhance the performance of heuristic algorithms and overcome the challenges faced by the K-means algorithm in clustering. Compared to the traditional SO algorithm, the improved θ-SO utilizes phase angles for boundary setting, allowing for efficient adjustments in the phase angle vector to expedite convergence. Additionally, it adopts a Gaussian distribution strategy to further enhance optimization accuracy. The challenge that the θ-SO aims to address pertains to the difficulties associated with data clustering and the resultant low clustering accuracy.

The rest of the manuscript is organized as follows: Section 2 introduces the related work on K-means and SO, while Section 3 focuses on the θ-SO algorithm we have proposed. Section 4 details the related experiments and their results. Finally, Section 5 concludes the work with a summary and final remarks.

2. Related Work

In this section, we present a brief overview of some related works that primarily focus on recognizing and understanding the K-means clustering algorithm. Meanwhile, we offer a concise review of the pertinent research endeavors that are centered primarily on the recognition and comprehension of the snake optimization algorithm.

2.1. K-Means Clustering Algorithm

The fundamental principle of the K-means clustering algorithm involves randomly selecting K points from the dataset as the initial centroids. Subsequently, it calculates the distance between each remaining sample and these cluster centers, assigning each sample to the cluster with the closest centroid. Each centroid and its assigned data points constitute a cluster. Once all samples in the dataset have been allocated to their respective clusters, the new centroid of each cluster is computed by taking the mean of the data points within that cluster. The algorithm then compares the centroids from the current iteration with those from the previous iteration. If there is no significant change in the centroids, indicating convergence of the sum of squared errors criterion, the algorithm terminates. Otherwise, it proceeds with another iteration, recalculating the centroids based on the current assignment of data points, without randomly selecting new initial centroids.

During the clustering process, it is paramount to maintain an optimal balance, ensuring that the distances between sample points within the same cluster are minimized, while simultaneously maximizing the distances between sample points belonging to distinct clusters. This is carried out to minimize the sum of squared errors (SSE), which is used as an objective function to measure clustering quality and is defined in Equation (1). Three commonly used distance metrics that are pivotal in the K-means clustering algorithm are Euclidean distance, Manhattan distance, and Minkowski distance, defined in Equations (2), (3), and (4), respectively. Typically, the Euclidean distance is used as a similarity measure.

Sum of squared error (SSE):

$$E = \sum_{i=1}^{k} \sum_{x \in C_i} d(x, c_i)^2, \tag{1}$$

where the dataset D = { $x_i \mid i = 1, \ldots, N$ }, consisting of n-dimensional vector data points, is to be partitioned into k clusters, and N is the total number of the data points. $C_1, \ldots, C_k$, where x_i represents the ith data point. Ultimately, the set D is divided into k clusters. E represents the sum of squared errors of all objects in the dataset, and x is a point in space representing the data objects belonging to a given cluster. c_i is the centroid of cluster C_i, while d represents their Euclidean distance.

In K-means clustering, various distance metrics are crucial for calculating the distances between individual data points and the centroids. These measurements are used to assign data points to their respective clusters, that is, the closest centroid based on the measured distance. Three distinct distance metric methods are encompassed and detailed as follows.

(1) Euclidean distance:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}, \tag{2}$$

(2) Manhattan distance:

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|, \tag{3}$$

(3) Minkowski distance:

$$d(x, y) = \left(\sum_{i=1}^{n} (|x_i - y_i|)^p\right)^{\frac{1}{p}}, \tag{4}$$

where x and y are the coordinates of two points, n represents the dimensionality, and p is a parameter. When $p = 2$, it corresponds to the Euclidean distance; meanwhile, when $p = 1$, it refers to the Manhattan distance.

In the K-means algorithm, the selection of initial cluster centers has a significant impact on the clustering results. For different initial cluster centers, the final clustering outcomes often differ, leading to poor stability of the K-means clustering algorithm [59]. Additionally, the choice of cluster centers can cause the clustering results to fall into local optima [60]. Nie et al. have reformulated the classical K-means objective function as a trace maximization problem, introducing a new formulation that eliminates the need to calculate cluster centers in each iteration and requires fewer additional intermediate variables during the optimization process [61].

Based on partial-order relations, Wan et al. have developed an enhanced K-means algorithm known as the multi-feature induced order K-means algorithm, which incorporates a novel centroid initialization method leveraging partial-order relations and utilizing a multi-feature induced ordered weighted average operator [62]. Yang et al. propose a sparse K-means clustering algorithm that incorporates anchor graph regularization to enhance the robustness of initial cluster center selection and enhance the sparsity of the membership matrix [63].

Additionally, automatic K-means clustering algorithms, incorporating nature-inspired metaheuristic strategies, outperform traditional clustering methods, especially in terms of their rapid convergence and capacity to generate high-quality solutions [64].

2.2. Snake Optimization Algorithm

Introduced by Hashim, the SO represents a pioneering approach that abstracts the intricate behaviors of snakes, including mating, feeding, and fighting, into a rigorous mathematical framework. The process of feeding within the algorithm is split into two distinct stages: exploration and exploitation. The core idea of SO lies in simulating the combat and mating behaviors of snakes to achieve global optimization. In nature, the behaviors of snakes are influenced by a variety of factors, including the availability of food, temperature, and so on. The SO algorithm incorporates these factors into its algorithm

design. By simulating the different behavioral patterns of snake swarms in scenarios of food scarcity and abundance, it aims to solve optimization problems. When the food threshold is not met, the SO algorithm enters the exploration phase. During this phase, snakes, represented as individuals in the population, roam randomly in the search space, simulating their natural behavior of searching for food. This random movement allows the algorithm to explore the search space thoroughly and discover new potential solutions. When food becomes available, the algorithm transitions to the exploitation phase. Here, snakes begin to converge towards areas with higher concentrations of food, analogous to moving towards better solutions in the optimization problem. Additionally, temperature, which affects snake behavior in the wild, is incorporated into the algorithm to influence the movement and search behavior of the individuals.

The SO approach differentiates the population by assigning each individual as either male or female. The initialization of SO involves creating a population through random generation. Moreover, considering snakes are ectothermic, temperature significantly affects their fighting and mating behaviors [8]. Each snake is assigned a random position in the search space and evaluated based on a fitness function specific to the optimization problem. The fitness of each snake is evaluated using a predefined objective function. This function measures the quality of the position of the snake in the search space and assigns a fitness score accordingly. The fitness score determines the ability of the snake to compete for food and mates in the simulation. The SO algorithm's combination of exploration and exploitation mechanisms ensures that it can effectively balance the need for global search and local refinement. This balance leads to faster convergence towards optimal or near-optimal solutions, even for complex and non-linear optimization problems. While innovative, it has limitations. Its performance is sensitive to parameter tuning, which can be challenging. Additionally, its exploration–exploitation balance may not be optimal for highly multi-modal problems, risking premature convergence. Moreover, SO lacks theoretical convergence guarantees, and its computational complexity scales poorly with problem size, limiting its applicability for large-scale or real-time optimization tasks.

Since the proposal of the SO algorithm, there have been several papers that have improved upon the algorithm. Li et al. use SO to address the threshold selection issue in the variable-step multiscale single threshold slope entropy [65]. Zheng et al. propose the compact SO to enhance indoor positioning accuracy [66]. Yao et al. introduce an advanced version of the SO, featuring novel dynamic update mechanisms and incorporating a mirror imaging approach inspired by convex lens imaging principles, aimed at addressing real-world engineering challenges [67]. Yan et al. incorporate a chaos mechanism during the initialization phase and employ a sigmoid-based acceleration coefficient to enhance SO, thereby delivering an optimal convolutional neural network structure [68].

2.3. Heuristics Algorithms and K-Means

Given the classic nature of the K-means algorithm, numerous heuristic algorithms have been combined with it for optimization purposes. In PSO, each candidate solution for a particular problem is envisioned as a particle traversing the vast landscape of the problem space, equipped with a unique velocity. Each particle, influenced by a blend of deterministic and stochastic factors, integrates information from its own historical best positions, the global best position, and current location, as well as from one or more neighboring particles within the swarm. After each iteration, during which all particles update their positions, the swarm collectively inches closer to the optimal solution of the objective function, gradually converging towards the desired outcome [69].

During the initial stages of the global search process, PSO experiences successful convergence. However, as it approaches the global optimum, a notable issue emerges: the search process slows down significantly. Alireza Ahmadyfard et al. proposed the PSO-KM algorithm by combining the PSO with the K-means, leveraging the faster convergence to the optimal solution offered by K-means to address this problem [70]. To better demonstrate the optimization capabilities of the proposed compact particle swarm optimization based on t-location–scale distribution, Ning Liu et al. conducted clustering experiments utilizing K-

means as a benchmark, allowing for a further comparison and evaluation of the optimization effects achieved by tCPSO [71].

Shruti Kapil et al. proposed the genetic K-means algorithm, leveraging an analogy between dataset scope and chromosome length. This innovative approach aims to address the challenges of high computational cost and time consumption in traditional k-means clustering, which are exacerbated by the number of data items, clusters, and iterations [72]. T. Namratha Reddy et al. applied ant colony optimization to the K-means clustering algorithm by incorporating two strategies: allowing ants to undertake a random walk and subsequently calculating the pick and drop probabilities of specific data items, and the alternative approach of calculating these probabilities first and directing ants towards the data items with the highest likelihood of being reassigned to a different cluster. These strategies aim to enhance the accuracy of the resulting clusters [73].

3. Phase-Angle-Encoded Snake Optimization Algorithm (θ-SO) for Clustering

This section delves into the theoretical underpinnings of the θ-SO algorithm, elucidating its fundamental principles and core mechanisms. While using the improved algorithm for clustering, the correspondence between individual dimensions and the solution to the clustering problem is elaborated.

3.1. Phase-Angle-Encoded Snake Optimization Algorithm

The key idea of the θ-SO is to map the search space of the algorithm to the polar search space. This process enables the algorithm to delve into the search space. The proposed θ-SO denotes the location of individuals as a phase angle vector $\theta = [\theta_1, \theta_2, \ldots, \theta_D]$, where each phase angle $\theta_i \in [-\pi/2, \pi/2]$ [74,75]. At the onset of the θ-SO procedure, akin to numerous metaheuristic methodologies, the first phase involves creating a randomized population with a uniform distribution to commence the optimization procedure. The initial population can be obtained by the subsequent Equation (5).

$$\theta_i = \theta_{min} + r \times (\theta_{max} - \theta_{min}), \tag{5}$$

where θ_i is the position of the ith individual, and r is the random number between 0 and 1. θ_{max} represents the upper bound, while θ_{min} represents the lower bound of the problem.

We divided the swarm into two equal groups: males and females.

$$\theta_m \approx N/2, \tag{6}$$

$$\theta_f = N - \theta_m, \tag{7}$$

where N represents the total population, θ_m signifies the count of male individuals, and θ_f denotes the number of female individuals.

We identified and registered the best individual within each category—the best male ($f_{best,m}$), the best female ($f_{best,f}$), and the best food location (f_{food}).

The temperature ($Temp$) can be obtained using Equation (8). The parameter settings, including temperature, c_1, threshold, c_2, and rand, are all consistent with those of the SO algorithm. This ensures that the simulated snakes exhibit similar behaviors in scenarios of food scarcity and abundance, as well as under conditions of high or low temperatures, mimicking their natural behavior of searching for food.

$$Temp = exp(\frac{-t}{T}), \tag{8}$$

where t denotes the current iteration number, and T represents the maximum allowable number of iterations.

Equation (9) provides a definition for the quantity of food(Q).

$$Q = c_1 \times exp(\frac{t-T}{T}). \tag{9}$$

where c_1 is constant and equals 0.5.

When $Q < Threshold$ ($Threshold$ = 0.25) is indicated, it signifies that the environment has insufficient food, prompting the snakes to enter the exploration phase for food in the θ-SO, the update causing individuals to conduct a random search according to Equations (10) and (11).

$$\theta_{i,m}^{t+1} = \theta_{random,m}^{t} \pm C_2 \times A_m \times ((\theta_{max} - \theta_{min}) \times rand + \theta_{min}), \tag{10}$$

$$\theta_{i,f}^{t+1} = \theta_{random,f}^{t} \pm C_2 \times A_f \times ((\theta_{max} - \theta_{min}) \times rand + \theta_{min}), \tag{11}$$

where $\theta_{i,m}^{t+1}$ and $\theta_{i,f}^{t+1}$, respectively, denote the locations of the ith male and female; meanwhile, $\theta_{random,m}^{t}$ and $\theta_{random,f}^{t}$ indicate the positions of snakes randomly selected from the male and female groups, respectively, at the tth iteration. The value of C_2 is configured to 0.05, while $rand$ denotes a random number between 0 to 1, and t represents the ongoing iteration of the algorithm. A_m and A_f symbolize the ability of male and female snakes to find the food, respectively. This ability is mathematically described by Equations (12) for males and (13) for females.

$$A_m = exp(\frac{-f_{rand,m}}{f_{i,m}}), \tag{12}$$

$$A_f = exp(\frac{-f_{rand,f}}{f_{i,f}}), \tag{13}$$

where $f_{rand,m}$ and $f_{rand,f}$, respectively, denote the fitness of $\theta_{rand,m}$ and $\theta_{rand,f}$, while $f_{i,m}$ and $f_{i,f}$ represent the fitness values corresponding to the i-th individual in both the male and female cohorts.

When $Q > Threshold$, it means that food exists, and the system enters the exploitation phase. During the exploitation phase, an abundance of food is accessible within the environment. If the $temperature > 0.6$, snakes persist in their food search. Subsequently, the locations of both male and female snakes can be updated utilizing Equation (14).

$$\theta_{i,j}^{t+1} = X_{food}^{t} \pm C_3 \times Temp \times rand \times (X_{food} - \theta_{i,j}^{t}), \tag{14}$$

where $\theta_{i,j}$ denotes the new location of an individual from both the male and female populations.

If $temperature < 0.6$, the snake will assume either a fighting mode or a mating mode.

If $rand > 0.6$, fighting mode occurs, as determined by Equations (15) and (16), where FM and FF are defined in Equations (17) and (18),

$$\theta_{i,m}^{t+1} = \theta_{i,m}^{t} \pm C_3 \times FM \times rand \times (X_{best,f} - \theta_{i,m}^{t}), \tag{15}$$

$$\theta_{i,f}^{t+1} = \theta_{i,f}^{t+1} \pm C_3 \times FF \times rand \times (X_{best,m} - \theta_{i,f}^{t+1}), \tag{16}$$

$$FM = exp(\frac{-f_{best,f}}{f_i}), \tag{17}$$

$$FF = exp(\frac{-f_{best,m}}{f_i}), \tag{18}$$

where FM and FF, respectively, signify the combative prowess of males and females, and $X_{best,m}$ and $X_{best,f}$, respectively, denote the locations of the most capable male and female entities within the group. $f_{best,f}$ and $f_{best,m}$, respectively, represent the fitness levels associated with $X_{best,f}$ and $X_{best,m}$.

If $rand < 0.6$, mating mode occurs, as determined by Equations (19) and (20), where MM and MF are defined in Equations (21) and (22):

$$\theta_{i,m}^{t+1} = \theta_{i,m}^{t} \pm C_3 \times M_m \times rand \times (Q \times \theta_{i,f}^{t} - \theta_{i,m}^{t}), \tag{19}$$

$$\theta_{i,f}^{t+1} = \theta_{i,f}^{t} \pm C_3 \times M_f \times rand \times (Q \times \theta_{i,m}^{t} - \theta_{i,f}^{t}), \tag{20}$$

$$MM = exp(\frac{-f_{i,f}}{f_{i,m}}), \tag{21}$$

$$MF = exp(\frac{-f_{i,m}}{f_{i,f}}), \tag{22}$$

where MM and MF represent the respective mating abilities of males and females.

If an egg hatches, the worst male and female individuals are chosen for replacement by Equations (23) and (24).

$$\theta_{worst,m} = \theta_{min} + rand \times (\theta_{max} - \theta_{min}), \tag{23}$$

$$\theta_{worst,f} = \theta_{min} + rand \times (\theta_{max} - \theta_{min}), \tag{24}$$

where $\theta_{worst,m}$ denotes the worst individual in the male group; meanwhile, $\theta_{worst,f}$ denotes the worst individual in the male group.

In this paper, unlike the SO algorithm that directly calculates the fitness value based on the position, the θ-SO algorithm maps the position of the current solution to the actual solution space within the problem threshold before evaluating its fitness value, as the boundaries of the phase angle do not correspond to the upper and lower bounds of the problem being solved. The transition from the polar search space to the original problem search space is achieved through the sinusoidal function and utilization of the subsequent Equation (25) as follows:

$$X_i^t = [(X_{max} - X_{min}) \times \sin(\theta_i^t) + X_{max} + X_{min}]/2, \tag{25}$$

where X_i^t is the mapped position, X_{max} represents the upper bound, while X_{min} represents the lower bound of the problem, and $\sin(\theta_i^t)$ represents the transformation of the current position. Figure 2 shows the curve of θ function.

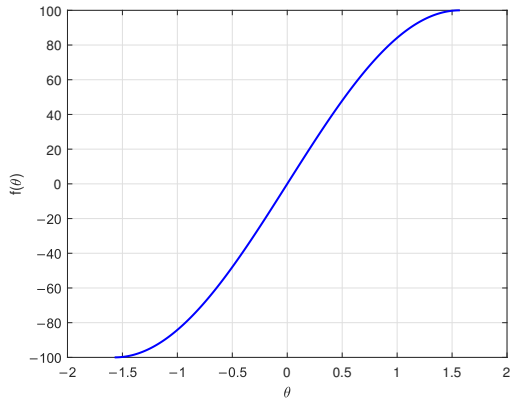

Figure 2. The Curve of θ Function.

The fitness calculation is obtained by the following Equation (26):

$$F_i^t = fitnessvalue(X_i^t) \tag{26}$$

where F_i^t represents the value of fitness, and $fitnessvalue$ is a general formula for fitness function used to evaluate the quality of a given solution, reflecting its performance in a specific optimization problem, to guide the search and selection process of the algorithm.

The determination of the best solution variable hinges on the fitness value, which functions as the standard in optimization algorithms for comparing the merit of diverse solutions. By evaluating the fitness value of each solution, the algorithm is capable of discerning which solutions lie closer to the optimal one. Within the scope of this paper, a lower fitness value indicates a solution that is more preferable. In the realm of clustering problems, this discernment is particularly made via a blend of accuracy and internal distance metrics. Accuracy

quantifies how well the clusters correspond to the genuine underlying structure of the data, whereas internal distance metrics assess the density and distinctiveness of the formed clusters.

By utilizing the aforementioned equation, the phase angle space is transformed into the suitable domain where the problem is being analyzed. This allows the algorithm to search the permissible area with greater accuracy. Algorithm 1 presents the illustrative pseudocode detailing the implementation of the θ-SO.

Algorithm 1 θ-SO pseudocode

Input: Dim(Dimensions), UB(Upper Bounds), LB(Lower Bounds), t(Current Iteration), T(Max Iteration)
Output: X_{best}(Best Solution)
Initialize the population randomly between $-\frac{\pi}{2}$ and $\frac{\pi}{2}$
while ($t \leq T$) **do**
    Map the phase angle to a position in the solution space using Equation (25)
    Evaluate all mapped positions in θ_m and θ_f
    Find best male $f_{best,m}$
    Find best female $f_{best,f}$
    Calculate $Temp$ using Equation (8)
    Calculate Q using Equation (9)
    if ($Q < 0.25$) **then**
    Perform Exploration Phase using Equations (10) and (11)
    else
        if ($temp > 0.6$) **then**
    Perform Exploitation Phase using Equation (14)
        else
            if ($rand > 0.6$) **then**
    Perform fight mode using Equations (15) and (16)
            else Perform mating mode using Equations (19) and (20)
    Change the worst male and female
            end if
        end if
    end if
    Sort fitness (N_m)
    if $Fitness$ (g^t_{bestm})>$Fitness$ (g^{t-1}_{bestm}) **then**
    $Count_m = Count_m + 1$
    end if
    if ($Count_m \geq 3$) **then**
    $Count_m = 0$
        for (i=1:5) **do**
    Generate $X_{newm(i)}$ according to Gaussian perturbation by Equation (27);
    Replace $X_m(N - i + 1) = X_{newm(i)}$
        end for
    end if
    Sort fitness (N_f)
    if $Fitness$ (g^t_{bestf})>$Fitness$ (g^{t-1}_{bestf}) **then**
    $Count_f = Count_f + 1$
    end if
    while ($Count_f \geq 3$) **do**
    $Count_f = 0$
        for i=1:5 **do**
    Generate $X_{newf(i)}$ according to Gaussian perturbation by Equation (27)
    Replace $X_f(N - i + 1) = X_{newf(i)}$
        end for
    end while
end while

The tendency of individuals within the population to converge towards the perceived best individual, if flawed, can misdirect the entire population, trapping it in a local optimum. Consequently, this study introduces an adaptation of the natural selection principle, aimed at empowering the population to escape such suboptimal traps. During the iteration, when the fitness value of the optimal individual in the population does not show significant improvement for consecutive generations, the algorithm will trigger the mechanism of survival of the fittest. The core of this mechanism is to select the top n individuals with the highest fitness after sorting, apply Gaussian disturbance to their chosen dimensions, and then replace the worst n individuals in the population with the disturbed individuals. The Gaussian perturbation strategy operates by randomly selecting d dimensions and sampling a new value for each dimension based on a Gaussian distribution. It is expressed in Equation (27), as follows:

$$\theta_{new}^{d} = Gaussian(d, \sigma^2) \tag{27}$$

where θ_{new}^{d} denotes the new position of the generated individual after Gaussian perturbation. d corresponds to a randomly selected dimension. σ^2 is the mean of the squared deviations between each individual and the average value. Through this operation, individuals can bypass local optima and prevent premature convergence.

3.2. K-Means Clustering Algorithm Based on θ-SO

Next, we integrate the proposed algorithm with K-means, exploring the application of an advanced optimization framework that transforms the classic K-means algorithm, enhancing its capabilities to tackle real-world challenges involving large and complex datasets. This integration provides a detailed examination of the proposed method, its theoretical underpinnings and empirical evaluations, demonstrating its advantages over traditional K-means.

Utilizing the devised algorithm for data clustering, the concept of candidate solutions assumes a pivotal role. Essentially, these solutions are embodied within one-dimensional arrays, where each array distinctly represents a potential strategy for grouping the data. In this framework, a candidate solution is construed as a set of K initial cluster centers, serving as the foundational points for the clustering process. The dimensionality of the cluster centers is dictated by the characteristics of the data objects being clustered. For example, in the scenario illustrated in Figure 3, a clustering problem arises involving data objects with seven distinct features. As a result, each element within the one-dimensional array representing a candidate solution aligns with one of these features, specifying the location of the corresponding cluster center within the multidimensional feature space.This methodology enables a thorough examination of diverse clustering configurations, as each candidate solution presents a unique arrangement of initial cluster center positions. By sequentially examining these candidate solutions and assessing their proficiency in grouping similar data objects, the optimal clustering strategy that most accurately mirrors the inherent structure of the data can be discerned. It should be noted that in Figure 3, the use of gray is solely intended to distinguish between various features and does not convey any sense of hierarchy or grading.

The K-means algorithm is the most well-known clustering algorithm that uses distance measurement [64,76]. Therefore, the sum of intra-cluster distances (SD) and the error rate (ER) [71,77] have been designed in the article to evaluate the performance of θ-SO with K-means. SD is calculated by aggregating the distances between each data point and its respective cluster center, as formulated in Equation (28). Evidently, a lower summation of these distances indicates a higher clustering quality, reflecting a tighter grouping of data within clusters. Meanwhile, the sum of intra-cluster distances serves as both the evaluation criterion and the fitness measure for optimizing the clustering process.

$$SD = \sum_{i=1}^{N} \sum_{j=1}^{K} f_{ij} \cdot |o_i - c_j| \tag{28}$$

where N signifies the total count of objects, while K denotes the number of clusters present. The notation o_i represents the i-th object, and c_j stands for the center of the j-th cluster. The coefficient f_{ij} serves as a weight, which is assigned a value of 1 when the i-th object, o_i, belongs to the j-th cluster centered at c_j. Conversely, if o_i does not belong to this cluster, f_{ij} is set to 0. $|o_i - c_j|$ represents the absolute distance between the data point o_i and the cluster center c_j. This mechanism allows for a precise differentiation of object–cluster associations in subsequent calculations or analyses.

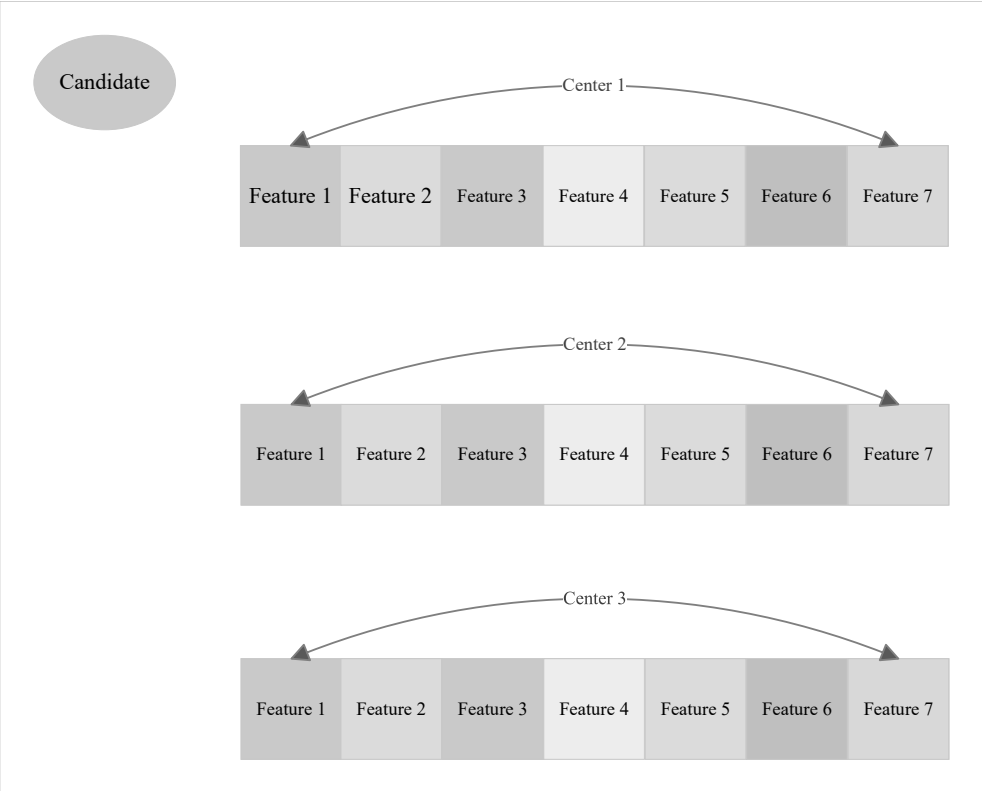

Figure 3. Illustration of Candidate Solutions and Cluster Centers.

Additionally, ER represents the percentage of data objects that are misclassified, indicating the proportion of objects incorrectly placed or assigned to their respective clusters. The clustering performance is quantified by the error rate, where a lower ER signifies superior clustering effectiveness and enhanced optimization capability of the algorithm. The calculation formula for the ER can be formulated as shown in Equation (29).

$$ER = \frac{misclassified}{total\ number} \times 100\% \tag{29}$$

where *misclassified* refers to the count of objects that have been incorrectly classified, and *total number* encompasses the entire set of objects being evaluated.

4. Results and Discussion

In this section, we meticulously compare the performance of the θ-SO against its original versions of the SO algorithm, as well as several other widely recognized optimization algorithms, such as ABC, FMO, PSO, and GA, using various benchmark functions. In addition, we conduct simulation experiments for K-means, and the results show that it can significantly reduce the clustering error rate. The experimental setup is in Table 1. Among them, MATLAB version R2022a is the primary programming language used in this paper.

Table 1. Hardware and Software of Experiments.

Details	Description
Device	Huawei MateBook 16s laptop
Processor	12th Generation Intel® Core™ i9-12900H processor
Memory	LPDDR5 RAM
Operating system	The latest Microsoft Windows 11 system
Software	MATLAB R2022a

4.1. Performance Test of θ-SO

In order to assess the feasibility and effectiveness of the proposed θ-SO-based method, we present the simulation experiments conducted to evaluate the algorithm and detail the algorithms selected for comparison, the specific problems addressed, the datasets utilized for testing, and the parameter values employed in the conducted experiments. The performance of the θ-SO is tested using 28 benchmark functions from CEC2013 [78]. These functions can be categorized into three primary types in Table 2: functions f_1 to f_5 are unimodal functions, f_6 to f_{20} are basic multimodal functions, and f_{21} to f_{28} are composition functions.

Table 2. CEC2013 Function.

No.	Function Name	Function Formula
f_1	Sphere Function	$f_1(\mathbf{x}) = \sum_{i=1}^{D} z_i^2 + f_1 \ ^*, z = x - o$
f_2	Rotated High Conditioned Elliptic Function	$f_2(\mathbf{x}) = \sum_{i=1}^{D} (10^6)^{\frac{i-1}{D-1}} z_i^2 + f_2 \ ^*, z = T_{OSZ}(M_1(x-o))$
f_3	Rotated Bent Cigar Function	$f_3(\mathbf{x}) = z_i^2 + 10^6 \sum_{i=2}^{D} z_i^2 + f_3 \ ^*, z = M_2 T_{asy}^{0.5}(M_1(x-o))$
f_4	Rotated Discus Function	$f_4(\mathbf{x}) = 10^6 z_i^2 + \sum_{i=2}^{D} z_i^2 + f_4 \ ^*, z = T_{OSZ}(M_1(x-o))$
f_5	Different Powers Function	$f_5(\mathbf{x}) = \sqrt{\sum_{i=1}^{D} \lvert z_i \rvert^{2+4\frac{i-1}{D-1}}} + f_5 \ ^*, z = x - o$
f_6	Rotated Rosenbrock's Function	$f_6(\mathbf{x}) = \sum_{i=1}^{D-1} (100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2) + f_6 \ ^*, z = M_1(\frac{2.048(x-o)}{100}) + 1$
f_7	Rotated Schaffers F7 Function	$f_7(\mathbf{x}) = (\frac{1}{D-1} \sum_{i=1}^{D-1} (\sqrt{z_i} + \sqrt{z_i}\sin^2(50 z_i^{0.2})))^2 + f_7 \ ^*,$ $z = \sqrt{y_i^2 + y_{i+1}^2}\ for\ i = 1, \ldots, D, y = \wedge^{10} M_2 T_{asy}^{0.5}(M_1(x-o))$
f_8	Rotated Ackley's Function	$f_8(\mathbf{x}) = -20exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} z_i^2}) - exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi z_i)) + 20 + e + f_8 \ ^*,$ $z = \wedge^{10} M_2 T_{asy}^{0.5}(M_1(x-o))$
f_9	Rotated Weierstrass Function	$f_9(\mathbf{x}) = \sum_{i=1}^{D}(\sum_{k=0}^{kmax}[a^k \cos(2\pi b^k (z_i + 0.5))]) - D\sum_{k=0}^{kmax}[a^k \cos(2\pi b^k \cdot 0.5)] + f_9 \ ^*,$ $a = 0.5, b = 3, kmax = 20, z = \wedge^{10} M_2 T_{asy}^{0.5}(M_1(\frac{0.5(x-o)}{100}))$
f_{10}	Rotated Griewank's Function	$f_{10}(\mathbf{x}) = \sum_{i=1}^{D} \frac{z_i^2}{4000} - \prod_{i=1}^{D} \cos(\frac{z_i}{\sqrt{i}}) + 1 + f_{10} \ ^*, z = \wedge^{100} M_1(\frac{600(x-o)}{100})$
f_{11}	Rastrigin's Function	$f_{11}(\mathbf{x}) = \sum_{i=1}^{D} (z_i^2 - 10\cos(2\pi z_i) + 10) + f_{11} \ ^*, z = \wedge^{10} T_{asy}^{0.2}(T_{osz}(\frac{5.12(x-o)}{100}))$
f_{12}	Rotated Rastrigin's Function	$f_{12}(\mathbf{x}) = \sum_{i=1}^{D} (z_i^2 - 10\cos(2\pi z_i) + 10) + f_{12} \ ^*, z = M_1 \Lambda^{10} M_2 T_{asy}^{0.2}(T_{osz}(M_1 \frac{5.12(x-o)}{100}))$
f_{13}	Non-Continuous Rotated Rastrigin's Function	$f_{13}(\mathbf{x}) = \sum_{i=1}^{D} (z_i^2 - 10\cos(2\pi z_i) + 10) + f_{13} \ ^*, z = M_1 \wedge^{10} M_2 T_{asy}^{0.2}(T_{osz}(y))$

Table 2. *Cont.*

No.	Function Name	Function Formula
f_{14}	Schwefel's Function	$f_{14}(\mathbf{z}) = 418.9829 \times D - \sum_{i=1}^{D} g(z_i) + f_{14}{}^{*}, z = \wedge^{10}(\frac{1000(x-o)}{100}) 4.209687462275036e+002$
f_{15}	Rotated Schwefel's Function	$f_{15}(\mathbf{z}) = 418.9829 \times D - \sum_{i=1}^{D} g(z_i) + f_{15}{}^{*}, z = \wedge^{10} M_1(\frac{1000(x-o)}{100}) 4.209687462275036e+002$
f_{16}	Rotated Katsuura Function	$f_{16}(\mathbf{x}) = \frac{10}{D^2} \prod_{i=1}^{D} (1 + i \sum_{j=1}^{32} \frac{\lvert 2^j z_i - round(2^j z_i) \rvert}{2^j})^{\frac{10}{D^{1.2}}} - \frac{10}{D^2} + f_{16}{}^{*}, z = M_2 \wedge^{100} (M_1 \frac{5(x-o)}{100})$
f_{17}	Lunacek Bi-Rastrigin Function	$f_{17}(\mathbf{x}) = min(\sum_{i=1}^{D} (\hat{x}_i - \mu_0)^2, dD + s \sum_{i=1}^{D} (\hat{x}_i - \mu_1)^2) + 10(D - \sum_{i=1}^{D} \cos(2\pi \hat{z}_i)) + f_{17}{}^{*}, z = \wedge^{100} (\hat{x} - \mu_0)$
f_{18}	Rotated Lunacek Bi-Rastrigin Function	$f_{18}(\mathbf{x}) = min(\sum_{i=1}^{D} (\hat{x}_i - \mu_0)^2, dD + s \sum_{i=1}^{D} (\hat{x}_i - \mu_1)^2) + 10(D - \sum_{i=1}^{D} \cos(2\pi \hat{z}_i)) + f_{18}{}^{*},$ $z = M_2 \wedge^{100} (M_1 (\hat{x} - \mu_0))$
f_{19}	Expanded Griewank's plus Rosenbrock's Function	$f_{19}(\mathbf{x}) = g_1(g_2(z_1, z_2)) + g_1(g_2(z_2, z_3)) + \ldots + g_1(g_2(z_{D-1}, z_D)) + g_1(g_2(z_D, z_1)) + f_{19}{}^{*},$ $g_1(x) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos(\frac{x_i}{\sqrt{i}}) + 1, z = M_1(\frac{5.12(x-o)}{100}) + 1$
f_{20}	Expanded Scaffer's F6 Function	$f_{20}(\mathbf{x}) = g(z_1, z_2) + g(z_2, z_3) + \ldots + g(z_{D-1}, z_D) + g(z_D, z_1) + f_{20}{}^{*},$ $g(x, y) = 0.5 + \frac{\sin^2(\sqrt{x^2 + y^2}) - 0.5}{(1 + 0.001(x^2 + y^2))^2}, z = M_2 T_{asy}^{0.5}(M_1(x - o))$
f_{21}	Composition Function 1 (n = 5, Rotated)	$f_{21}(\mathbf{x}) = \sum_{i=1}^{n} \{\omega_i * [\lambda_i g_i(x) + bias_i]\} + f_{21}^{*}, f_i' = f_i - f_i^{*}, g_1 = f_6', g_2 = f_5', g_3 = f_3', g_4 = f_4', g_5 = f_1'$
f_{22}	Composition Function 2 (n = 3, Unrotated)	$f_{22}(\mathbf{x}) = \sum_{i=1}^{n} \{\omega_i * [\lambda_i g_i(x) + bias_i]\} + f_{22}^{*}, f_i' = f_i - f_i^{*}, g_{1-3} = f_{14}'$
f_{23}	Composition Function 3 (n = 3, Rotated)	$f_{23}(\mathbf{x}) = \sum_{i=1}^{n} \{\omega_i * [\lambda_i g_i(x) + bias_i]\} + f_{23}^{*}, f_i' = f_i - f_i^{*}, g_{1-3} = f_{15}'$
f_{24}	Composition Function 4 (n = 3, Rotated)	$f_{24}(\mathbf{x}) = \sum_{i=1}^{n} \{\omega_i * [\lambda_i g_i(x) + bias_i]\} + f_{24}^{*}, f_i' = f_i - f_i^{*}, g_1 = f_{15}', g_2 = f_{12}', g_3 = f_9', \sigma[20, 20, 20]$
f_{25}	Composition Function 5 (n = 3, Rotated)	$f_{25}(\mathbf{x}) = \sum_{i=1}^{n} \{\omega_i * [\lambda_i g_i(x) + bias_i]\} + f_{25}^{*}, f_i' = f_i - f_i^{*}, g_1 = f_{15}', g_2 = f_{12}', g_3 = f_9', \sigma[10, 30, 50]$
f_{26}	Composition Function 6 (n = 5, Rotated)	$f_{26}(\mathbf{x}) = \sum_{i=1}^{n} \{\omega_i * [\lambda_i g_i(x) + bias_i]\} + f_{26}^{*}, f_i' = f_i - f_i^{*}, g_1 = f_{15}', g_2 = f_{12}', g_3 = f_2', g_4 = f_9', g_5 = f_{10}'$
f_{27}	Composition Function 7 (n = 5, Rotated)	$f_{27}(\mathbf{x}) = \sum_{i=1}^{n} \{\omega_i * [\lambda_i g_i(x) + bias_i]\} + f_{27}^{*}, f_i' = f_i - f_i^{*}, g_1 = f_{10}', g_2 = f_{12}', g_3 = f_{15}', g_4 = f_9', g_5 = f_1'$
f_{28}	Composition Function 8 (n = 5, Rotated)	$f_{28}(\mathbf{x}) = \sum_{i=1}^{n} \{\omega_i * [\lambda_i g_i(x) + bias_i]\} + f_{28}^{*}, f_i' = f_i - f_i^{*}, g_1 = f_{19}', g_2 = f_7', g_3 = f_{15}', g_4 = f_{20}', g_5 = f_1'$

The CEC2013 test suite is the most recognized, classic, and widely used test set in the field of heuristic algorithms. CEC2013 encompasses a variety of function types, including unimodal, multimodal, hybrid, and composite functions, enabling it to simulate the complexity and difficulty of various optimization problems. This diversity and comprehensiveness make CEC2013 more reliable and accurate in evaluating optimization algorithms. Additionally, as an early-released benchmark test suite, CEC2013 has been tested by time and is widely recognized and used by researchers. Its well-designed function set can comprehensively evaluate the performance of optimization algorithms, thus occupying an irreplaceable classic position in the research and development of optimization algorithms. Liu et al. utilized the CEC2013 datasets to demonstrate that the t-location–scale distribution exhibits superior optimization performance compared to other heuristic algorithms [71]. Zheng et al. employed the CEC2013 datasets to showcase that compact snake optimization outperforms other heuristic algorithms in terms of optimization performance [66].

Table 3 presents a comparative analysis of the performance of θ-SO versus that of conventional heuristic algorithms. To ensure a rigorous and equitable evaluation, the experimental setup maintained consistency across all algorithms, employing a population size of 30 individuals for every one of the six algorithms across all problems. Ensuring a consistent number of individuals across all algorithms facilitates a more accurate and equitable assessment of their optimization capabilities. Therefore, when comparing the performance of different algorithms, we adopted this approach, which essentially involves controlling variables. By maintaining a uniform population size, we eliminate a potential variable that might otherwise influence the results, allowing us to focus solely on the inherent strengths and weaknesses of each algorithm in terms of their ability to find optimal solutions. Furthermore, all the compared algorithms were configured to run for a maximum of 1000 iterations with a dimension of 30. In adherence to the standards set by CEC2013, the search range for all these algorithms was confined within the specified interval of $[-100, 100]$.

Table 3. Comparative Analysis of θ-SO Algorithm and Traditional Heuristic Algorithms.

Function	θ-SO	SO	ABC	FMO	PSO	GA
f_1	-1.39×10^3	-1.40×10^3	1.07×10^4	-9.02×10^1	-1.40×10^3	9.24×10^4
f_2	1.66×10^7	1.78×10^7	3.83×10^8	9.54×10^6	6.18×10^6	2.62×10^9
f_3	6.16×10^9	4.48×10^9	1.09×10^{17}	8.05×10^9	5.96×10^9	7.84×10^{22}
f_4	4.91×10^4	4.91×10^4	7.28×10^4	7.52×10^4	2.07×10^4	3.47×10^6
f_5	-9.89×10^2	-9.79×10^2	7.98×10^3	-9.80×10^2	-9.49×10^2	8.02×10^4
f_6	-8.48×10^2	-7.93×10^2	1.12×10^3	-8.42×10^2	-8.30×10^2	2.42×10^4
f_7	-6.49×10^2	-6.66×10^2	1.69×10^5	-6.19×10^2	-3.80×10^2	1.12×10^8
f_8	-6.79×10^2	-6.79×10^2	-6.79×10^2	-6.79×10^2	-6.79×10^2	-6.79×10^2
f_9	-5.70×10^2	-5.68×10^2	-5.60×10^2	-5.59×10^2	-5.62×10^2	-5.53×10^2
f_{10}	-3.76×10^2	-4.42×10^2	1.69×10^3	-3.80×10^2	-4.93×10^2	1.35×10^4
f_{11}	-3.21×10^2	-3.05×10^2	7.60×10^1	-1.35×10^2	7.80×10^1	1.15×10^3
f_{12}	-7.41×10^1	-9.27×10^1	1.58×10^2	3.32×10^1	1.26×10^2	1.24×10^3
f_{13}	7.83×10^1	5.51×10^1	2.46×10^2	1.28×10^2	3.37×10^2	1.33×10^3
f_{14}	1.57×10^3	1.70×10^3	4.86×10^3	7.46×10^3	4.38×10^3	9.76×10^3
f_{15}	5.36×10^3	7.59×10^3	7.70×10^3	7.83×10^3	4.66×10^3	9.26×10^3
f_{16}	2.02×10^2	2.03×10^2	2.03×10^2	2.03×10^2	2.02×10^2	2.05×10^2
f_{17}	4.47×10^2	4.57×10^2	7.50×10^2	7.14×10^2	7.69×10^2	3.04×10^3
f_{18}	5.85×10^2	6.74×10^2	9.13×10^2	8.03×10^2	8.97×10^2	3.11×10^3
f_{19}	5.08×10^2	5.09×10^2	2.58×10^5	5.14×10^2	5.43×10^2	1.31×10^7
f_{20}	6.13×10^2	6.13×10^2	6.15×10^2	6.15×10^2	6.15×10^2	6.15×10^2
f_{21}	9.73×10^2	9.74×10^2	2.76×10^3	1.07×10^3	1.02×10^3	7.24×10^3
f_{22}	2.69×10^3	3.14×10^3	5.94×10^3	8.85×10^3	6.73×10^3	1.10×10^4
f_{23}	6.12×10^3	8.14×10^3	9.25×10^3	9.41×10^3	6.91×10^3	1.08×10^4
f_{24}	1.28×10^3	1.28×10^3	1.30×10^3	1.30×10^3	1.34×10^3	1.62×10^3
f_{25}	1.39×10^3	1.39×10^3	1.41×10^3	1.42×10^3	1.48×10^3	1.55×10^3
f_{26}	1.52×10^3	1.50×10^3	1.49×10^3	1.51×10^3	1.58×10^3	1.66×10^3
f_{27}	2.31×10^3	2.25×10^3	2.63×10^3	2.51×10^3	2.82×10^3	3.29×10^3
f_{28}	2.09×10^3	2.44×10^3	7.28×10^3	2.45×10^3	5.79×10^3	1.31×10^4

In Table 3, the cells marked in bold signify that the method exhibits excellent performance in the benchmark function. The significant finding from Table 3 is that the θ-SO emerges with the highest number of best results. The data from experiments reveal that the θ-SO has better test performance than SO in 18 functions, better performance than ABC in 26 functions, better performance than FMO in 24 functions, better performance than PSO in 20 functions, and outperforms all of GA. This indicates its effectiveness and robustness. These findings strongly suggest that the results of θ-SO algorithm, which incorporates phase angle encoding of the SO algorithm, not only can obtain better results on most testing functions but also remarkably enhances the optimization performance compared to other algorithms. It is effective and robust and presents a profound opportunity to explore viable spaces for bolstering evolutionary performance during the ongoing evolutionary process.

Next, we further demonstrate the effectiveness of the algorithm through the evaluation of convergence curves. Figure 4 presents the convergence curves of several benchmark functions, offering a visual representation of their performance.

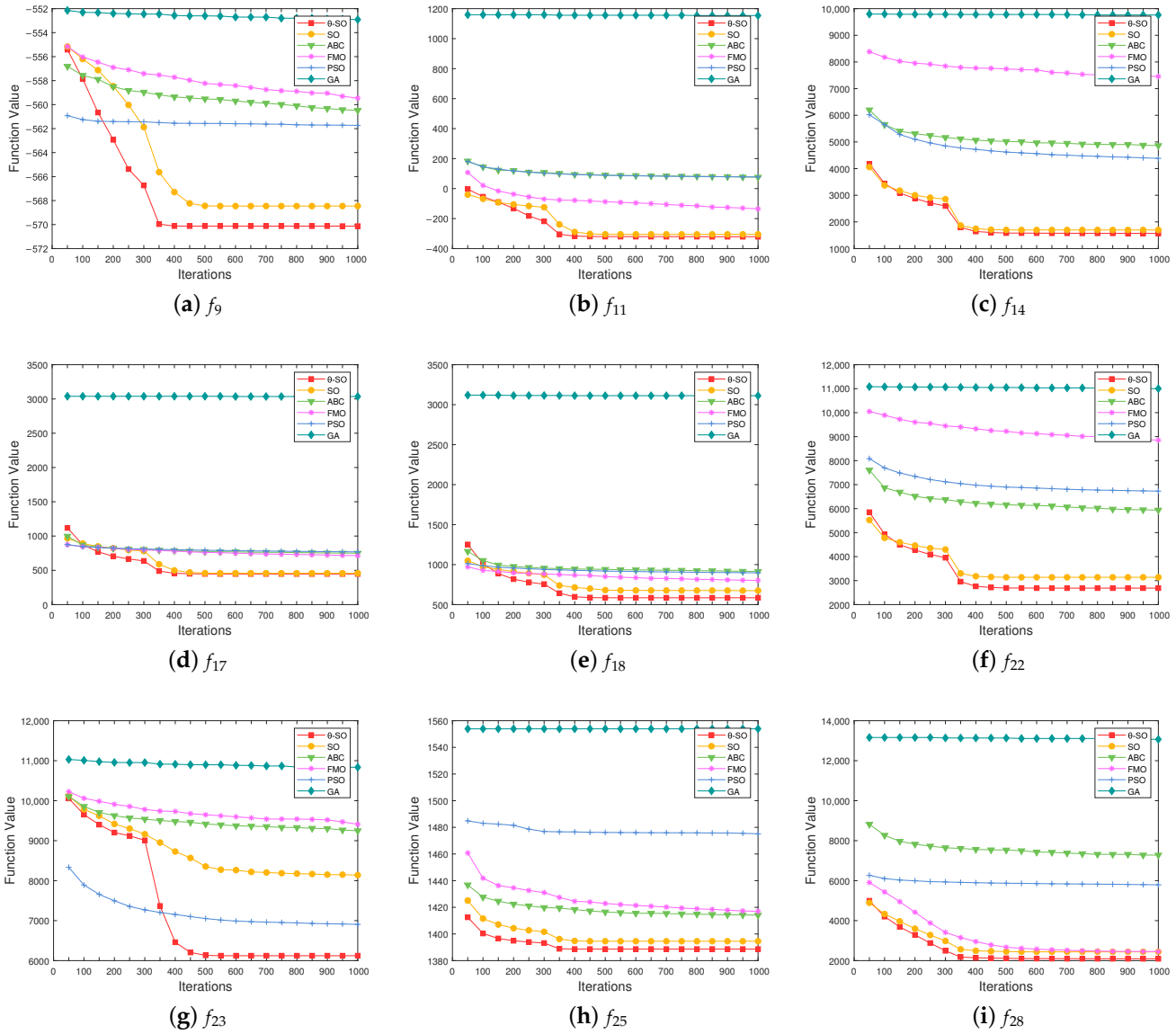

Figure 4. Performance Comparative Analysis of θ-SO and Popular Heuristic Algorithms.

From Figure 4, it can be observed that in comparison to other algorithms, the proposed algorithm exhibits good performance. This means that θ-SO can find smaller function values and converges faster on test functions f_9 , f_{11}, f_{14}, f_{17}, f_{18}, f_{22}, f_{23}, f_{25}, and f_{28}. On the basic multimodal function f_9, PSO converges faster than θ-SO during the initial stage of iteration. However, the function value of θ-SO rapidly decreases before reaching the 350th iteration and consistently approximates the best optimal value after the 200th iteration, exploiting the global optimal value to achieve higher convergence accuracy.

On the f_{11}, f_{14}, and f_{17} functions, the convergence curve of θ-SO is close to that of SO and even briefly surpassed by SO during the early iteration stages from Figure 4, but Table 3 clearly shows that θ-SO ultimately achieves the optimal value. Except for the poor performance of GA, other algorithms like SO, ABC, FMO, PSO, and θ-SO can all locate a good value at the beginning of the iteration on the f_{18} function. On the composition function f_{22} and f_{28}, θ-SO outperforms all other algorithms and sustains its advantage after surpassing them around the 150th iteration. On the function f_{23}, despite not converging best initially, θ-SO is able to effectively escape local extrema and swiftly converge towards the optimal value as the iteration exceeds 350 with effective improvement. The θ-SO outperforms the other algorithms consistently throughout the entire iteration process on the function f_{25}.

The good performance in the composition functions indicates that θ-SO effectively avoids local extrema and achieves high convergence accuracy when dealing with complex functions.

4.2. Application of θ-SO to Clustering

To assess the efficacy of the novel algorithm, it was deployed within the realm of clustering challenges, leveraging a diverse selection of four datasets sourced from reputable machine learning archives: Iris, Wine, Seeds, and Contraceptive Method Choice (CMC). Because each dataset exhibits a distinct level of complexity, Kan Ni et al. conducted a series of experiments utilizing datasets including Iris, Glass, CMC, and Wine, with the aim of assessing and contrasting the clustering efficacy of the differential evolution artificial bee colony-fuzzy clustering algorithm against other methods [79]. Similarly, to assess the clustering performance of the FCM algorithm with the improved density-sensitive distance, datasets like Wine, Iris, Seeds, CMC, and others were used [80].

The four clustering datasets are widely recognized within the data science and machine learning communities as being both typical and frequently used. Due to their well-defined structures and clear distinctions between different clusters, these datasets are often employed as benchmarks to evaluate the performance of clustering algorithms. Consequently, this paper has chosen to utilize these four datasets. When employing the θ-SO for data clustering, the candidate solution to the clustering problem is represented as a 1-dimensional array. In this array, each candidate solution comprises K initial cluster centers, with each individual unit in the array corresponding to a dimension of the cluster center.

The summary in Table 4 outlines the key features of the utilized datasets, including their names, the number of clusters, the number of features, and the number of data objects. The Iris dataset was divided into three classifications, each containing 50 samples. The Wine dataset was divided into classifications with 59, 71, and 48 samples, respectively. The Seeds dataset was divided into three classifications, each comprising 70 samples. The CMC dataset was divided into classifications with 629, 334, and 510 samples, respectively. The θ-SO algorithm is compared with its original version of the SO algorithm, as well as several well-known algorithms, including ABC, FMO, PSO, and GA, which we have used for comparison to test its performance.

Table 4. Key Features of the Test Datasets.

Datasets	Clusters	Features	Objects
Iris	3	4	150 (50, 50, 50)
Wine	3	13	178 (59, 71, 48)
Seeds	3	7	210 (70, 70, 70)
CMC	3	9	1473 (629, 334, 510)

In this series of experiments, various datasets undergo clustering using distinct algorithms, and the ER is computed for each individual experiment. Table 5 presents the mean error rate achieved by the clustering algorithms, focusing on the best 10 iterations out of 100 independent simulation runs conducted on the test datasets.

Table 5. Comparison of Clustering Error Rates Across Different Test Datasets.

Datasets	K-Means	ABC	FMO	PSO	GA	SO	θ-SO
Iris	13.42%	9.27%	10.13%	10.00%	19.73%	9.33%	8.67%
Wine	31.14%	28.60%	30.79%	35.67%	29.27%	28.99%	28.54%
Seeds	10.48%	10.71%	10.48%	15.62%	24.52%	10.48%	10.43%
CMC	60.68%	60.16%	60.29%	60.50%	58.40%	60.11%	58.01%

In this table, the horizontal rows represent the algorithms, specifically K-means, ABC, FMO, PSO, GA, SO, and the θ-SO algorithm, while the vertical columns represent the datasets, namely, Iris, Wine, Seeds, and CMC. Apart from K-mean algorithms, heuristic

algorithms are all based on a combination with clustering algorithms to perform classification, thereby obtaining the results. Worth mentioning, the θ-SO algorithm consistently outperforms the rest, demonstrating the lowest average error rate across all test datasets, as clearly indicated in Table 5. On the Iris dataset, it surpasses the original K-means algorithm by a substantial margin of nearly 5 percentage points. Similarly, on the Wine dataset, the θ-SO algorithm excels, outperforming the PSO by approximately 7 percentage points. The Seeds dataset further underscores the superiority of the algorithm, as it outperforms the GA algorithm by an impressive 14 percentage points and the PSO by nearly 5 percentage points. Notably, even on the CMC data, where the difference in error rates is less pronounced, the proposed θ-SO algorithm still manages to outperform K-means, ABC, FMO, PSO, and SO by a consistent two percentage points, demonstrating its robust and consistent performance advantages.

Next, Table 6 in the article presents a summary of the time required for the experiments.

Table 6. Comparison of Clustering Run Time(s) Across Different Test Datasets.

Datasets	K-Means	ABC	FMO	PSO	GA	SO	θ-SO
Iris	70	46	46	**45**	47	46	46
Wine	101	77	85	67	75	**60**	**60**
Seeds	96	**65**	66	**65**	66	66	**65**
CMC	1092	1959	1144	1174	1192	896	**867**

From Table 6, we can observe that on the Iris dataset, the running time of θ-SO is similar to that of ABC, FMO, and other algorithms, all at 46 s. Although it is slightly longer by 1 s compared to PSO, its performance is relatively good. On the Wine dataset, the running time of θ-SO is 60 s, tied for the shortest with SO. On the Seeds dataset, although the running times of several algorithms are relatively close, the running time of θ-SO, which is equal to that of ABC and PSO at 65 s, still ranks among the shortest. On the CMC dataset, the running time of θ-SO is 867 s, far lower than that of other algorithms. These data demonstrate that θ-SO not only boasts efficient running times but also exhibits relatively stable performance across different datasets.

Meanwhile, Table 7 presents a concise overview of the SD achieved by the clustering algorithms, organized with algorithms listed horizontally and datasets vertically for enhanced clarity and comparison. This summary concisely summarizes the key performance metrics for each algorithm, including the mean, optimal, worst, and standard deviation of the achieved distances.

Table 7. The Sum of Intra-Cluster Distances Across Multiple Datasets Using Various Algorithms.

Datasets	Criteria	K-Means	ABC	FMO	PSO	GA	SO	θ-SO
Iris	Mean	105.7290	98.1073	96.8642	96.8297	181.7896	96.9365	96.7558
	Optimal	97.3259	96.6965	96.7429	96.6734	165.6193	96.6599	96.6555
	Worst	128.4042	100.2745	97.1645	97.9127	198.2818	97.6482	96.9233
	Std	12.3876	1.0284	0.1429	0.3834	9.3276	0.4094	0.1021
Wine	Mean	16,963.0450	16,538.4289	16,318.0643	16,323.6800	18,900.4460	16,307.2794	16,297.0141
	Optimal	16,555.6794	16,348.4203	16,308.8201	16,310.9651	17,697.5615	16,297.3842	16,294.0796
	Worst	23,755.0495	16,861.4818	16,331.1883	16,339.6993	20,372.7736	16,327.9950	16,300.3322
	Std	1180.6942	162.3547	8.6356	9.9501	862.0817	8.6233	2.3578
Seeds	Mean	587.9040	323.8719	312.5240	312.4245	484.7676	312.2745	311.8223
	Optimal	587.3186	312.5663	312.0396	312.1142	402.9336	311.8838	311.8100
	Worst	588.7820	352.1443	313.2203	312.8770	533.8758	313.0029	311.8358
	Std	0.7557	15.0272	0.3943	0.2213	40.0170	0.3469	0.0110
CMC	Mean	5543.4234	5660.5799	5601.3604	5583.9164	7626.8047	5553.9000	5534.8952
	Optimal	5542.1821	5578.1252	5579.5982	5565.0366	7045.7836	5539.3442	5533.2193
	Worst	5545.3334	5829.7017	5628.1325	5612.5687	8128.6454	5566.4741	5536.4578
	Std	1.5238	89.0009	15.9879	16.5816	326.0442	10.3200	1.2966

For the Iris dataset, the θ-SO algorithm achieved the mean, optimal and worst values of 96.75583, 96.65555, and 96.92326, respectively. Additionally, the standard deviation value of merely 0.1 is much lower than 12.38759 for K-means and 9.32758 for GA, indicating the superior performance and stability of the θ-SO algorithm compared to other methods. For the Wine dataset, the mean, optimal, and worst values for the Wine dataset were 16,297.01413, 16,294.07960, and 16,300.33225, respectively. Although the standard deviation of 2.35784 is not as low as that for the Iris dataset, it is still lower than that of other algorithms. In the Seeds dataset, the θ-SO algorithm achieved a worst value of 311.83579, which is lower than the optimal results achieved by all other algorithms. Similarly, the same applies to the CMC dataset. Based on the aforementioned results, it is evident that the proposed algorithm surpasses the other test algorithms in four out of the examined datasets. Its ability to discover high-quality solutions, coupled with a low standard deviation, underscores its superiority. In simpler terms, the θ-SO algorithm consistently converges towards the global optimum across all iterations, whereas the other algorithms risk becoming stuck in local optimum solutions.

To validate the minimized sum of intra-cluster distances as outlined in Table 7, the optimal centroids identified by the θ-SO algorithm for the test datasets are presented in Tables 8–11. By systematically allocating the data points within each datasets to their respective centroids listed in Table 8, the ideal values achieved in Table 7 are replicated. Table 8, Table 9, Table 10, and Table 11, respectively, present the centroids for the Iris, Wine, Seeds, and CMC datasets.

Table 8. Optimal Centroids for Iris Datasets by the θ-SO Algorithm.

Center 1	Center 2	Center 3
6.73368	5.01222	5.91910
3.06867	3.40296	2.79549
5.63239	1.47179	4.41112
2.10688	0.23369	1.40133

Table 9. Optimal Centroids for Wine Datasets by the θ-SO Algorithm.

Center 1	Center 2	Center 3
11.03000	14.01008	11.03000
5.46730	3.30646	2.32901
3.23000	3.23000	2.82646
10.78553	20.76729	10.60000
99.86794	93.06828	98.31607
0.98000	1.69196	0.98000
5.08000	3.67211	2.99557
0.13001	0.66000	0.19378
0.41000	1.35218	1.77120
3.08268	1.28000	1.28000
0.48000	0.48000	1.71000
1.29493	4.00000	1.69149
1152.27248	462.92829	692.11271

Table 10. Optimal Centroids for Seeds Datasets by the θ-SO Algorithm.

Center 1	Center 2	Center 3
11.92009	14.60652	18.76627
13.29476	14.45346	16.30508
0.85068	0.87806	0.88042
5.24514	5.56259	6.21840
2.85396	3.27796	3.74920
4.66124	2.64018	3.32926
5.11177	5.15714	6.07750

Table 11. Optimal Centroids for CMC Datasets by the θ-SO Algorithm.

Center 1	Center 2	Center 3
24.42210	43.58460	33.54514
3.07447	4.00000	3.11408
3.50320	3.46291	3.53122
1.78841	4.53441	3.66586
0.92241	0.79996	0.81364
0.79809	0.77414	1.00000
2.28357	1.77331	2.10225
2.97353	3.51501	3.26426
0.04313	0.06957	0.03841

For instance, when we consider the Seeds datasets in Table 10, it typically contains three cluster centers and seven features. The values of the first cluster center are 11.92009, 13.29476, 0.85068, 5.24514, 2.85396, 4.66124, and 5.11177; the values of the second cluster center are 14.60652, 14.45346, 0.87806, 5.56259, 3.27796, 2.64018, and 5.15714; and the values of the third cluster center are 18.76627, 16.30508, 0.88042, 6.21840, 3.74920, 3.32926, and 6.07750. In the context of validating the results, the optimal value for the SD achieved by the θ-SO algorithm, as reported in Table 7 (specifically, 311.8100), should be replicated by accurately assigning all data points in the Seeds datasets to their nearest centers identified in Table 10. Failure to achieve this target value would indicate a potential discrepancy in either the best values presented in Table 7, the optimal centroids listed in Table 10, or both. This same verification process can be extended to any other test datasets to ensure the accuracy and consistency of the clustering outcomes.

In general, the optimized K-means algorithm can address a variety of practical problems. For instance, in the commercial sector, it can be utilized for customer segmentation by collecting multidimensional feature data such as purchase history, browsing behavior, age, gender, and so on, to categorize customers into different groups, enabling enterprises to formulate more targeted marketing strategies. In the field of natural language processing, it can be applied to document classification and topic modeling. In the realm of image processing, it can treat pixels in images as data points and perform clustering based on their color, brightness, and other features.

5. Conclusions

In this paper, we present a novel optimization algorithm named θ-SO, which is grounded in the foundations of the existing SO algorithm. The θ-SO algorithm employs a new strategy that redefines its traditional search space by mapping it onto a polar domain. This new mapping approach not only simplifies the architecture of the algorithm, making it easier to implement, but also eliminates the need for cumbersome parameter tuning, further enhancing its usability. The θ-SO algorithm underwent performance evaluation using 28 test functions from the CEC2013 datasets. On more than two-thirds of the test functions, θ-SO exhibits superior performance compared to other algorithms. Additionally, our integration of the θ-SO algorithm with the K-means clustering algorithm on various test sets showed improvement in clustering accuracy during simulation experiments. Looking ahead, we aspire to devise groundbreaking strategies and delve deeper into the untapped potential of fusing innovative algorithms with the K-means clustering approach, anticipating noteworthy achievements and groundbreaking outcomes from these endeavors.

Author Contributions: D.X.: contributed to the conceptualization, methodology, data curation, formal analysis, visualization, and prepared the original draft of the writing. S.-Y.P.: involved in validation, investigation, formal analysis, and conducted writing—review and editing. N.L.: contributed to the methodology, resources, formal analysis, and conducted writing—review and editing. S.-K.L.: involved in investigation, resources, formal analysis, and provided supervision. W.-M.Z.: provided supervision, funding acquisition, and project administration. All authors have read and agreed to the published version of the manuscript.

Funding: This research is funded by the National Natural Science Foundation of China, No. 61932005.

Data Availability Statement: The data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

MA	Metaheuristic algorithm
EA	Evolutionary algorithm
GA	Genetic algorithm
EP	Evolutionary programming
ES	Evolutionary strategies
GP	Genetic programming
DE	Differential evolution
BBO	Biogeography-based optimization
SI	Swarm intelligence
PSO	Particle swarm optimization
WOA	Whale optimization algorithm
ABC	Artificial bee colony
ACO	Colony optimization
GWO	Grey wolf optimization
FA	Firefly algorithm
SOA	Seagull optimization algorithm
ALO	Ant lion optimizer
SSA	Squirrel search algorithm
CS	Cuckoo search
CSA	Crow search algorithm
BA	Bat algorithm
CGO	Chaos game optimization
SRO	Search and rescue optimization
COA	Cognitive behavior optimization algorithm
AOA	Archimedes optimization algorithm
FWA	Fireworks algorithm
MVO	Multiverse optimizer
LA	Lichtenberg algorithm
PRO	Poor and rich optimization
AOS	Atomic orbital search
SCA	Sine cosine algorithm
HMS	Human mental search
SO	Snake optimization
θ-SO	Phase-angle-encoded snake optimization
K-means	K-means clustering algorithm
SSE	Sum of squared error
FMO	Fish migration optimization
CMC	Contraceptive Method Choice
SD	Sum of intra-cluster distance
ER	Error rate

References

1. Bianchi, L.; Dorigo, M.; Gambardella, L.M.; Gutjahr, W.J. A survey on metaheuristics for stochastic combinatorial optimization. *Nat. Comput.* **2009**, *8*, 239–287. [CrossRef]
2. Abualigah, L.; Gandomi, A.H.; Elaziz, M.A.; Hussien, A.G.; Khasawneh, A.M.; Alshinwan, M.; Houssein, E.H. Nature-inspired optimization algorithms for text document clustering—A comprehensive analysis. *Algorithms* **2020**, *13*, 345. [CrossRef]
3. Pasha, J.; Dulebenets, M.A.; Kavoosi, M.; Abioye, O.F.; Wang, H.; Guo, W. An optimization model and solution algorithms for the vehicle routing problem with a "factory-in-a-box". *IEEE Access* **2020**, *8*, 134743–134763. [CrossRef]
4. Adetunji, K.E.; Hofsajer, I.W.; Abu-Mahfouz, A.M.; Cheng, L. A review of metaheuristic techniques for optimal integration of electrical units in distribution networks. *IEEE Access* **2020**, *9*, 5046–5068. [CrossRef]

5. Shukla, A.K.; Tripathi, D.; Reddy, B.R.; Chandramohan, D. A study on metaheuristics approaches for gene selection in microarray data: Algorithms, applications and open challenges. *Evol. Intell.* **2020**, *13*, 309–329. [CrossRef]
6. Slowik, A.; Kwasnicka, H. Nature inspired methods and their industry applications—Swarm intelligence algorithms. *IEEE Trans. Ind. Inform.* **2017**, *14*, 1004–1015. [CrossRef]
7. Blum, C.; Roli, A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv. (CSUR)* **2003**, *35*, 268–308. [CrossRef]
8. Hashim, F.A.; Hussien, A.G. Snake Optimizer: A novel meta-heuristic optimization algorithm. *Knowl.-Based Syst.* **2022**, *242*, 108320. [CrossRef]
9. Zhao, S.; Zhang, T.; Ma, S.; Chen, M. Dandelion Optimizer: A nature-inspired metaheuristic algorithm for engineering applications. *Eng. Appl. Artif. Intell.* **2022**, *114*, 105075. [CrossRef]
10. Liu, L.; Fei, T.; Zhu, Z.; Wu, K.; Zhang, Y. A survey of evolutionary algorithms. In Proceedings of the 2023 4th International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE), Hangzhou, China, 25–27 August 2023; pp. 22–27.
11. Zbigniew, M. Genetic algorithms+ data structures= evolution programs. *Comput. Stat.* **1996**, *24*, 372–373.
12. Holland, J.H. Genetic algorithms. *Sci. Am.* **1992**, *267*, 66–73. [CrossRef]
13. Alhijawi, B.; Awajan, A. Genetic algorithms: Theory, genetic operators, solutions, and applications. *Evol. Intell.* **2023**, *17*, 1245–1256. [CrossRef]
14. Storn, R.; Price, K. Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
15. Ma, H.; Simon, D.; Siarry, P.; Yang, Z.; Fei, M. Biogeography-based optimization: A 10-year review. *IEEE Trans. Emerg. Top. Comput. Intell.* **2017**, *1*, 391–407. [CrossRef]
16. Pant, M.; Zaheer, H.; Garcia-Hernandez, L.; Abraham, A. Differential Evolution: A review of more than two decades of research. *Eng. Appl. Artif. Intell.* **2020**, *90*, 103479.
17. Deng, W.; Shang, S.; Cai, X.; Zhao, H.; Song, Y.; Xu, J. An improved differential evolution algorithm and its application in optimization problem. *Soft Comput.* **2021**, *25*, 5277–5298. [CrossRef]
18. Chakraborty, A.; Kar, A.K. Swarm intelligence: A review of algorithms. In *Nature-Inspired Computing and Optimization: Theory and Applications*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 475–494.
19. Yang, X.S.; Deb, S.; Zhao, Y.X.; Fong, S.; He, X. Swarm intelligence: Past, present and future. *Soft Comput.* **2018**, *22*, 5923–5933. [CrossRef]
20. Cao, L.; Cai, Y.; Yue, Y. Swarm intelligence-based performance optimization for mobile wireless sensor networks: Survey, challenges, and future directions. *IEEE Access* **2019**, *7*, 161524–161553. [CrossRef]
21. Nayak, J.; Swapnarekha, H.; Naik, B.; Dhiman, G.; Vimal, S. 25 years of particle swarm optimization: Flourishing voyage of two decades. *Arch. Comput. Methods Eng.* **2023**, *30*, 1663–1725. [CrossRef]
22. Fidanova, S.; Fidanova, S. Ant colony optimization. In *Ant Colony Optimization and Applications*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 3–8.
23. Kaya, E.; Gorkemli, B.; Akay, B.; Karaboga, D. A review on the studies employing artificial bee colony algorithm to solve combinatorial optimization problems. *Eng. Appl. Artif. Intell.* **2022**, *115*, 105311. [CrossRef]
24. Almufti, S.M.; Ahmad, H.B.; Marqas, R.B.; Asaad, R.R. Grey wolf optimizer: Overview, modifications and applications. *Int. Res. J. Sci. Technol. Educ. Manag.* **2021**, *1*, 44–56.
25. Rana, N.; Latiff, M.S.A.; Abdulhamid, S.M.; Chiroma, H. Whale optimization algorithm: A systematic review of contemporary applications, modifications and developments. *Neural Comput. Appl.* **2020**, *32*, 16245–16277. [CrossRef]
26. Kumar, V.; Kumar, D. A systematic review on firefly algorithm: Past, present, and future. *Arch. Comput. Methods Eng.* **2021**, *28*, 3269–3291. [CrossRef]
27. Dhiman, G.; Kumar, V. Seagull optimization algorithm: Theory and its applications for large-scale industrial engineering problems. *Knowl.-Based Syst.* **2019**, *165*, 169–196. [CrossRef]
28. Abualigah, L.; Shehab, M.; Alshinwan, M.; Mirjalili, S.; Elaziz, M.A. Ant lion optimizer: A comprehensive survey of its variants and applications. *Arch. Comput. Methods Eng.* **2021**, *28*, 1397–1416. [CrossRef]
29. Dhaini, M.; Mansour, N. Squirrel search algorithm for portfolio optimization. *Expert Syst. Appl.* **2021**, *178*, 114968. [CrossRef]
30. Guerrero-Luis, M.; Valdez, F.; Castillo, O. A review on the cuckoo search algorithm. In *Fuzzy Logic Hybrid Extensions of Neural and Optimization Algorithms: Theory and Applications*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 113–124.
31. Hussien, A.G.; Amin, M.; Wang, M.; Liang, G.; Alsanad, A.; Gumaei, A.; Chen, H. Crow search algorithm: Theory, recent advances, and applications. *IEEE Access* **2020**, *8*, 173548–173565. [CrossRef]
32. Agarwal, T.; Kumar, V. A systematic review on bat algorithm: Theoretical foundation, variants, and applications. *Arch. Comput. Methods Eng.* **2021**, *29*, 2707–2736. [CrossRef]
33. Guo, B.; Zhuang, Z.; Pan, J.S.; Chu, S.C. Optimal design and simulation for PID controller using fractional-order fish migration optimization algorithm. *IEEE Access* **2021**, *9*, 8808–8819. [CrossRef]
34. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95—International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948.
35. Eberhart, R.; Kennedy, J. A new optimizer using particle swarm theory. In Proceedings of the MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, 4–6 October 1995; pp. 39–43.

36. Talatahari, S.; Azizi, M. Chaos game optimization: A novel metaheuristic algorithm. *Artif. Intell. Rev.* **2021**, *54*, 917–1004. [CrossRef]
37. Anuradha, D.; Subramani, N.; Khalaf, O.I.; Alotaibi, Y.; Alghamdi, S.; Rajagopal, M. Chaotic search-and-rescue-optimization-based multi-hop data transmission protocol for underwater wireless sensor networks. *Sensors* **2022**, *22*, 2867. [CrossRef] [PubMed]
38. Li, M.; Zhao, H.; Weng, X.; Han, T. Cognitive behavior optimization algorithm for solving optimization problems. *Appl. Soft Comput.* **2016**, *39*, 199–222. [CrossRef]
39. Hashim, F.A.; Hussain, K.; Houssein, E.H.; Mabrouk, M.S.; Al-Atabany, W. Archimedes optimization algorithm: A new metaheuristic algorithm for solving optimization problems. *Appl. Intell.* **2021**, *51*, 1531–1551. [CrossRef]
40. Zare, M.; Narimani, M.R.; Malekpour, M.; Azizipanah-Abarghooee, R.; Terzija, V. Reserve constrained dynamic economic dispatch in multi-area power systems: An improved fireworks algorithm. *Int. J. Electr. Power Energy Syst.* **2021**, *126*, 106579. [CrossRef]
41. Ghannadi, P.; Kourehli, S.S. Multiverse optimizer for structural damage detection: Numerical study and experimental validation. *Struct. Des. Tall Spec. Build.* **2020**, *29*, e1777. [CrossRef]
42. Pereira, J.L.J.; Francisco, M.B.; Diniz, C.A.; Oliver, G.A.; Cunha Jr, S.S.; Gomes, G.F. Lichtenberg algorithm: A novel hybrid physics-based meta-heuristic for global optimization. *Expert Syst. Appl.* **2021**, *170*, 114522. [CrossRef]
43. Moosavi, S.H.S.; Bardsiri, V.K. Poor and rich optimization algorithm: A new human-based and multi populations algorithm. *Eng. Appl. Artif. Intell.* **2019**, *86*, 165–181. [CrossRef]
44. Azizi, M. Atomic orbital search: A novel metaheuristic algorithm. *Appl. Math. Model.* **2021**, *93*, 657–683. [CrossRef]
45. Abualigah, L.; Diabat, A. Advances in sine cosine algorithm: A comprehensive survey. *Artif. Intell. Rev.* **2021**, *54*, 2567–2608. [CrossRef]
46. Mousavirad, S.J.; Ebrahimpour-Komleh, H. Human mental search: A new population-based metaheuristic optimization algorithm. *Appl. Intell.* **2017**, *47*, 850–887. [CrossRef]
47. Ghazal, T.M. Performances of k-means clustering algorithm with different distance metrics. *Intell. Autom. Soft Comput.* **2021**, *30*, 735–742. [CrossRef]
48. Zou, M.; Zeng, Q.; Zhang, X. Weakly-supervised Action Learning in Procedural Task Videos via Process Knowledge Decomposition. *IEEE Trans. Circuits Syst. Video Technol.* **2024**, *34*, 5575–5588. [CrossRef]
49. Benabdellah, A.C.; Benghabrit, A.; Bouhaddou, I. A survey of clustering algorithms for an industrial context. *Procedia Comput. Sci.* **2019**, *148*, 291–302. [CrossRef]
50. 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]
51. Singh, S.; Srivastava, S. Review of clustering techniques in control system: Review of clustering techniques in control system. *Procedia Comput. Sci.* **2020**, *173*, 272–280. [CrossRef]
52. Dafir, Z.; Lamari, Y.; Slaoui, S.C. A survey on parallel clustering algorithms for big data. *Artif. Intell. Rev.* **2021**, *54*, 2411–2443. [CrossRef]
53. Fahad, A.; Alshatri, N.; Tari, Z.; Alamri, A.; Khalil, I.; Zomaya, A.Y.; Foufou, S.; Bouras, A. A survey of clustering algorithms for big data: Taxonomy and empirical analysis. *IEEE Trans. Emerg. Top. Comput.* **2014**, *2*, 267–279. [CrossRef]
54. Tian, K.; Zhou, S.; Guan, J. Deepcluster: A general clustering framework based on deep learning. In *Machine Learning and Knowledge Discovery in Databases: European Conference (ECML PKDD 2017)*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 809–825.
55. Lloyd, S. Least squares quantization in PCM. *IEEE Trans. Inf. Theory* **1982**, *28*, 129–137. [CrossRef]
56. Ahmed, M.; Seraj, R.; Islam, S.M.S. The k-means algorithm: A comprehensive survey and performance evaluation. *Electronics* **2020**, *9*, 1295. [CrossRef]
57. Arora, P.; Varshney, S.; et al. Analysis of k-means and k-medoids algorithm for big data. *Procedia Comput. Sci.* **2016**, *78*, 507–512. [CrossRef]
58. Daoudi, S.; Anouar Zouaoui, C.M.; El-Mezouar, M.C.; Taleb, N. Parallelization of the K-Means++ Clustering Algorithm. *Ing. Syst. d'Inf.* **2021**, *26*, 59–66. [CrossRef]
59. 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.* **2023**, *622*, 178–210. [CrossRef]
60. Miraftabzadeh, S.M.; Colombo, C.G.; Longo, M.; Foiadelli, F. K-means and alternative clustering methods in modern power systems. *IEEE Access* **2023**, *11*, 119596–119633. [CrossRef]
61. Nie, F.; Li, Z.; Wang, R.; Li, X. An effective and efficient algorithm for K-means clustering with new formulation. *IEEE Trans. Knowl. Data Eng.* **2022**, *35*, 3433–3443. [CrossRef]
62. Wan, B.; Huang, W.; Pierre, B.; Cheng, Y.; Zhou, S. K-Means algorithm based on multi-feature-induced order. *Granul. Comput.* **2024**, *9*, 45. [CrossRef]
63. Yang, X.; Zhao, W.; Xu, Y.; Wang, C.D.; Li, B.; Nie, F. Sparse K-means clustering algorithm with anchor graph regularization. *Inf. Sci.* **2024**, *667*, 120504. [CrossRef]
64. Ezugwu, A.E.; Ikotun, A.M.; Oyelade, O.O.; Abualigah, L.; Agushaka, J.O.; Eke, C.I.; Akinyelu, A.A. A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects. *Eng. Appl. Artif. Intell.* **2022**, *110*, 104743. [CrossRef]
65. Li, Y.; Tang, B.; Jiao, S.; Su, Q. Snake optimization-based variable-step multiscale single threshold slope entropy for complexity analysis of signals. *IEEE Trans. Instrum. Meas.* **2023**, *72*, 6505313. [CrossRef]

66. Zheng, W.; Pang, S.; Liu, N.; Chai, Q.; Xu, L. A compact snake optimization algorithm in the application of WKNN fingerprint localization. *Sensors* **2023**, *23*, 6282. [CrossRef]
67. Yao, L.; Yuan, P.; Tsai, C.Y.; Zhang, T.; Lu, Y.; Ding, S. ESO: An enhanced snake optimizer for real-world engineering problems. *Expert Syst. Appl.* **2023**, *230*, 120594. [CrossRef]
68. Yan, C.; Razmjooy, N. Optimal lung cancer detection based on CNN optimized and improved Snake optimization algorithm. *Biomed. Signal Process. Control* **2023**, *86*, 105319. [CrossRef]
69. Gad, A.G. Particle swarm optimization algorithm and its applications: A systematic review. *Arch. Comput. Methods Eng.* **2022**, *29*, 2531–2561. [CrossRef]
70. Ahmadyfard, A.; Modares, H. Combining PSO and k-means to enhance data clustering. In Proceedings of the 2008 International Symposium on Telecommunications, Tehran, Iran, 27–28 August 2008; pp. 688–691.
71. Liu, N.; Liu, S.; Chai, Q.W.; Zheng, W.M. A method for analyzing Stackelberg attack–defense game model in 5G by tCPSO. *Expert Syst. Appl.* **2023**, *228*, 120386. [CrossRef]
72. Kapil, S.; Chawla, M.; Ansari, M.D. On K-means data clustering algorithm with genetic algorithm. In Proceedings of the 2016 Fourth International Conference on Parallel, Distributed and Grid Computing (PDGC), Waknaghat, India, 22–24 December 2016; pp. 202–206.
73. Reddy, T.N.; Supreethi, K. Optimization of K-means algorithm: Ant colony optimization. In Proceedings of the 2017 International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 18–19 July 2017; pp. 530–535.
74. Baziar, A.; Kavousi-Fard, A. Considering uncertainty in the optimal energy management of renewable micro-grids including storage devices. *Renew. Energy* **2013**, *59*, 158–166. [CrossRef]
75. Hosseinnezhad, V.; Babaei, E. Economic load dispatch using θ-PSO. *Int. J. Electr. Power Energy Syst.* **2013**, *49*, 160–169. [CrossRef]
76. Xu, R.; Wunsch, D. Survey of clustering algorithms. *IEEE Trans. Neural Netw.* **2005**, *16*, 645–678. [CrossRef]
77. Hatamlou, A. Black hole: A new heuristic optimization approach for data clustering. *Inf. Sci.* **2013**, *222*, 175–184. [CrossRef]
78. Liang, J.J.; Qu, B.; Suganthan, P.N.; Hernández-Díaz, A.G. *Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session on Real-Parameter Optimization*; Technical Report 201212; Computational Intelligence Laboratory, Zhengzhou University: Zhengzhou, China; Nanyang Technological University: Singapore, 2013; pp. 281–295.
79. Ni, K. A Clustering Algorithm Combining Fuzzy C-Means and Artificial Bee Colony Algorithm. *Int. J. Innov. Comput. Inf. Control* **2024**, *20*, 297–311.
80. Cui, R. An Improved Fuzzy C-Means Clustering Algorithm Considering Data Density Distribution. In Proceedings of the 2024 IEEE 2nd International Conference on Control, Electronics and Computer Technology (ICCECT), Jilin, China, 26–28 April 2024; pp. 1332–1336.