applied sciences

MDPI

Review

Initialisation Approaches for Population-Based Metaheuristic Algorithms: A Comprehensive Review

Jeffrey O. Agushaka and Absalom E. Ezugwu *

School of Mathematics, Statistics and Computer Science, University of KwaZulu-Natal, King Edward Road, Pietermaritzburg 3201, KwaZulu-Natal, South Africa; 218088307@stu.ukzn.ac.za
* Correspondence: EzugwuA@ukzn.ac.za

Abstract: A situation where the set of initial solutions lies near the position of the true optimality (most favourable or desirable solution) by chance can increase the probability of finding the true optimality and significantly reduce the search efforts. In optimisation problems, the location of the global optimum solution is unknown a priori, and initialisation is a stochastic process. In addition, the population size is equally important; if there are problems with high dimensions, a small population size may lie sparsely in unpromising regions, and may return suboptimal solutions with bias. In addition, the different distributions used as position vectors for the initial population may have different sampling emphasis; hence, different degrees of diversity. The initialisation control parameters of population-based metaheuristic algorithms play a significant role in improving the performance of the algorithms. Researchers have identified this significance, and they have put much effort into finding various distribution schemes that will enhance the diversity of the initial populations of the algorithms, and obtain the correct balance of the population size and number of iterations which will guarantee optimal solutions for a given problem set. Despite the affirmation of the role initialisation plays, to our knowledge few studies or surveys have been conducted on this subject area. Therefore, this paper presents a comprehensive survey of different initialisation schemes to improve the quality of solutions obtained by most metaheuristic optimisers for a given problem set. Popular schemes used to improve the diversity of the population can be categorised into random numbers, quasirandom sequences, chaos theory, probability distributions, hybrids of other heuristic or metaheuristic algorithms, Lévy, and others. We discuss the different levels of success of these schemes and identify their limitations. Similarly, we identify gaps and present useful insights for future research directions. Finally, we present a comparison of the effect of population size, the maximum number of iterations, and ten (10) different initialisation methods on the performance of three (3) population-based metaheuristic optimizers: bat algorithm (BA), Grey Wolf Optimizer (GWO), and butterfly optimization algorithm (BOA).

Keywords: initialisation; metaheuristics; metaheuristics optimisers; population size

Citation: Agushaka, J.O.; Ezugwu, A.E. Initialisation Approaches for Population-Based Metaheuristic Algorithms: A Comprehensive Review. *Appl. Sci.* **2022**, *12*, 896. https://doi.org/10.3390/app12020896

Academic Editor: Mario Dante Lucio Giacobini

Received: 10 November 2021
Accepted: 10 January 2022
Published: 17 January 2022

1. Introduction

The primary concern of optimisation is finding either the minima or maxima of the objective function, subject to some given constraints. Optimisation problems naturally occur in machine learning, artificial intelligence, computer science, and operations research. Optimisation has been used to improve processes in all human endeavours. A wide variety of techniques for optimisation exist. These techniques include linear programming, quadratic programming, convex optimization, interior-point method, trust-region method, conjugate-gradient methods, evolutionary algorithms, heuristics, and metaheuristics [1]. The era of artificial intelligence ushered in techniques for optimisation that are capable of finding near-optimal solutions to challenging and complex real-world optimisation problems. Then came the nature-inspired and bio-inspired metaheuristic optimization era, with huge successes recorded and increasing popularity over the past four decades.

Many attributed the popularity of nature-inspired and bio-inspired metaheuristics optimization algorithms to their ability to find near-optimal solutions [2]. This success can be attributed to how these nature-inspired and bio-inspired metaheuristics optimizers mimic natural phenomena [3]. These natural phenomena have inspired the development of almost all of the metaheuristic algorithms. Evolutionary techniques are gaining popularity in the same vein, too, with many novel techniques developed regularly. The performance of evolutionary techniques matches the nature-inspired or bio-inspired algorithms [4].

The successes of these metaheuristic optimizers on real-world problems come with tremendous challenges. These challenges arise from the fact that real-world optimisation problems are complex and have multiple nonlinear constraints. The ability of optimisers to navigate these challenges and achieve optimality depends heavily on how the initial population is distributed, especially for gradient-based optimizers [5]. Though metaheuristic optimizers are gradient-free, they must also be initialised; thus, they are greatly influenced by the nature of the initial population, especially the large-scale multimodal problems. Population-based metaheuristic algorithms show varying abilities to reach a global optimum when the initialisation scheme is varied [6].

Interestingly, in the last decade, there has been an exponential growth in the number of proposed nature-inspired optimisation algorithms. Furthermore, there has been a corresponding claim of novelty and solid capability of the algorithms serving as powerful optimisation tools. Unfortunately, most algorithms do not seem to draw inspiration from nature or incorporate any successful methodology that mimics natural phenomena or systems [7]. From the concept of the theorem of No Free Lunch, many real-world optimisation problems still require new approaches or methods to be solved perfectly or optimally. The theorem proved that any method could solve a problem efficiently, but no single method can effectively solve all problems. Studies have shown the popularity and successes of proposing algorithms that mimic the behaviours of animals to solve optimisation problems with reasonable accuracy.

The commonly used distribution for initialisation by most metaheuristic algorithms is the random number generator, which generates sequences (used as position vectors) that follow the uniform probability distribution. However, these sequences do not have low discrepancies, or are not equidistributed in a given search area, and do not efficiently cover the search space [8]. On the other hand, quasirandom numbers can be generated with low discrepancy; these have been proven to have optimal discrepancy because they tend to cover the search space better and are helpful in optimisation [9]. Low discrepancy sequences, like Van der Corput, Sobol, Faure, and Halton, are potent computational method tools and have been used to improve the performance of optimisation algorithms. Many other approaches exist in the literature, and these are presented in this paper.

A situation where the set of initial solutions lies near the position of the true optimality by chance, can increase the probability of finding the true optimality and significantly reduce the search efforts. In optimisation problems, the location of the true optimality is unknown a priori, and initialisation is a stochastic process. Additionally, the population size is equally important and when considering problems with high dimensions, small population size may lie sparsely in unpromising regions, and can return suboptimal solutions with bias. In addition, the different distributions used as position vectors for the initial population may have a different sampling emphasis and hence different degrees of diversity.

To demonstrate the importance of initialisation, consider the Bukin N. 6 function shown in Figure 1. We assumed a search space of $[-20, 0] \times [-8, 4]$. The advanced arithmetic optimization algorithm (nAOA) [10] was initialised with beta distribution, and the distribution of the population after the first iteration is shown in Figure 2. The blue dots represent the current location of the population, the red asterisk (*) represents the current best solution, and the red star (★) denotes the global optimal solution of the Bukin function. The nAOA converged towards the optimal solution after a few iterations, as shown in Figure 3. Similarly, the nAOA was initialised with the random number, and the distribution

of the population after the first iteration is shown in Figure 4. The distribution of the population of nAOA quickly falls into a local optimum after a few iterations, as shown in Figure 5.

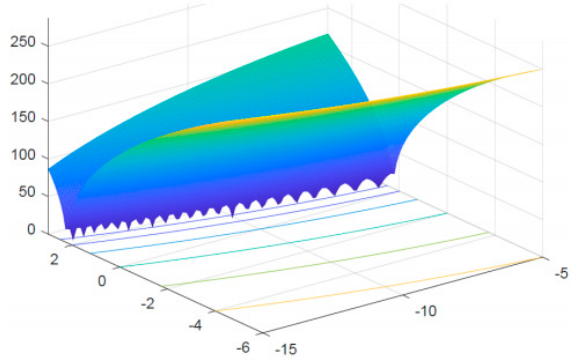

Figure 1. Bukin N. 6 landscape.

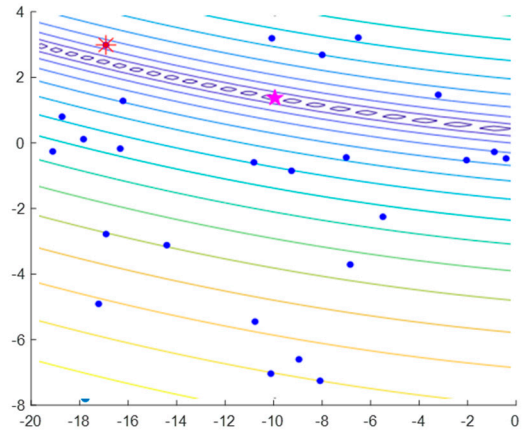

Figure 2. The beta distribution of the population after the first iteration.

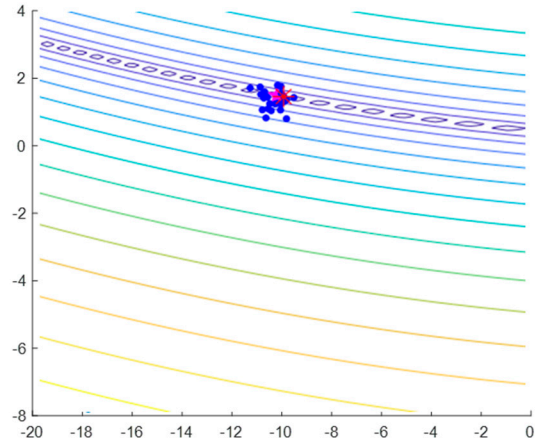

Figure 3. The beta distribution of the population after a few iterations.

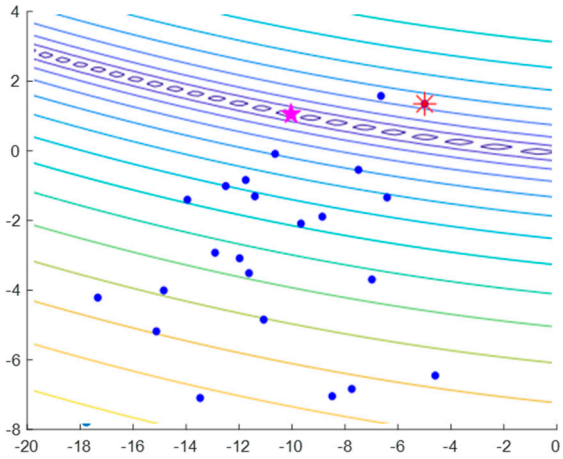

Figure 4. The random number distribution of the population after the first iteration.

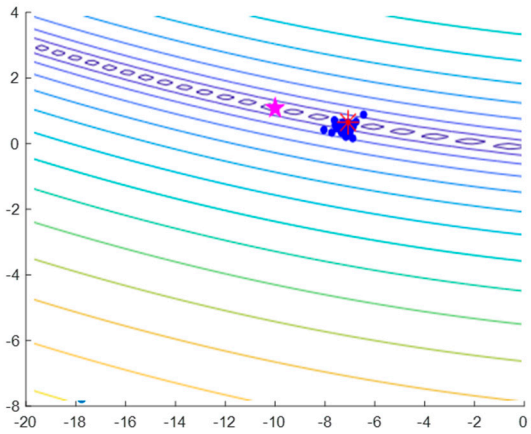

Figure 5. The random number distribution of the population after a few iterations.

Although initialisation plays a significant role in the performance of most metaheuristic optimizers, few studies or surveys have been conducted on the subject area. A search using the keywords survey OR review, initialisation (initialization), and metaheuristics, yielded no comprehensive review or survey articles in the literature. However, in discussing PSO variants, ref. [11] provide a paragraph on attempts to improve PSO performance using different initialisation schemes. The authors discuss how low discrepancy sequences and variants of opposition-based learning enhance the initial swarm population. Another attempt using GA was presented by [12], where the effect of three initialisation functions, namely, nearest neighbour (NN), insertion (In), and Solomon's heuristic, were studied. Li, Liu, and Yang [13] evaluated the effect of 22 different probability distribution initialisation methods on the convergence and accuracy of five optimisation algorithms. In this regard, we formulate the research question given below to accomplish our work:

What literature modified the initialisation control parameters comprising size and diversity of population and the maximum number of iterations to improve the algorithms' performance?

The following questions are formulated to answer the main research question:

i. What research exists that used distributions other than the random number for initialisation of the population to improve the performance of metaheuristic algorithms?
ii. What study exists that fine-tuned the population size and the number of iterations of different algorithms?
iii. What are the major initialisation distributions used by the population-based algorithm?

iv. What problems were solved by the modified algorithms?
v. What are other challenges yet to be explored by researchers in the research area?

To the best of our knowledge, no survey or review article focuses on general efforts to improve the performances of different metaheuristic optimizers using different initialisation schemes in the literature, which motivates the current research contribution. Therefore, this study presents a comprehensive survey of different initialisation methods employed by metaheuristic algorithm designers and optimisation enthusiasts to improve the performance of the different metaheuristic optimizers available in the literature. The study covers articles published between 2000–2021, and the specific contributions of this paper are summarised as follows:

i. We present a comprehensive review of the different distributions used to improve the diversity of the initial population of population-based metaheuristic algorithms.
ii. We categorise the schemes into random numbers, quasirandom sequences, chaos theory, probability distributions, hybrids of other heuristic or metaheuristic algorithms, Lévy, and others.
iii. We also discuss the different levels of success of these schemes and identify their limitations.
iv. An in-depth highlight of the glossary of efforts to improve the performance of metaheuristic algorithms using several initialisation schemes is presented. Metaheuristic research enthusiasts can easily reference this glossary.
v. Finally, we provide the research gaps, useful insights, and future directions.

The rest of the paper is organised as follows. In Section 2, we provide the methodology used for collecting papers. The major initialisation methods used to improve the performance of the algorithms are presented in Section 3. In Section 4, we discuss the various application areas of the present study. Results and discussion of findings from our experiment are presented in Section 5. Finally, Section 6 presents the concluding remarks.

2. Methodology and Paper Collection Technique

This section discussed the procedure used for paper selection, collection, and review. Search keywords, search techniques, data sources, databases, plus inclusion and exclusion criteria are explained. We followed the systematic literature review procedure provided in the work of [14], and we were guided by the work of [15].

2.1. Keywords

In order to retrieve relevant articles to achieve our review goal, we carefully selected some useful keywords, that we used to search the database: initialization (initialisation), metaheuristic, optimization (optimisation), OR algorithm. The initial search for these articles was carried out between 20 to 24 September 2020, and the final search was carried out between 25 to 30 October 2021. Articles retrieved based on the keywords searched were perused during each search in order to collect more related articles from their citations and references sections.

2.2. Academic Databases

The keywords selected were used to search and retrieve relevant works from the body of literature. We targeted only articles that are published in reputable peer-review journals, edited books, and conference proceedings indexed in two (2) academic databases. The Web of Science (WoS) and Scopus repositories are the academic databases that we used to extract articles. These are repositories with high-quality articles that are published in SCI-indexed journals and ranked international conferences. We performed a search based on the above keywords in these repositories up to 2021.

2.3. Inclusion/Exclusion Criteria

We formulated some inclusion and exclusion criteria in order to collect solely relevant literature examples. The collected articles are either included or excluded based on some

criteria after perusing their titles, abstracts, conclusions and, in some cases, the complete content. The selected criteria are given in Table 1.

Table 1. Inclusion/Exclusion Criteria.

Inclusion	Exclusion
Articles that used different initialisation schemes to improve the performance of metaheuristic algorithm	While we discussed the commonly used pseudo-random number initialization scheme, we excluded algorithms that used the scheme. Including these articles would mean reviewing the entire metaheuristic algorithms, which is outside the scope of this work
Articles published in reputable peer-review journals, conference proceedings, and edited books	Articles published as part of textbooks, abstracts, editorials, and keynote speeches
Articles that are written in the English language	Articles that are written in other languages besides English

2.4. Eligibility

We applied the inclusion and exclusion criteria to determine the eligibility of the selected articles. A total of 99 articles were returned by WoS and 58 articles were returned by Scopus repositories, respectively. Figure 6 shows the document type of distribution from the WoS repository, where 83 articles, 16 conference proceedings, five (5) early access, and one book chapter have been published. Similarly, Figure 7 shows the distribution of document types from Scopus, with 39 articles, 17 conference papers, one book, and one conference review, respectively. Both figures show that more articles are published in journals than in conferences and book chapters.

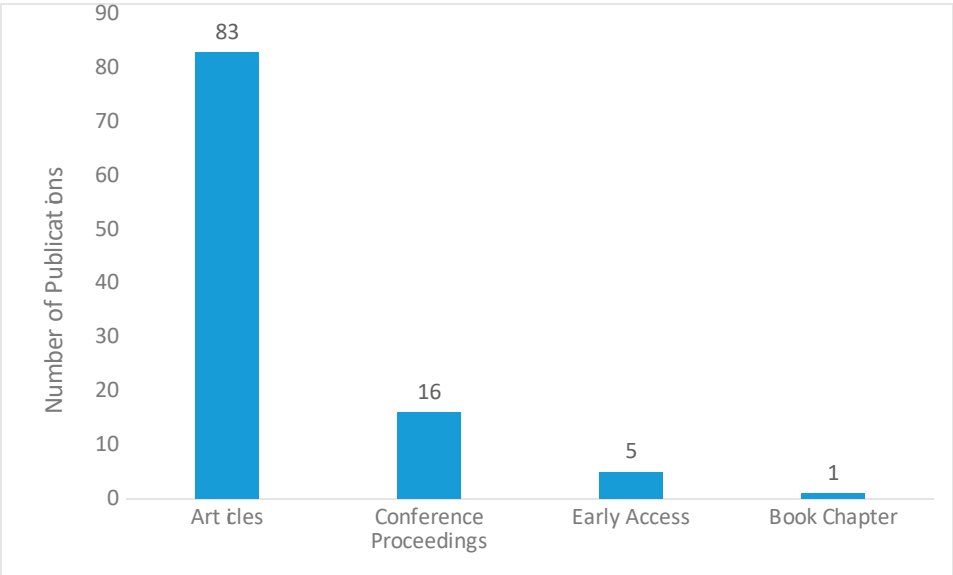

Figure 6. Document Type Distribution from WoS.

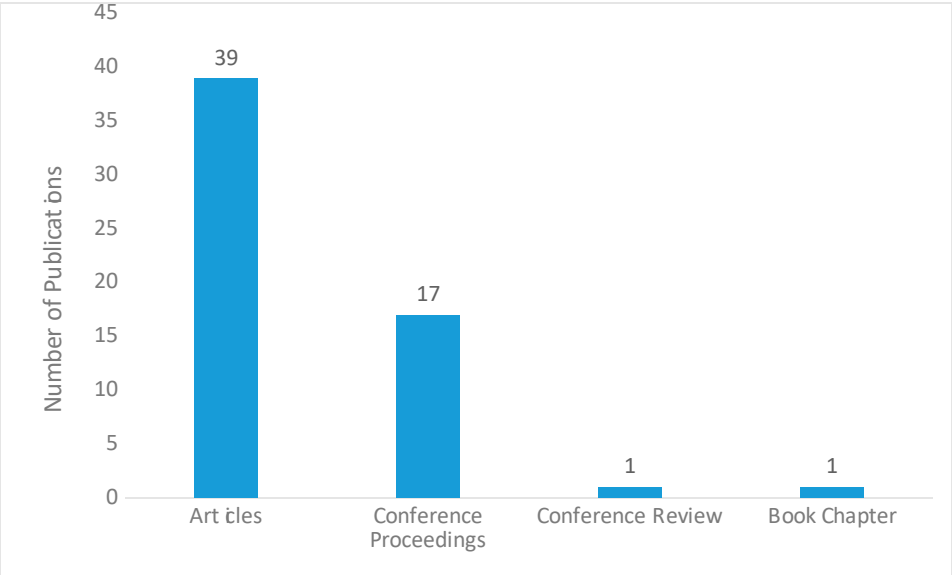

Figure 7. Document Type Distribution from Scopus.

After cross-referencing the two repositories, we found many papers that intersect both, and we excluded these articles from the other repository. In addition, we found articles that were included in the search because they contained the keyword "initialization", but they did not relate to our research; hence, we also excluded them. A total of 52 articles were selected for this survey, after applying the inclusion criteria.

3. Major Initialisation Methods

This section discusses the updated efforts on improving the initial condition of the population of metaheuristic algorithms. This provided an answer to our research question, what research examples exist that used distributions other than the random number for initialisation of the population, to improve the performance of metaheuristic algorithms? The different initialisation schemes identified in the literature were summarised or categorised into pseudo-random number or Monte Carlo methods, quasirandom methods, probability distributions, hybrid, chaos theory, Lévy, and ad hoc knowledge of the domain, and others. The categorisation was performed to aid our discussion of the schemes that we identified.

3.1. Pseudo-Random Number or Monte Carlo Methods

By default, the random number generation or Monte Carlo method is the most used initialisation scheme for most metaheuristic algorithms. It uses the uniform probability distribution to generate uniform pseudo-random number sequences that are used as location vectors for the population. Many population-based metaheuristic algorithms use this scheme, and interested readers can refer to the respective optimisers for details. The role of the random number generation, as an essential part of the initialisation process, has been greatly emphasised [16,17]. Despite its popularity, the random number sequence suffers because its discrepancy is not low and does not efficiently cover the search space [8]. The discrepancy of the random number greatly influences how genuinely random the resulting randomly generated solutions are within the solution search spaces [18]. Research works, such as those by [19,20], have shown that the random number does not result in an optimal discrepancy that will aid the convergence of the algorithms. Figure 8 shows how the random numbers tend to form clusters after several iterations, instead of filling up the search space. This is a significant disadvantage of using random number generators to initialise the population of the metaheuristic algorithms.

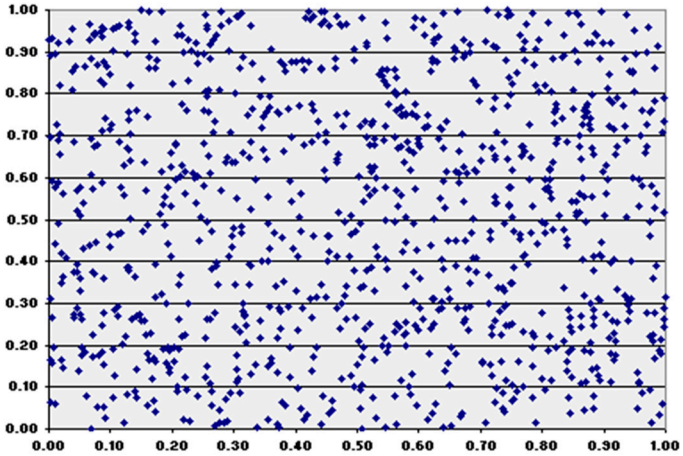

Figure 8. Random numbers after 1000 iterations.

We did not include a table for this category because most existing metaheuristic algorithms belong here. A table for this would be huge, and there is no area of application that this scheme has not been applied to.

3.2. Quasirandom Methods

Quasirandom number generators are known to generate sequences that are proven to have low discrepancy [9]. Low discrepancy sequences, like Van der Corput, Sobol, Faure, and Halton, are potent computational method tools, which have been used to improve the performance of optimisation algorithms. Quasirandom numbers are effective initialisation mechanisms for metaheuristic algorithms to uniformly cover the search space in order to obtain the optimal solution. The particle swarm population in the work of [21] was initialised using the randomized low discrepancy sequences of Halton, Sobol, and Faure. The three modified PSO were applied to the benchmark test functions, and results were then compared with the global best PSO. This showed that PSO was significantly improved with Sobol, while the results showed a varying improvement with Faure and Halton. Similarly, the Van der Corput and Sobol sequences were used to initialise the PSO and were then applied to solve the benchmark functions [8]. The results obtained were promising when compared to the original PSO.

The krill population in the KH algorithm was initialised using the Faure, Sobol, and Var der Corptut sequences [22]. The benchmark test functions were used to test the efficacy of the modified KH. Our findings revealed significant improvements in the performance of the KH algorithm when initialised using Faure, Sobol, and Var der Corptut low-discrepancy sequences, which was also the case with the guaranteed convergence particle swarm optimization (GCPSO) algorithm, using the Niching methods to initialise the swarm population [23]; the Niching methods are based on the Faure low-discrepancy sequence, and the benchmark test functions were used to evaluate the performance of GCPSO, with promising results.

The initialisation schemes that were implemented using low-discrepancy sequences are known to perform poorly, as the problem dimension or graph size scales up. Figure 9 shows how the Halton sequence spreads and fills the search space at the 1000th iteration, improving the convergence of algorithms. We have noted authors [24] who use the Halton sequence to initialise the search agents of the Wingsuit Flying Search (WFS) algorithm.

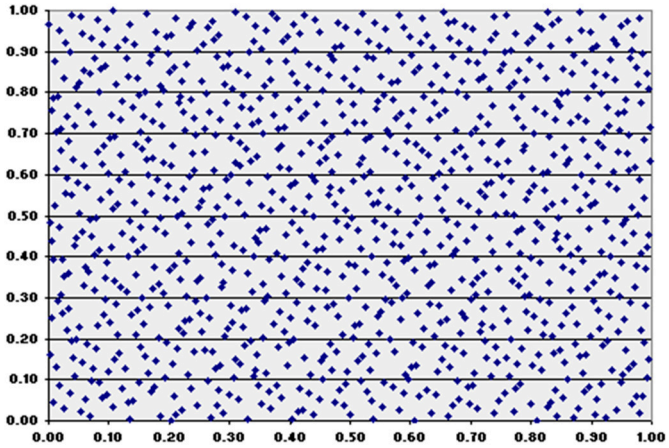

Figure 9. Halton sequence after 1000th iteration.

Table 2 summarises the glossary of efforts that used low discrepancy sequences (quasirandom numbers) to initialise the population of some metaheuristic optimizers. Interested readers can refer to the references for more details about the efforts. In all the papers reviewed in this section, the authors claimed that fine-tuning the initialisation control parameters (population size and diversity and maximum number of iterations or function evaluations) improved the performance of the algorithm.

Table 2. Summary of quasirandom methods.

Reference	Year	Initialisation Scheme	Optimisation Problem
[21]	2007	Halton, Sobol, and Faure	Benchmark test functions
[8]	2008	Van der Corput and Sobol sequences	Benchmark test functions
[25]	2018	A quasirandom sequence Torus	Benchmark test function
[22]	2020	Van der Corput Faure and Sobol sequences	Benchmark test function
[26]	2005	Halton low-discrepancy sequence	Benchmark function
[27]	2005	Sobol and Halton sequences	Benchmark function
[23]	2002	Faure sequences	Benchmark functions
[28]	2009	Sobol sequence	Benchmark functions
[29]	2021	Halton, Sobol, and Torus	Benchmark functions
[30]	2012	Sobol sequences	Image Segmentation
[31]	2017	Latin hypercube sampling (LHS)	Optimisation of structural components under Fatigue

3.3. Probability Distributions

The probability distribution describes the possible values and likelihood that a random number is effective within a defined interval. Different probability distributions and their rigorous statistical properties can be used to initialise the population of metaheuristic algorithms. Li, Liu, and Yang [13] used variants of Beta distribution, uniform distribution, normal distribution, logarithmic normal distribution, exponential distribution, Rayleigh distribution, Weibull distribution, and Latin hypercube sampling [31] to form 22 different initialisation schemes in order to evaluate PSO, CS, DE, ABC, and GA. The variants of the probability distributions are as follows:

- Beta distribution

The Beta distribution is a continuous probability distribution over the interval (0,1). It can be written as $X \sim Be(a, b)$. Varying the values of a and b resulted in a variant of

the Beta distribution, generating sequences with different behaviours in the search space. Three variants of the Beta distribution were used,

- Uniform distribution

A uniform distribution is defined over the interval [a, b], and it is usually written as $X \sim U(a, b)$. One variant of the normal distribution was used.

- Normal distribution

The Gaussian Normal distribution is usually written as $X \sim N(\mu, \sigma^2)$. In addition, varying the values of μ and σ^2 resulted in three (3) variants of the normal distribution, which generates sequences with different behaviours in the search space.

- Logarithmic normal distribution

The logarithmic normal distribution is often written as $lnX \sim N(\mu, \sigma^2)$. Four (4) variants of the logarithmic normal distribution were created by varying the values of μ and σ^2.

- Exponential distribution

An exponential distribution is asymmetric with a long tail and can be written as $X \sim \exp(\lambda)$. Varying λ, resulted in three variants of the distribution which were used to initialise the population of the five algorithms.

- Rayleigh distribution

The Rayleigh distribution can be written as $X \sim Rayleigh(\sigma)$. Three (3) variants of the distribution were created by varying the value of σ.

- Weibull distribution

This distribution can be considered as a generalisation of a few other distributions. It can be written as $X \sim Weibull(\lambda, k)$. For example, $k = 1$ corresponds to an exponential distribution, while $k = 2$ leads to the Rayleigh distribution. In the same vein, three variants of the distribution were created.

The convergence and accuracy of five metaheuristic optimizers were evaluated on the benchmark test functions and the CEC2020 test functions. These optimisers are then initialised using 22 different initialization schemes [13]. The findings of those authors showed that PSO and CS are more sensitive to the initialisation scheme used, whereas DE was less susceptible to the initialisation scheme used. In addition, PSO relies on a greater population size, whereas CS requires a lesser population size. DE does well with an increased number of iterations. The Beta, Rayleigh, and exponential distributions are great performers as the results showed that they greatly influence the convergence of the optimisers used.

Georgioudakis, Lagaros, and Papadrakakis [31] incorporated Latin hypercube sampling (LHS) to initialise four (4) optimisers; namely, the evolution strategies (ES), covariance matrix adaptation (CMA), elitist covariance matrix adaptation (ECMA) and differential evolution (DE). They use these optimisers to investigate the relation between the geometry of the structural components, and their service life. They aimed to improve the service life of structural components under fatigue. Their choice of LHS instead of the random Monte Carlo simulation optimised the number of samples needed to calculate the problem regarding the formulation of the statistical quantities.

The stochastic fractal search (SFS) technique was used in the work of [32] to improve the performance of the multi-layer perceptron neural network. It was used to obtain the optimal set of weights and threshold parameters. The hybrid approach was tested on EEE 14- and 118-bus systems, and the results were compared with other non-optimized MLP (optimized MLP based on genetic algorithm (MLP-GA) and Particle Swarm Optimization (MLP-PSO)). The precision was up by 20–50%, and the computational time was down by 30–50%. However, SFS tends to ignore local search; the correct balance between the global and local search is desired. Similarly, the levy-flight was replaced by stochastic random

sampling of simpler fat-tailed distributions enhanced with scaled-chaotic sequences to boost cuckoo search (CS) performance in solving the complex wellbore trajectories problem [33].

Probability distributions generally suffer from issues such as equiprobable disjunct intervals and errors in correlations between variables. We summarise efforts in this category in Table 3.

Table 3. Summary of PDF category.

Reference	Year	Initialisation Scheme	Optimisation Problem
[32]	2017	Stochastic fractal search technique	Dynamic state estimation (DSE) problem at the filtering stage
[13]	2020	Variants of Beta distribution, uniform distribution, normal distribution, logarithmic normal distribution, exponential distribution, Rayleigh distribution, Weibull distribution, and Latin hypercube Ssmpling	Benchmark function
[31]	2017	Latin hypercube sampling	Structural components under fatigue
[34]	2011	Stochastic demands	Vehicle routing problem
[35]	2012	Smart sampling	Benchmark functions
[36]	2018	Log logistic	Training of artificial neural network
[37]	2016	Neural fuzzy inference	Flood susceptibility modeling
[38]	2019	Adaptive neuro-fuzzy inference	Groundwater potential mapping
[33]	2016	Stochastic random a sampling of simpler fat-tailed distributions enhanced with scaled-chaotic sequences	Complex wellbore trajectories

3.4. Hybrid with Other Metaheuristic Algorithms

Most researchers used another metaheuristic algorithm to find an optimal solution for the initial position of the population in this approach. Metaheuristic algorithms with a high convergence rate in a specific problem domain are often used to find an initial solution. These solutions are then fed into the other metaheuristic algorithms as the initial conditions. A hybridization of ABC and TS was proposed in the work of [39], where the bee population was initialised using the randomized breadth-first search. The performance of their hybrid was better than the algorithms they compared it with; however, it suffers from the time complexity problem of BFS. The authors [40] initialised the monarch butterfly algorithm by equally partitioning the search space and in the F and T random distribution to mutate the divided population. The results showed significant improvements. The Krill in the work of [41] were initialised using the pairwise linear optimisation, which uses fuzzy rules to create clusters that are used as the initial point for the KH. However, the results showed that this improvement would only suit systems based on fuzzy approximators. The CRO was improved using the VNS algorithm with a new processor selection model for the initialisation. The results are promising; however, parameter sensitivity still needs to be resolved [42].

The cuckoo population was initialised using quasi-opposition-based learning (QOBL) [43]. Reaching the optimal search is enhanced by considering a guess and its quasi-opposite guess. The initialisation schemes of BA are improved using a quasirandom sequence with low discrepancy called Torus [25]. Their results were good; however, the results were not evaluated for higher-dimensional problems. Four (4) different dispatching rules (DR)-based initialisation strategies were used by [44], with varying advantages and disadvantages. The best result was obtained when all of the strategies were used together, which means that the diversity of the population contributed less to the algorithm's overall performance. In [45], a scheme inspired by SAM was developed, and it is a simplified heuristic model that begins the swarm search with an initial set of high-quality solutions.

ABC was used to find the optimal cluster centre of the FCM [46]. An improved ABC was also proposed to solve the vehicle routing problem (VRP) [47]. Among other improvements, the bees were initialised using push forward insertion. An improved DE, named the enhanced differential evolution algorithm (EDE), used the opposition-based learning for the initialisation, along with other improvements, in order to enhance the performance of DE [48]. The optimised stream clustering algorithm (OpStream) used an optimal solution of a metaheuristic algorithm to initialise the first set of the cluster [49]. The optimal solution of the optimal shortening of covering arrays (OSCAR) problem was used as the initialisation function of a metaheuristic algorithm [50].

Mandal, Chatterjee, and Maitra [51] used the PSO to solve the problem that hampered the Chan and Vese algorithm for image segmentation problems, which is low-performance if the contours are not well initialised; the contours are initialised simultaneously with the population. Their hybrid solution made contour initialisation irrelevant to the performance of the algorithm. Another effort was presented by [52], where a scheme to initialise the fuzzy c-means (FCM) clustering algorithm using the PSO was proposed. Finding the optimal cluster centres was set as the objective function of the PSO.

A memetic algorithm that uses the greedy randomized adaptive search procedure (GRASP) metaheuristic and path relinking to initialise and mutate the population was proposed [53]. However, the scalability of the MA was untested. The authors [54] proposed an initialisation scheme that used both the Metropolis-Hastings (MH) and function domain contraction technique (FDCT). MH is helpful when generating the direct sequence of a PD that is difficult. However, MH is best for high multidimensional complex optimisations, as these are problem-dependent. In such a situation, the FDCT is then employed. The FDCT is a sequential three-step solution starting with a random solution generator; and if this is not feasible, then the GBEST PSO generator is applied. If the previous two fail, then the search space reduction technique (SSRT) is applied. These steps ensure that the initialised population leads to a better solution.

Competitive swarm optimizer (CSO) is a variant of PSO used by [55] to improve the extreme learning machines (ELM) network by depending on the individual competition of the particles, which optimise its weights and structure. Although the results show great promise, it took more training time to generate effective models. Sawant, Prabukumar, and Samiappan [56] evaluated an approach to initialise the cuckoo nest based on the correlation between the spectral band of the nest that was proposed. The goal is to ensure convergence by making sure the location of the nest does not repeat. The k-means clustering algorithm is used to select specific clusters on the band based on their correlation coefficient. Another approach is presented to resolve the lack of diversity of PSO and its sensitivity to initialisation, which quickly leads to premature convergence. The crown jewel defence (CJD) is used to escape being stocked in the local optima by relocating and reinitialising the global and local best position. However, the performance of this improvement is not tested in higher dimensions [57].

The DE and local search were combined to improve or enhance the chances of an optimal solution to the hybrid flow-shop scheduling problem [58]. The brainstorm optimisation (BSO) was improved in the work of [59] by implementing a scheme that allows for a reinitialisation scheme to be triggered, based on the current population. In the work of [60], those authors used FA to detect the maxima and number of image clusters through a histogram-based segmentation; the maxima are then used to initialise the parameter estimates of the Gaussian mixture model (GMM). In the work [61], the authors proposed a scheme that enhances the initial conditions of an algorithm by considering these initial conditions to be a sub-optimisation problem where the initial conditions are the parameters to be optimised by the MLA. Their obtained results showed improvements compared to the other algorithms used. The FA was also used in the work of [62] as an optimiser to obtain the initial location of the translation parameters for WNNs. This led to a reduction in the number of hidden nodes of WNN and significantly increased the simultaneous generalisation capability of WNNs.

However, time and computational complexity may be a problem for this approach. In addition, a lack of a proven way to hybridise these algorithms greatly depends on the experience of the researcher. A summary of research efforts in this category is given in Table 4.

Table 4. Summary of hybrid methods.

Reference	Year	Initialisation Scheme	Optimisation Problem
[51]	2014	PSO	Image segmentation
[53]	2015	GRASP	Far from most string problem (FFMSP)
[55]	2020	Competitive swarm optimizer (CSO)	Train single hidden layer feed forward Networks (SLFN)
[56]	2019	K-Means clustering algorithm	Band selection of hyperspectral images
[60]	2018	Firefly algorithm	Greyscale image segmentation
[52]	2013	PSO	Mahalanobis distance and post-segmentationCorrection
[46]	2014	ABC	Geo-demographic analysis
[62]	2017	Firefly algorithm	Wavelet neural networks (WNNs)
[49]	2019	Metaheuristic algorithm	Dynamic Data Streams
[63]	2017	Hyper-heuristic	Benchmark test functions
[64]	2020	Hybrid of fuzzy metaheuristics (e.g., FATPSO) and the base TsC algorithms	Different areas
[65]	2015	Heuristic initialization strategies	Dynamic flexible job shop scheduling problems
[66]	2014	Opposition-based learning	Benchmark test function
[42]	2015	VNS	Task Scheduling
[48]	2015	Opposition-based learning	Benchmark function
[43]	2018	Quasi-opposition based learning (QOBL)	Parameter estimation of photovoltaic (PV) models
[57]	2012	Crown jewel defense (CJD)	Benchmark test functions
[58]	2020	DE combined with local search	Hybrid flow-shop scheduling problem
[41]	2017	Pairwise linear optimization	Takagi-Sugeno fuzzy systems
[67]	2018	Greedy algorithm	Multi-skill resource-constrained project scheduling problem
[68]	2020	Nawaz-Enscore-Ham (NEH) algorithm	Slabs and beams manufacturing

3.5. Chaos Theory

Chaos theory describes the unpredictability of systems, and over the years, many advances have been made in this area. Chaotic sequences follow these properties: sensitive to initial conditions, ergodicity, and randomicity. This type of sequence has the advantages of introducing chaos or unpredictability into the optimisation, increasing the range of chaotic motion, and using these chaotic induced variables to search the space effectively [69].

Using the logistic chaotic function, ref. [70] proposed novel improvements on the CS, and one of these improvements is the use of the logistic chaotic function to initialise the population. While their results are promising, they suffer from high computational complexity. The same scheme was used in the work of [71] to improve BA, where the bat population was initialised using chaotic sequences, instead of the random number generator. In addition, the bacterial population of BFO was initialised using chaotic sequences that were generated using logistic mapping [72]. Similarly, the butterflies in the work of [73] were initialised using the homogenous chaotic sequence which were adapted

to the ultraviolet changes. Among other improvements proposed in the work of [74], the chaotic initialisation strategy was used to initialise the whales in the multi-strategy ensemble whale optimization algorithm (MSWOA).

The chaos theory was used to initialise the moth-flame optimization (MFO) [75], firefly algorithm (FA) [76], artificial bee colony (ABC) [77], biogeography based optimization (BBO) [78]), krill herd (KH) [79], water cycle algorithm (WCA) [80], and grey wolf optimizer (GWO) [81]. In all, the authors claimed superiority of their results over other algorithms; however, high computational complexity remains an issue for this category, and we provide a summary of efforts in Table 5.

Table 5. Summary of chaotic based methods.

Reference	Year	Initialisation Scheme	Optimisation Problem
[71]	2014	Chaotic Sequence	Benchmark function
[70]	2017	Logistic Chaotic Function	Enhancement of satellite images
[72]	2018	Chaotic Sequence	Benchmark test functions
[73]	2019	Chaotic Sequence	Synthesis and structure optimization of antenna arrays
[74]	2020	Chaotic Theory	Analog circuits intelligent fault diagnosis
[75]	2017	Chaotic Theory	Medical diagnoses
[76]	2013	Chaotic Theory	Benchmark test function
[77]	2011	Chaotic Theory	Benchmark test function
[78]	2014	Chaotic Theory	Benchmark test function
[79]	2014	Chaotic Theory	Benchmark test function
[80]	2017	Chaotic Theory	Benchmark test function
[82]	2018	Chaotic Theory	Global optimization and feature selection
[83]	2020	Chaotic Theory	Medical diagnosis problems
[84]	2014	Chaotic Theory	Truss structures
[85]	2012	Chaotic Theory	Image matching
[81]	2018	Chaotic Theory	Benchmark test function

3.6. Ad Hoc Knowledge of the Domain

In the ad hoc knowledge of the domain approach, the authors used background knowledge of the domain to design the initialisation scheme of an algorithm. The nature of the problem is what influences the diversity and spread of the initial population. The scheme proposed in the work of [86] used this scheme to generate initial solutions, serving as the initial point for the metaheuristic method. Their results were better and, in some cases, competitive; however, we believe that this method is excessively problem-dependent as such a generalisation is impossible. In the same vein, ref. [87] proposed the initialisation of the bats method, based on ad hoc knowledge of the PV domain. Precisely, they used the peaks with similar duty ratios that occur at the power versus duty ratio of the boost converter curve. Yao et al. [88] used the objective function to minimise the wear and tear of the actuators when initialising the population.

The clans in EHO [89] were initialised by considering the acoustic decay model that is used to obtain the distance between the sensor and the noise source. Depending on the noise level, the intersection of the source coordinates will be at the radii, which is less likely to be single. The clans are initialised, while being based at the centre of the intersection. The technique suffers from being problem-dependent and requires much adaptation before being used in other domains. Finally, a scheme to help PSO avoid reinitialisation to capture the global peaks, when PSO changes its position and value in the P-V curve, was developed

by [90]. Particles are sent to areas of anticipated peaks; once located, particles are sent there to cater for them. Table 6 gives a summary of this approach.

Table 6. Summary of methods based on ad hoc knowledge.

Reference	Year	Initialisation Scheme	Optimisation Problem
[86]	2017	Ad hoc knowledge	Energy market Participation portfolios
[88]	2020	Ad hoc knowledge	Morphing aircraft
[54]	2019	Metropolis–Hastings (MH) and function domain contraction technique (FDCT)	Reservoir drainage plan optimisation problem
[91]	2020	Imaging the search space	Black box setting called COCO
[87]	2020	Ad hoc knowledge	Photovoltaic energy systems under dynamic partial shading
[89]	2020	Ad hoc knowledge	Energy-based acoustic localization
[92]	2020	Ad hoc knowledge	Nonconvex economic dispatch problem
[90]	2020	Ad hoc knowledge	PV Systems

3.7. Lévy Flights

A two-way approach to improving the initialisation scheme for the bees algorithm was also proposed [93]. The patch environment and levy motion imitate the natural food environment and the foraging motion of the bees, respectively. Although the patch concept is used in the original Bees algorithm for the neighbourhood search, its use for initialisation and the levy motion greatly improved its performance. In addition, the performance of the GWO algorithm [94] was enhanced using the Lévy flight (LF) and greedy selection. An improved modified GWO algorithm is proposed to solve global or real-world optimisation problems. In order to boost the efficacy of GWO, strategies are integrated with the modified hunting phases. However, no test was carried out on a specific optimisation domain; hence, no comparison was made. A glossary of efforts on the use of this approach is given in Table 7, and authors claimed superiority of their results over other algorithms.

Table 7. Summary of levy flight methods.

Reference	Year	Initialisation Scheme	Optimisation Problem
[94]	2017	Lévy flight (LF) and greedy selection	Benchmark test function
[95]	2012	Chaotic Lévy Motion	Nonlinear dynamic biological systems
[96]	2016	Lévy Motion	Steel space frames
[97]	2018	Lévy Motion	Neural network training
[98]	2020	Lévy Motion	Benchmark
[99]	2018	Lévy Motion	Engineering design problems
[100]	2019	Lévy Motion	Data clustering problems
[101]	2016	Lévy Motion	Global optimization
[93]	2013	Lévy Motion	Benchmark test function

3.8. Others

Other approaches to improve the diversity, spread, and optimality of the initial population of metaheuristic algorithms exist in the literature. This category includes approaches that used mathematical and statistical functions to aid the initial population in an exhaustive search.

A nonlinear simplex method was used to initialise the swarms [102]. Their results showed that the particles gravitated better towards the excellent quality solutions. An approach where a particle is placed in the centre, and the rest of the particles are spread around it in the search space was considered by [103]. Their result is promising; however, it is not entirely without bias. The use of complex-valued encoding for metaheuristic optimization research is gaining attention from researchers. A comprehensive and extensive overview of this approach is presented in [104].

The complex-valued encoding metaheuristic algorithms have been applied significantly in function optimization, engineering optimization design, and combinatorial optimization. The regular metaheuristic algorithms are based on continuous or discrete encoding. The advantage of the complex-valued encoding metaheuristic algorithm is that it expands the search region and efficiently avoids falling into the local minimum. Finally, eight metaheuristic algorithms were enhanced using the complex-valued encoding, and they were tested using 29 benchmark test functions and five engineering optimisation design problems. The superiority of complex-valued encoding was proved by analysing and comparing the results with statistical significance, and the complex-valued encoding metaheuristic algorithm returned the best performances. We present a summary of what authors have done in this category in Table 8.

Table 8. Other approaches to improving initialising schemes.

Reference	Year	Initialisation Scheme	Optimisation Problem
[61]	2015	Multi-Layer Line Search Methods	Benchmark test function
[40]	2017	Equal Partition and F/T Mutation	Benchmark test function
[44]	2019	Dispatching Rules	Benchmark test function
[47]	2020	Push-Forward Insertion Heuristic	Vehicle routing problem
[45]	2017	Siemens Approximation Method (SAM)	Discrete time-cost trade-off problem
[105]	2020	Angle Probability List strategy	3-D protein structure prediction problem
[106]	2010	Sequential Constructive Crossover Operator	Travelling salesman problem
[107]	2016	Mathematical Programming, Constraint Programming and Machine Learning	Benchmark function
[39]	2013	Randomized Breadth-First Search	Cyclic antibandwidth problem

4. Areas of Application

Much of the research that improved the performance of metaheuristic algorithms, by improving the nature and diversity of the initial population of the algorithms, have been applied in different areas of human endeavour, with significant successes recorded. Figure 10 gives the various application areas of the articles that were found in the literature.

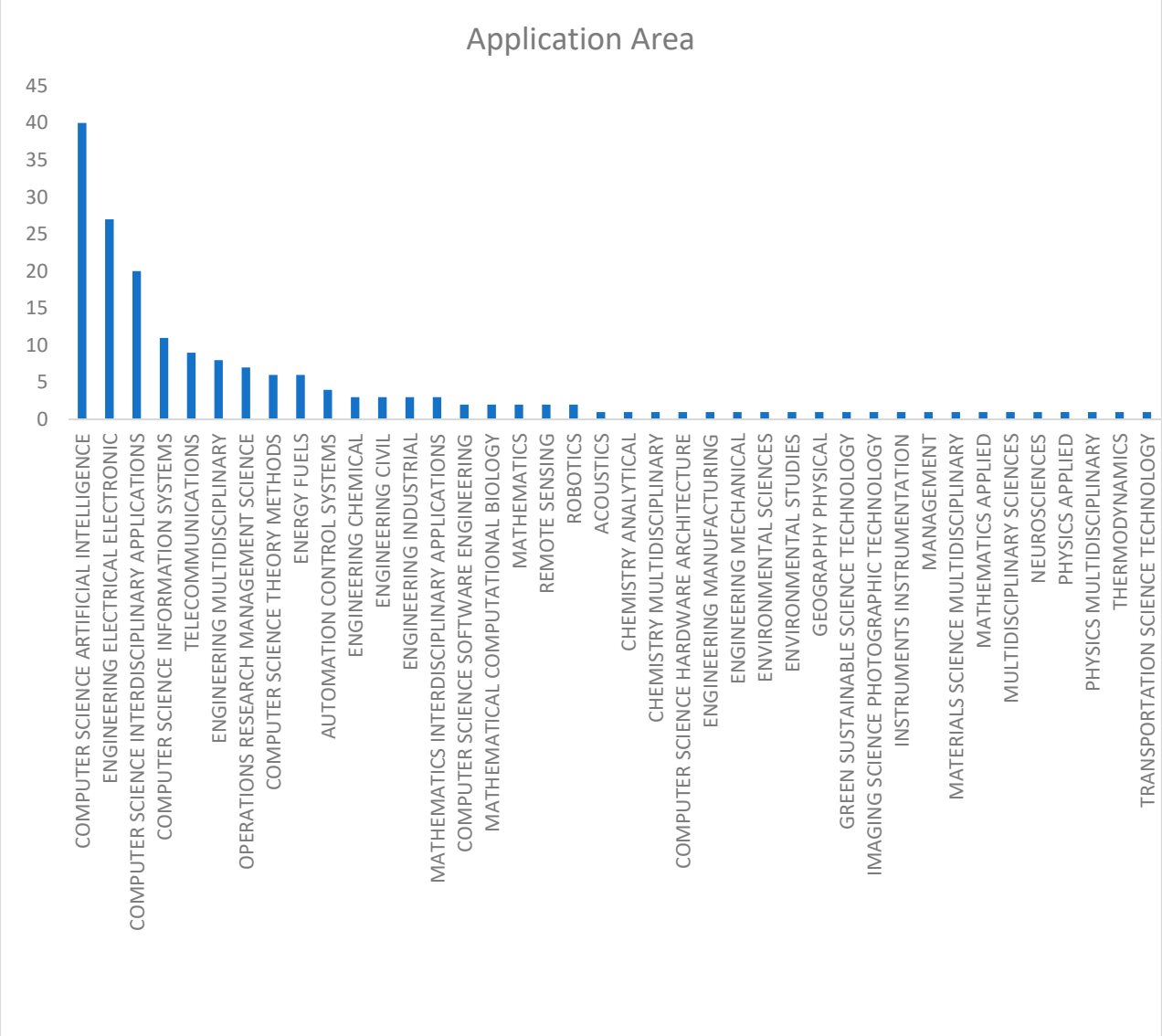

Figure 10. Application areas reported in the literature.

4.1. Computer Science

Figure 10 shows the computer science subcategory as having the highest number of publications, and this can be attributed to the fact that the optimisation problems naturally occur in this area, with over 43 articles published in this area in journals indexed in Scopus and over 60 articles published in journals indexed in WoS. This means that the vast majority of these improvements are applied to solve optimisation problems in computer science, particularly in the area of artificial intelligence. The area of artificial intelligence alone has about 40 articles that are indexed in WoS; this is the most researched area in computer science. The most cited paper in this area is that of [67], who proposed a hybrid of differential evolution and greedy algorithm to exploit the advantages of both methods to improve initialisation, among other improvements. This was used in solving the multi-skill resource-constrained project scheduling problem, and it has been cited 30 times. The hybrid of metaheuristic algorithms is the most common initialisation approach used, apart from the random number generator. The chaos theory and low discrepancy sequences are also popular in this area of application.

4.2. Engineering

Optimisation problems naturally occur in engineering with many metaheuristic algorithms being used to solve problems in this domain; further, this area has the second-highest number of publications. WoS subdivided this category into electrical, electronic, multidisciplinary, industrial, manufacturing, telecommunication, and mechanical categories, whereas Scopus combined them into one category. The sub-area of electrical electronics is the most researched area, with over 25 articles indexed in WoS. A total of 12 articles are indexed in Scopus and over 30 articles in WoS. The most cited article in this category is by [65], in which the authors developed a multi-objective evolutionary algorithm (MOEA)-based proactive-reactive method. This introduced a stability objective, and heuristic initialisation strategies used for the initial solution, and the decision-making approach are also validated. The article was cited 105 times. The area of telecommunications is also well researched and it has nine articles indexed in the repository.

4.3. Mathematics

The area of mathematics has provided the foundation for optimisation techniques used by metaheuristic algorithms. Over 28 articles indexed in Scopus are related to this area, and WoS further divided this category into multidisciplinary, computational biology, interdisciplinary, and applied mathematics. This area intersects with engineering and computer science, and many articles in this category are also classified under these other categories.

4.4. Others

We categorise all the areas with five publications into the category: others. This category comprises automation control systems, remote sensing, robotics, acoustics, chemistry, environmental sciences, management, transportation science technology, energy, neuroscience, and social sciences. Clearly, we see that 90% of articles published in this subject area and indexed in WoS or Scopus are primarily applied or solved problems in the area of computer science and engineering. Great diversity in the application areas can also be alluded to, as can be seen in pockets of research that fall under other categories with few publications.

5. Experiment, Result, and Discussion

5.1. Experimental Setup

This section presents the three (3) different metaheuristic algorithms and ten (10) initialisation schemes used in our work. The choice of these algorithms and initialisation methods is based on their performances in solving optimisation problems, the availability of codes online, and part of many other algorithms and initialisation methods we are using in our current research projects. Table 9 summarises these algorithms, including the year the article was first published, the authors, and the application area of the first publication. Tables 10 and 11 summarise the ten initialisation schemes and the control parameters of the algorithms as were used for the experiments, respectively.

Table 9. Summary of algorithms used.

S/N	Algorithm	Authors and Year of Publication	Application Area (When the Algorithm Was First Published/Proposed)
1	BA	[108]	Benchmark test function
2	GWO	[109]	Benchmark test function, tension/compression spring, welded beam, pressure vessel designs, and optical engineering
3	BOA	[110]	Benchmark test functions, spring design, welded beam design, and gear train design

Table 10. Initialisation schemes.

S/N	Initialization Scheme	Function
1	Random	rand
2	Beta	Betarnd(3, 2)
3	Beta	Betarnd(2.5, 2.5)
4	Uniform	Unifrnd(0, 1)
5	Logarithmic normal	Lognrnd(0, 0.5)
6	Exponential	Exprnd(0.5)
7	Rayleigh	Raylrnd(0.4)
8	Weibull	Wblrnd(1, 1)
9	Latin hypercube sampling	lhsdesign
10	Sobol	Sobol

Table 11. Algorithm-specific parameters.

S/N	Algorithm	Parameters
1	BA	A = rand(N,1), r = rand(N,1), alpha = 0.5, gamma = 0.5, and ro = 0.001.
2	GWO	Alpha_pos = zeros (1, dim), Alpha_score = inf, Beta_pos = zeros (1, dim), Beta_score = inf, Delta_pos = zeros (1, dim), and Delta_score = inf.
3	BOA	Probability switch (p) = 0.8, power_exponent = 0.1, and sensory_modality = 0.01

The variation of the population size and number of iterations are as given in Table 12. The variation is such that a large population size goes with a small number of iterations and vice versa. We also included situations where the two are relatively even.

Table 12. Population size and number of iterations.

Initialization Parameters	Values							
Population size	10	20	30	50	100	300	500	1000
Number of iterations	1000	900	800	600	500	300	100	10

We also conducted a series of experiments to evaluate the effect of the initialisation schemes presented in Table 10 on the three metaheuristic algorithms. We carried out the experiments on ten classical test functions, namely: sphere, quartic, Zakharov, Schwefel 1.2, Booth, Michalewicz, Rastrigin, Rosenbrock, Griewank, and Ackley, consisting of a wide variety of separable, unimodal, non-separable multimodal, numbers of local optima, and multi-dimensional problems. F1 and F2 are unimodal and separable benchmark functions with dimension (D) set at 30. Additionally, F3 and F4 have dimensions set at 30D and are unimodal and non-separable benchmark functions. Similarly, F5, F6, and F7 are multimodal and separable benchmark functions with dimensions set at 2D, 10D, and 30D, respectively. The multimodal and non-separable benchmark functions are F8, F9, and F10, with dimensions set to 30D.

All algorithms were implemented in MATLAB R2019a, and the experiments were conducted using Windows 10 OS, Intel Core i7-8550U CPU, 16G RAM. The number of maximum iterations is set at 1000, and the number of independent runs is set at 20. We round up any solution value less than 10^{-8} to zero, and the results are reported using the following performance indicators: best, worst, mean, standard deviation, and the algorithm mean runtime. We then statistically compared the results from the experiments using Friedman's test and post-hoc analysis, based on the Wilcoxon signed ranks test.

5.2. Results and Discussion

The experiment results on the effect of population size and the maximum number of iterations on the metaheuristic algorithms considered are presented in Tables 13–15. The Friedman test results for all of the results are given in Table 16. This showed a statistically significant difference in the effect of population size and number of iterations for all algorithms tested. The chi-square and p-value as shown in Table 16, and all the p-values are less than the tolerance level of 0.05. Post hoc analysis with Wilcoxon signed-rank tests was conducted with a Bonferroni correction applied, resulting in a significance level set at $p < 0.001$.

The test results for BA are shown in Table 13. We noted that the best results are returned when the population size is 1000, the number of iterations is 10, and it has the lowest mean rank, as shown in the corresponding column in Table 16. Further post hoc results confirmed that a significant difference occurred between this comparison. The implication is that BA performed better with larger population sizes. Similarly, the results for GWO are given in Table 14, and it can be seen that GWO failed to return the optimal solution for Rosenbrock. However, it performed optimally when the population size was 50, the number of iterations was 600, and the lowest mean rank was recorded in this category. A further post hoc test confirmed that GWO performed better when the number of iterations was greater. The results for BOA are presented in Table 15, and we noted that excellent results are returned for small population sizes. The least mean rank is returned when the population size is 30, and the number of iterations is 800. The post hoc test confirmed that BOA performs optimally for a greater number of iterations.

Table 13. Result for Bat Algorithm.

Function	Value	Bat Algorithm							
		Pop = 10	**Pop = 20**	**Pop = 30**	**Pop = 50**	**Pop = 100**	**Pop = 300**	**Pop = 500**	**Pop = 1000**
		Iter = 1000	**Iter = 900**	**Iter = 800**	**Iter = 600**	**Iter = 500**	**Iter = 300**	**Iter = 100**	**Iter = 10**
Sphere	Mean	1.94E+04	1.68E+04	1.59E+04	1.55E+04	1.43E+04	1.1882E+04	1.1242E+04	1.0300E+04
	Stand.Div	1.98E+04	1.72E+04	1.64E+04	1.61E+04	1.46E+04	1.2227E+04	1.1441E+04	1.0369E+04
	Best	1.15E+04	7.26E+03	1.05E+04	9.01E+03	8.02E+03	5.5988E+03	7.3909E+03	7.0207E+03
	Worst	2.78E+04	2.38E+04	2.46E+04	2.42E+04	1.98E+04	1.6507E+04	1.6244E+04	1.1869E+04
	MeanRunTimes	1.53E+00	2.86E+00	7.23E+00	8.54E+00	1.64E+01	1.1428E+01	12.8373	6.8391
Rastrigin	Mean	3.16E+02	3.04E+02	2.90E+02	2.83E+02	2.77E+02	2.7082E+02	264.1494	252.9087
	Stand.Div	3.18E+02	3.05E+02	2.91E+02	2.84E+02	2.78E+02	2.7126E+02	264.4724	253.7484
	Best	2.15E+02	2.73E+02	2.43E+02	2.43E+02	2.16E+02	2.3388E+02	238.5740	206.2512
	Worst	3.73E+02	3.59E+02	3.36E+02	3.14E+02	3.29E+02	2.9137E+02	285.5669	289.4242
	MeanRunTimes	1.85E+00	3.22E+00	4.54E+00	6.18E+00	9.79E+00	1.1708E+01	13.1386	6.9371
Rosenbrock	Mean	2.96E+07	2.26E+07	1.76E+07	1.40E+07	1.52E+07	9.0472E+06	7.6228E+06	6.7477E+06
	Stand.Div	3.40E+07	2.42E+07	2.00E+07	1.51E+07	1.59E+07	9.8445E+06	8.0408E+06	6.9390E+06
	Best	1.09E+07	6.00E+06	6.26E+06	2.39E+06	5.92E+06	2.5159E+06	3.8750E+06	3.8616E+06
	Worst	7.99E+07	4.30E+07	4.11E+07	2.49E+07	2.55E+07	1.6890E+07	1.3239E+07	9.3581E+06
	MeanRunTimes	1.45E+00	3.09E+00	6.45E+00	4.81E+00	1.14E+01	1.1363E+01	12.6469	6.7982
Griewank	Mean	1.94E+02	1.67E+02	1.54E+02	1.43E+02	1.18E+02	1.1163E+02	93.7006	94.5310
	Stand.Div	2.01E+02	1.71E+02	1.57E+02	1.46E+02	1.21E+02	1.1527E+02	95.0456	97.1846
	Best	1.09E+02	1.09E+02	7.05E+01	9.88E+01	7.94E+01	4.0860E+01	65.6311	49.3315
	Worst	3.16E+02	2.51E+02	2.09E+02	2.01E+02	1.76E+02	1.6464E+02	134.4593	139.8530
	MeanRunTimes	1.94E+00	3.54E+00	7.19E+00	8.81E+00	1.66E+01	1.6149E+01	13.4474	7.1280

Table 14. Result for Grey Wolf Optimizer.

Function	Value	Grey Wolf Optimizer							
		Pop = 10	**Pop = 20**	**Pop = 30**	**Pop = 50**	**Pop = 100**	**Pop = 300**	**Pop = 500**	**Pop = 1000**
		Iter = 1000	**Iter = 900**	**Iter = 800**	**Iter = 600**	**Iter = 500**	**Iter = 300**	**Iter = 100**	**Iter = 10**
Sphere	Mean	0	0	0	0	0	0	2.3000E-06	50.3358
	Stand.Div	0	0	0	0	0	0	2.7289E-06	53.0968
	Best	0	0	0	0	0	0	5.6828E-07	21.0005
	Worst	0	0	0	0	0	0	5.6884E-06	91.1243
	MeanRunTimes	1.59E+00	2.91E+00	6.88E+00	6.50E+00	9.91E+00	4.5197E+00	3.6921	1.4936
Rastrigin	Mean	0	0	0	0	0	5.0086E+00	14.6950	84.8322
	Stand.Div	0	0	0	0	0	6.7312E+00	15.1616	86.5337
	Best	0	0	0	0	0	2.6057E-08	8.6052	53.7818
	Worst	0	0	0	0	0	1.4721E+01	20.8869	115.8360
	MeanRunTimes	1.73E+00	2.87E+00	3.88E+00	4.31E+00	5.75E+00	4.6616E+00	3.9513	1.6358
Rosenbrock	Mean	2.56E+01	2.56E+01	2.56E+01	2.52E+01	2.54E+01	2.5108E+01	26.5188	1.3708E+03
	Stand.Div	2.56E+01	2.56E+01	2.56E+01	2.52E+01	2.54E+01	2.5115E+01	26.5425	1.6446E+03
	Best	2.41E+01	2.42E+01	2.41E+01	2.40E+01	2.42E+01	2.4025E+01	25.0393	429.0925
	Worst	2.62E+01	2.62E+01	2.70E+01	2.61E+01	2.62E+01	2.6104E+01	28.7730	4.0747E+03
	MeanRunTimes	1.52E+00	3.23E+00	5.52E+00	3.80E+00	6.96E+00	4.4560E+00	3.5834	1.4551
Griewank	Mean	3.76E-04	6.21E-04	9.90E-04	0	0	6.0142E-03	0.0056	1.4594
	Stand.Div	1.68E-03	2.78E-03	3.13E-03	0	0	1.4804E-02	0.0092	1.4767
	Best	0	0	0	0	0	0	1.6031E-06	1.1595
	Worst	7.52E-03	1.24E-02	9.91E-03	0	0	6.0169E-02	0.0211	1.9960
	MeanRunTimes	1.73E+00	3.02E+00	5.34E+00	6.25E+00	9.52E+00	6.3599E+00	3.8414	1.7894

The results of experiments that were conducted to show the effect of 10 different initialisation schemes on the algorithms are presented, and the findings in the experiments are discussed. The best, worst, mean, standard deviation, and the mean runtime results obtained from the experiments are shown in Tables 17–19. It can be seen from the results, that the ten different initialisation schemes have a different effect on the performance of the algorithms. For some functions, the results are better than others. For some functions, the results appeared to be inconsistent because, while the best value is accurate, the mean value seemed to be inaccurate. The inconsistency could mean that the initial population is close to the global optimum when the best value was returned. It could also mean that the diversity is best suited to the function, hence, its ability to yield a good result. In other cases, more iterations might be needed, or a different diverse population might be used to achieve the desired result.

Table 15. Result for Butterfly Optimization Algorithm.

Function	Value	Butterfly Optimization Algorithm							
		Pop = 10	**Pop = 20**	**Pop = 30**	**Pop = 50**	**Pop = 100**	**Pop = 300**	**Pop = 500**	**Pop = 1000**
		Iter = 1000	**Iter = 900**	**Iter = 800**	**Iter = 600**	**Iter = 500**	**Iter = 300**	**Iter = 100**	**Iter = 10**
Sphere	Mean	0	0	0	0	0	1.6922E-08	4.9084E-05	7.6957E-06
	Stand.Div	0	0	0	0	0	1.7860E-08	5.0361E-05	1.3852E-05
	Best	0	0	0	0	0	1.0376E-08	3.0436E-05	3.0197E-07
	Worst	0	0	0	0	0	3.2348E-08	7.5007E-05	5.0646E-05
	MeanRunTimes	1.06E+00	2.71E+00	7.82E+00	1.11E+01	2.68E+01	2.2878E+01	29.6371	22.1779
Rastrigin	Mean	0	0	0	0	0	4.7966E-06	0.0028	1.5485
	Stand.Div	0	0	0	0	0	5.3872E-06	0.0029	1.5518
	Best	0	0	0	0	0	1.5685E-06	0.0013	1.3340
	Worst	0	0	0	0	0	1.1829E-05	0.0041	1.7193
	MeanRunTimes	1.33E+00	3.02E+00	5.16E+00	7.31E+00	1.78E+01	2.3616E+01	29.8871	22.4889
Rosenbrock	Mean	2.89E+01	2.89E+01	2.88E+01	2.88E+01	2.88E+01	2.8787E+01	28.7735	28.9139
	Stand.Div	2.89E+01	2.89E+01	2.88E+01	2.88E+01	2.88E+01	2.8787E+01	28.7735	28.9139
	Best	2.88E+01	2.88E+01	2.88E+01	2.88E+01	2.87E+01	2.8750E+01	28.7340	28.8728
	Worst	2.89E+01	2.89E+01	2.89E+01	2.89E+01	2.89E+01	2.8832E+01	28.8062	28.9504
	MeanRunTimes	9.82E-01	2.85E+00	6.86E+00	6.82E+00	1.91E+01	2.3152E+01	31.1730	22.1920
Griewank	Mean	0	0	0	0	0	5.7050E-06	0.0068	1.1715
	Stand.Div	0	0	0	0	0	5.7511E-06	0.0068	1.1715
	Best	0	0	0	0	0	4.0656E-06	0.0059	1.1497
	Worst	0	0	0	0	0	6.9312E-06	0.0079	1.1881
	MeanRunTimes	1.29E+00	3.06E+00	7.07E+00	1.09E+01	2.58E+01	3.8891E+01	31.5429	22.5601

Table 16. Friedman test result.

	BA	**GWO**	**BOA**
Pop = 10 Iter = 1000	7.11	5.04	4.89
Pop = 20 Iter = 900	6.30	4.07	4.22
Pop = 30 Iter = 800	5.59	4.63	3.26
Pop = 50 Iter = 600	5.19	3.65	3.67
Pop = 100 Iter = 500	4.30	3.91	3.35
Pop = 300 Iter = 300	3.30	3.70	4.43
Pop = 500 Iter = 100	2.70	5.13	5.67
Pop = 1000 Iter = 10	1.52	5.87	6.52
N	10	10	10
Chi-Square	113.914	29.445	49.249
df	7	7	7
Asymp. Sig.	0.000	0.000	0.000

Table 17. Results for BA.

Function	Value	Rand	Betarnd(3,2)	Betarnd(2.5,2.5)	Unifrnd(0,1)	Lognrnd(0,0.5)	Exprnd(0.5)	Raylrnd(0.4)	Wblrnd(1,1)	Lhsdesign()	Sobol()
F1	Mean	9.79E+03	4.94E+03	4.57E+03	9.51E+03	2.53E+04	1.78E+04	6.78E+03	3.43E+04	9.35E+03	3.02E+03
	Stand.Div	9.93E+03	5.04E+03	4.65E+03	9.75E+03	2.60E+04	1.80E+04	6.91E+03	3.47E+04	9.56E+03	3.12E+03
	Best	6.89E+03	3.44E+03	3.23E+03	5.93E+03	1.28E+04	1.26E+04	4.05E+03	2.34E+04	4.85E+03	9.86E+02
	Worst	1.43E+04	6.85E+03	6.47E+03	1.35E+04	3.82E+04	2.14E+04	9.18E+03	4.34E+04	1.14E+04	4.45E+03
	MeanRunTimes	6.79E+00	6.86E+00	6.87E+00	6.82E+00	8.20E+00	7.64E+00	6.84E+00	8.16E+00	9.98E+00	6.79E+00
F2	Mean	4.93E+00	1.20E+00	1.26E+00	4.65E+00	1.37E+01	1.18E+01	2.43E+00	4.22E+01	5.65E+00	8.74E-01
	Stand.Div	5.26E+00	1.28E+00	1.34E+00	5.14E+00	1.49E+01	1.30E+01	2.58E+00	4.59E+01	6.21E+00	1.04E+00
	Best	1.33E+00	3.75E-01	3.22E-01	1.46E+00	5.48E+00	5.24E+00	9.30E-01	1.26E+01	2.27E+00	1.61E-01
	Worst	8.00E+00	1.88E+00	2.04E+00	9.69E+00	2.88E+01	2.41E+01	3.73E+00	7.73E+01	1.10E+01	2.39E+00
	MeanRunTimes	7.89E+00	7.97E+00	8.04E+00	7.95E+00	9.37E+00	9.15E+00	8.07E+00	9.29E+00	1.12E+01	7.88E+00
F3	Mean	2.03E+01	4.50E+01	1.23E+01	2.11E+01	1.35E+02	2.05E+01	1.18E+01	6.27E+01	2.20E+01	5.54E+01
	Stand.Div	2.12E+01	4.93E+01	1.29E+01	2.22E+01	1.44E+02	2.18E+01	1.27E+01	6.59E+01	2.37E+01	6.63E+01
	Best	7.76E+00	1.65E+01	4.65E+00	1.10E+01	5.61E+01	7.22E+00	3.91E+00	3.13E+01	1.04E+01	1.72E+01
	Worst	3.41E+01	8.45E+01	2.32E+01	3.61E+01	2.49E+02	3.24E+01	2.27E+01	1.22E+02	4.43E+01	1.71E+02
	MeanRunTimes	9.68E-01	9.75E-01	9.83E-01	9.91E-01	1.13E+00	1.05E+00	9.77E-01	1.11E+00	1.28E+00	9.94E-01
F4	Mean	1.61E+04	1.64E+04	7.44E+03	1.77E+04	5.91E+05	3.53E+04	1.08E+04	6.20E+04	1.83E+04	1.56E+04
	Stand.Div	1.65E+04	1.69E+04	7.61E+03	1.79E+04	6.68E+05	3.61E+04	1.10E+04	6.57E+04	1.89E+04	1.98E+04
	Best	6.73E+03	6.15E+03	5.21E+03	1.21E+04	1.83E+05	2.19E+04	6.90E+03	2.15E+04	9.53E+03	6.28E+03
	Worst	2.26E+04	2.33E+04	1.05E+04	2.33E+04	1.69E+06	4.86E+04	1.59E+04	1.02E+05	2.66E+04	5.95E+04
	MeanRunTimes	1.13E+01	1.13E+01	1.13E+01	1.13E+01	1.26E+01	1.17E+01	1.14E+01	1.25E+01	1.49E+01	1.14E+01
F5	Mean	1.99E-02	1.80E-02	2.85E-02	5.89E-02	8.43E-02	5.46E-02	2.62E-02	8.63E-02	5.15E-02	4.24E-01
	Stand.Div	2.86E-02	2.74E-02	5.32E-02	9.10E-02	1.28E-01	8.41E-02	6.90E-02	1.58E-01	8.19E-02	8.22E-01
	Best	1.21E-04	2.95E-04	3.61E-06	9.74E-04	6.15E-04	2.58E-04	1.41E-04	2.31E-05	1.45E-04	1.62E-04
	Worst	8.47E-02	9.53E-02	1.70E-01	2.66E-01	3.59E-01	2.49E-01	2.96E-01	5.17E-01	1.89E-01	2.50E+00
	MeanRunTimes	8.48E-02	8.51E-02	8.72E-02	8.84E-02	9.44E-02	8.82E-02	8.59E-02	9.13E-02	1.02E-01	8.77E-02

Table 17. *Cont.*

Function	Value	Rand	Betarnd(3,2)	Betarnd(2.5,2.5)	Unifrnd(0,1)	Lognrnd(0,0.5)	Exprnd(0.5)	Raylrnd(0.4)	Wblrnd(1,1)	Lhsdesign()	Sobol()
F6	Mean	4.62E+00	4.03E+00	3.86E+00	4.47E+00	5.49E+00	4.74E+00	4.26E+00	5.15E+00	4.61E+00	4.92E+00
	Stand.Div	4.62E+00	4.08E+00	3.88E+00	4.49E+00	5.50E+00	4.76E+00	4.27E+00	5.17E+00	4.63E+00	4.94E+00
	Best	3.85E+00	2.35E+00	3.09E+00	3.41E+00	4.94E+00	3.80E+00	3.61E+00	3.68E+00	3.69E+00	4.06E+00
	Worst	5.15E+00	4.81E+00	4.46E+00	5.15E+00	5.90E+00	5.41E+00	4.84E+00	5.80E+00	5.12E+00	5.94E+00
	MeanRunTimes	1.07E+00	1.08E+00	1.07E+00	1.08E+00	1.18E+00	1.11E+00	1.07E+00	1.13E+00	1.38E+00	1.09E+00
F7	Mean	2.62E+02	2.14E+02	2.13E+02	2.58E+02	4.06E+02	2.98E+02	2.29E+02	3.72E+02	2.53E+02	2.22E+02
	Stand.Div	2.62E+02	2.15E+02	2.14E+02	2.58E+02	4.08E+02	2.99E+02	2.29E+02	3.72E+02	2.54E+02	2.22E+02
	Best	2.35E+02	1.76E+02	1.88E+02	2.23E+02	3.26E+02	2.60E+02	2.02E+02	3.27E+02	1.94E+02	1.93E+02
	Worst	2.98E+02	2.40E+02	2.50E+02	2.94E+02	4.65E+02	3.31E+02	2.55E+02	4.37E+02	2.83E+02	2.45E+02
	MeanRunTimes	6.95E+00	6.92E+00	6.92E+00	7.00E+00	8.38E+00	8.08E+00	7.03E+00	8.32E+00	1.01E+01	7.04E+00
F8	Mean	6.69E+06	1.76E+06	1.29E+06	5.86E+06	5.90E+07	1.49E+07	3.33E+06	7.78E+07	5.77E+06	7.60E+05
	Stand.Div	7.39E+06	1.89E+06	1.39E+06	6.25E+06	6.65E+07	1.54E+07	3.56E+06	8.29E+07	6.26E+06	8.66E+05
	Best	3.00E+06	8.85E+05	3.77E+05	2.36E+06	1.49E+07	6.32E+06	1.10E+06	3.32E+07	2.24E+06	1.89E+05
	Worst	1.25E+07	3.17E+06	2.31E+06	1.07E+07	1.31E+08	2.21E+07	5.41E+06	1.37E+08	1.12E+07	1.70E+06
	MeanRunTimes	6.75E+00	6.86E+00	6.86E+00	6.79E+00	8.16E+00	7.97E+00	6.91E+00	8.11E+00	9.98E+00	6.81E+00
F9	Mean	8.88E+01	4.60E+01	3.74E+01	9.50E+01	2.42E+02	1.60E+02	6.30E+01	2.92E+02	9.00E+01	3.06E+01
	Stand.Div	9.06E+01	4.67E+01	3.79E+01	9.73E+01	2.48E+02	1.63E+02	6.45E+01	2.95E+02	9.23E+01	3.29E+01
	Best	3.99E+01	3.10E+01	2.80E+01	5.21E+01	1.48E+02	1.12E+02	3.82E+01	2.19E+02	5.93E+01	1.77E+01
	Worst	1.18E+02	6.18E+01	4.67E+01	1.36E+02	3.64E+02	2.38E+02	8.79E+01	3.70E+02	1.31E+02	5.72E+01
	MeanRunTimes	7.08E+00	7.14E+00	7.17E+00	7.09E+00	8.47E+00	7.94E+00	7.14E+00	8.47E+00	1.03E+01	7.13E+00
F10	Mean	1.53E+01	1.36E+01	1.27E+01	1.56E+01	2.00E+01	1.86E+01	1.39E+01	1.99E+01	1.57E+01	1.04E+01
	Stand.Div	1.53E+01	1.36E+01	1.27E+01	1.56E+01	2.00E+01	1.86E+01	1.40E+01	1.99E+01	1.58E+01	1.05E+01
	Best	1.38E+01	1.16E+01	1.17E+01	1.38E+01	2.00E+01	1.60E+01	1.18E+01	1.96E+01	1.44E+01	8.63E+00
	Worst	1.63E+01	1.48E+01	1.40E+01	1.66E+01	2.00E+01	2.00E+01	1.51E+01	2.00E+01	1.69E+01	1.31E+01
	MeanRunTimes	8.22E+00	7.84E+00	7.61E+00	8.26E+00	8.24E+00	8.25E+00	8.07E+00	8.26E+00	1.14E+01	8.24E+00

Table 18. Results for BOA.

Function	Value	Rand	Betarnd(3,2)	Betarnd(2.5,2.5)	Unifrnd(0,1)	Lognrnd(0,0.5)	Exprnd(0.5)	Raylrnd(0.4)	Wblrnd(1,1)	Lhsdesign()	Sobol()
F1	Mean	0	0	0	0	0	0	0	0	0	0
	Stand.Div	0	0	0	0	0	0	0	0	0	0
	Best	0	0	0	0	0	0	0	0	0	0
	Worst	0	0	0	0	0	0	0	0	0	0
	MeanRunTimes	4.01E+00	4.42E+00	4.23E+00	4.05E+00	4.24E+00	3.97E+00	4.03E+00	4.01E+00	4.26E+00	4.06E+00
F2	Mean	0	0	0	0	0	0	0	0	0	0
	Stand.Div	0	0	0	0	0	0	0	0	0	0
	Best	0	0	0	0	0	0	0	0	0	0
	Worst	0	0	0	0	0	0	0	0	0	0
	MeanRunTimes	7.35E+00	7.78E+00	7.66E+00	7.38E+00	7.65E+00	7.35E+00	7.36E+00	7.38E+00	7.59E+00	7.45E+00
F3	Mean	0	0	0	0	0	0	0	0	0	0
	Stand.Div	0	0	0	0	0	0	0	0	0	0
	Best	0	0	0	0	0	0	0	0	0	0
	Worst	0	0	0	0	0	0	0	0	0	0
	MeanRunTimes	7.31E-01	7.84E-01	7.78E-01	7.47E-01	7.84E-01	7.33E-01	7.42E-01	7.36E-01	7.69E-01	7.53E-01
F4	Mean	0	0	0	0	0	0	0	0	0	0
	Stand.Div	0	0	0	0	0	0	0	0	0	0
	Best	0	0	0	0	0	0	0	0	0	0
	Worst	0	0	0	0	0	0	0	0	0	0
	MeanRunTimes	1.63E+01	1.67E+01	1.65E+01	1.62E+01	1.66E+01	1.63E+01	1.63E+01	1.65E+01	1.72E+01	1.64E+01
F5	Mean	1.04E-07	8.27E-08	1.03E-07	9.58E-08	5.76E-08	1.18E-07	1.30E-07	9.02E-08	1.20E-07	1.02E-07
	Stand.Div	1.41E-07	1.22E-07	1.42E-07	1.14E-07	7.62E-08	1.71E-07	1.97E-07	1.23E-07	1.71E-07	1.40E-07
	Best	1.33E-08	0	0	0	0	1.00E-08	1.50E-08	0	0	0
	Worst	3.58E-07	4.13E-07	3.11E-07	2.25E-07	1.57E-07	5.57E-07	7.05E-07	3.28E-07	5.12E-07	3.76E-07
	MeanRunTimes	8.82E-02	9.21E-02	9.15E-02	9.03E-02	9.10E-02	8.95E-02	8.96E-02	8.96E-02	9.08E-02	9.02E-02

Table 18. *Cont.*

Function	Value	Rand	Betarnd(3,2)	Betarnd(2.5,2.5)	Unifrnd(0,1)	Lognrnd(0,0.5)	Exprnd(0.5)	Raylrnd(0.4)	Wblrnd(1,1)	Lhsdesign()	Sobol()
F6	Mean	1.40E+00	1.01E+00	1.33E+00	1.22E+00	1.33E+00	1.27E+00	1.24E+00	1.23E+00	1.44E+00	1.48E+00
	Stand.Div	1.63E+00	1.35E+00	1.56E+00	1.36E+00	1.55E+00	1.47E+00	1.36E+00	1.41E+00	1.63E+00	1.66E+00
	Best	2.64E-01	1.23E-01	5.07E-01	1.48E-01	4.09E-01	3.69E-01	2.35E-01	1.48E-01	5.29E-01	4.74E-01
	Worst	3.15E+00	3.59E+00	3.38E+00	2.52E+00	3.26E+00	2.99E+00	2.31E+00	2.69E+00	3.73E+00	3.14E+00
	MeanRunTimes	1.11E+00	1.18E+00	1.15E+00	1.12E+00	1.15E+00	1.12E+00	1.12E+00	1.12E+00	1.14E+00	1.13E+00
F7	Mean	0	0	0	0	0	0	0	0	0	0
	Stand.Div	0	0	0	0	0	0	0	0	0	0
	Best	0	0	0	0	0	0	0	0	0	0
	Worst	0	0	0	0	0	0	0	0	0	0
	MeanRunTimes	4.11E+00	4.51E+00	4.35E+00	4.16E+00	4.35E+00	4.09E+00	4.15E+00	4.13E+00	4.29E+00	4.20E+00
F8	Mean	2.52E+01	2.51E+01	2.53E+01	2.51E+01	2.50E+01	2.51E+01	2.52E+01	2.51E+01	2.51E+01	2.53E+01
	Stand.Div	2.53E+01	2.51E+01	2.53E+01	2.51E+01	2.50E+01	2.51E+01	2.52E+01	2.52E+01	2.51E+01	2.53E+01
	Best	2.37E+01	2.38E+01	2.39E+01	2.33E+01	2.36E+01	2.42E+01	2.39E+01	2.40E+01	2.42E+01	2.41E+01
	Worst	2.62E+01	2.70E+01	2.62E+01	2.62E+01	2.62E+01	2.62E+01	2.62E+01	2.62E+01	2.60E+01	2.61E+01
	MeanRunTimes	3.95E+00	4.35E+00	4.14E+00	3.96E+00	4.14E+00	3.92E+00	3.97E+00	3.95E+00	4.20E+00	4.00E+00
F9	Mean	0	0	0	0	0	3.73E-04	0	0	3.73E-04	0
	Stand.Div	0	0	0	0	0	1.67E-03	0	0	1.67E-03	0
	Best	0	0	0	0	0	0	0	0	0	0
	Worst	0	0	0	0	0	7.46E-03	0	0	7.46E-03	0
	MeanRunTimes	4.27E+00	4.67E+00	4.48E+00	4.28E+00	4.49E+00	4.27E+00	4.27E+00	4.33E+00	4.42E+00	4.32E+00
F10	Mean	0	0	0	0	2.02E+01	0	0	2.04E+00	0	0
	Stand.Div	0	0	0	0	2.02E+01	0	0	6.44E+00	0	0
	Best	0	0	0	0	2.02E+01	0	0	0	0	0
	Worst	0	0	0	0	2.03E+01	0	0	2.04E+01	0	0
	MeanRunTimes	4.11E+00	4.51E+00	4.33E+00	4.12E+00	4.89E+00	4.06E+00	4.14E+00	4.25E+00	4.25E+00	4.18E+00

Table 19. Results for GWO.

Function	Value	Rand	Betarnd(3,2)	Betarnd(2.5,2.5)	Unifrnd(0,1)	Lognrnd(0,0.5)	Exprnd(0.5)	Raylrnd(0.4)	Wblrnd(1,1)	Lhsdesign()	Sobol()
F1	Mean	0	0	0	0	0	0	0	0	0	0
	Stand.Div	0	0	0	0	0	0	0	0	0	0
	Best	0	0	0	0	0	0	0	0	0	0
	Worst	0	0	0	0	0	0	0	0	0	0
	MeanRunTimes	3.76E+00	3.77E+00	3.77E+00	3.77E+00	3.78E+00	3.74E+00	3.71E+00	3.74E+00	3.75E+00	3.74E+00
F2	Mean	0	0	0	0	0	0	0	0	0	0
	Stand.Div	0	0	0	0	0	0	0	0	0	0
	Best	0	0	0	0	0	0	0	0	0	0
	Worst	0	0	0	0	0	0	0	0	0	0
	MeanRunTimes	6.49E+00	6.47E+00	6.53E+00	6.53E+00	6.52E+00	6.49E+00	6.42E+00	6.45E+00	6.50E+00	6.48E+00
F3	Mean	0	0	0	0	0	0	0	0	0	0
	Stand.Div	0	0	0	0	0	0	0	0	0	0
	Best	0	0	0	0	0	0	0	0	0	0
	Worst	0	0	0	0	0	0	0	0	0	0
	MeanRunTimes	6.24E-01	6.24E-01	6.31E-01	6.33E-01	6.37E-01	6.28E-01	6.25E-01	6.22E-01	6.31E-01	6.32E-01
F4	Mean	0	0	0	0	0	0	0	0	0	0
	Stand.Div	0	0	0	0	0	0	0	0	0	0
	Best	0	0	0	0	0	0	0	0	0	0
	Worst	0	0	0	0	0	0	0	0	0	0
	MeanRunTimes	1.36E+01	1.37E+01	1.37E+01	1.35E+01	1.38E+01	1.37E+01	1.36E+01	1.37E+01	1.43E+01	1.37E+01
F5	Mean	1.28E-03	7.20E-04	8.16E-04	8.13E-04	8.42E-04	9.36E-04	1.09E-03	1.57E-03	1.13E-03	5.43E-04
	Stand.Div	2.02E-03	9.37E-04	1.09E-03	1.07E-03	1.12E-03	1.71E-03	1.67E-03	2.34E-03	1.74E-03	8.19E-04
	Best	1.29E-04	5.74E-05	3.47E-05	8.95E-05	4.32E-05	2.07E-05	2.22E-05	5.83E-05	5.98E-06	2.78E-06
	Worst	7.04E-03	2.35E-03	3.25E-03	3.39E-03	2.83E-03	4.76E-03	4.36E-03	6.90E-03	3.88E-03	2.68E-03
	MeanRunTimes	7.42E-02	7.41E-02	7.48E-02	7.47E-02	7.48E-02	7.51E-02	7.40E-02	7.39E-02	7.58E-02	7.38E-02

Table 19. *Cont.*

Function	Value	Rand	Betarnd(3,2)	Betarnd(2.5,2.5)	Unifrnd(0,1)	Lognrnd(0,0.5)	Exprnd(0.5)	Raylrnd(0.4)	Wblrnd(1,1)	Lhsdesign()	Sobol()
F6	Mean	4.41E+00	3.89E+00	3.91E+00	4.14E+00	3.80E+00	4.67E+00	4.28E+00	4.33E+00	4.22E+00	4.25E+00
	Stand.Div	4.42E+00	3.90E+00	3.92E+00	4.16E+00	3.82E+00	4.68E+00	4.29E+00	4.35E+00	4.25E+00	4.27E+00
	Best	3.73E+00	3.36E+00	3.29E+00	3.34E+00	2.55E+00	3.83E+00	3.55E+00	3.60E+00	2.67E+00	3.29E+00
	Worst	4.76E+00	4.53E+00	4.40E+00	4.77E+00	4.37E+00	5.37E+00	4.63E+00	5.01E+00	4.89E+00	4.84E+00
	MeanRunTimes	9.23E-01	9.31E-01	9.30E-01	9.18E-01	9.44E-01	9.16E-01	9.21E-01	9.24E-01	9.27E-01	9.16E-01
F7	Mean	0	0	0	0	0	0	0	0	0	0
	Stand.Div	0	0	0	0	0	0	0	0	0	0
	Best	0	0	0	0	0	0	0	0	0	0
	Worst	0	0	0	0	0	0	0	0	0	0
	MeanRunTimes	4.14E+00	4.12E+00	4.17E+00	4.14E+00	4.15E+00	4.15E+00	4.15E+00	4.13E+00	4.15E+00	4.14E+00
F8	Mean	2.88E+01	2.87E+01	2.88E+01	2.88E+01	2.87E+01	2.89E+01	2.89E+01	2.87E+01	2.88E+01	2.88E+01
	Stand.Div	2.88E+01	2.87E+01	2.88E+01	2.88E+01	2.87E+01	2.89E+01	2.89E+01	2.87E+01	2.88E+01	2.88E+01
	Best	2.88E+01	2.87E+01	2.88E+01	2.88E+01	2.86E+01	2.89E+01	2.88E+01	2.87E+01	2.88E+01	2.88E+01
	Worst	2.89E+01	2.87E+01	2.89E+01	2.89E+01	2.87E+01	2.89E+01	2.89E+01	2.87E+01	2.89E+01	2.89E+01
	MeanRunTimes	3.69E+00	3.69E+00	3.68E+00	3.67E+00	3.70E+00	3.68E+00	3.65E+00	3.68E+00	3.69E+00	3.67E+00
F9	Mean	0	0	0	0	0	0	0	0	0	0
	Stand.Div	0	0	0	0	0	0	0	0	0	0
	Best	0	0	0	0	0	0	0	0	0	0
	Worst	0	0	0	0	0	0	0	0	0	0
	MeanRunTimes	4.20E+00	4.19E+00	4.20E+00	4.18E+00	4.19E+00	4.16E+00	4.18E+00	4.15E+00	4.19E+00	4.16E+00
F10	Mean	0	0	0	0	0	0	0	0	0	0
	Stand.Div	0	0	0	0	0	0	0	0	0	0
	Best	0	0	0	0	0	0	0	0	0	0
	Worst	0	0	0	0	0	0	0	0	0	0
	MeanRunTimes	3.97E+00	3.97E+00	3.97E+00	3.96E+00	4.04E+00	3.99E+00	3.95E+00	4.07E+00	3.97E+00	3.97E+00

The results for experiments conducted on BA are given in Table 17. The betarnd(3,2), betarnd(2.5,2.5), raylrnd(0.4), and sobol outperformed the rand for most functions. To obtain the general effect of the initialisation schemes on BA, we treated the ten initialisation schemes as observations for the Friedman's test, and the summary is given in the corresponding column in Table 20. The p-value is 0.000, which is less than $\alpha = 0.05$, and hence we rejected the hypothesis. This means that the performance of BA is sensitive to the initialisation schemes. After a post hoc test based on the Wilcoxon signed ranks test of all the initialisation schemes using a Bonferroni correction with a significance level set at $p < 0.001$, the betarnd(2.5,2.5) returned the lowest mean and is ranked first and, therefore, we recommended betarnd(2.5,2.5) for BA.

Table 20. Friedman's test for classical functions.

	BA	BOA	GWO
	Mean Rank	**Mean Rank**	**Mean Rank**
rand	7.10	12.35	12.30
betarnd(3,2)	4.20	9.25	9.90
betarnd(2.5,2.5)	2.70	12.05	10.90
unifrnd(0,1)	8.00	9.65	11.20
lognrnd(0,0.5)	17.70	10.60	9.50
exprnd(0.5)	10.00	12.40	13.10
raylrnd(0.4)	4.90	12.25	12.40
wblrnd(1,1)	16.20	10.90	12.30
lhsdesign()	7.90	12.90	12.20
Sobol()	5.70	12.65	11.20
Test Statistics [a]			
N	10	10	10
Chi-Square	160.917	33.892	25.217
df	22	22	22
Asymp. Sig.	0.000	0.050	0.287

[a] Friedman's Test.

The result for BOA is shown in Table 18, and it showed that the betarnd(3,2) and unifrnd(0,1) are the best performing initialisation schemes. As shown in Table 20, BOA has a p-value of 0.050, which is equal to $\alpha = 0.05$, hence we retained the hypothesis. This means that BOA is not sensitive to the initialisation schemes. Similarly, the results for GWO (Table 19) showed that lognrnd(0,0.5) and betarnd(3,2) are the best performing initialisation schemes. The Friedman's test result showed that the p-value is 0.287, which is greater than $\alpha = 0.05$, hence we retained the hypothesis. This means that BOA is not sensitive to the initialisation schemes.

6. Conclusions

So many works exist in the literature that clearly outline the nature of the role of the initial population in the overall performance of metaheuristic algorithms. However, despite the role that initialisation plays and the efforts put forward by researchers in this research area, to our knowledge, no comprehensive survey of articles on the subject area exists. Therefore, the present study presents a comprehensive survey of different approaches to improving performances of metaheuristic optimizers, using their initialisation scheme. We also show the publication trends for research in this area, and the number of citations. Finally, we provided a glossary of efforts that have been made to improve the performance of metaheuristic algorithms using their initialisation scheme. We also include the areas of application of these improvements for easy reference by metaheuristic research enthusiasts.

The number of articles published to date in the repositories that were discussed earlier showed that the area which focuses on the initialisation of the population of metaheuristic algorithms is relatively uncharted. Many of these metaheuristic algorithms have been proposed; however, less effort has been made regarding their initialisation scheme. Most

researchers opt for the commonly used random number generator whose disadvantages have been significantly studied. The ease of implementation of the random number generator may have contributed to its use by researchers. On the one hand, the hybridisation of metaheuristic algorithms has yielded great results in the literature. Authors have had a great degree of success in using different initialisation schemes for the algorithms. We see a promising avenue whereby researchers can explore these high-performing initialisation schemes to assess their efficacy. The size of the population and the iteration number can be varied along with these schemes. This can help in increasing the performance of the algorithms.

Our experiments demonstrate that for the classical functions under consideration, BA is sensitive to the initialisation schemes, whereas GWO and BOA are not. The sensitivity of the algorithms is also problem-dependent, meaning that some functions were insensitive to the initialisation scheme. The population size and number of iterations play a role in the performance of the algorithms. We discovered that BA performed better with larger population sizes. GWO and BOA performed better when the number of iterations was greater. This conclusion is heavily dependent on the dimension problem; however, we believe that good population diversity and number of iterations will most likely lead to optimal solutions.

We also identified the need for an initialisation method for these algorithms that are best suited to the specific problem domain with statistical backing to yield an optimal solution for that set of problems. Unfortunately, most papers on meta-heuristics usually perform very little statistical validation, and if they do it is only on a single problem that the researchers describe. Benchmarking meta-heuristics with systematic and sound statistical techniques is usually lacking from many published works in the literature. In addition, a tuning/adaptive scheme could be developed, and this scheme should be capable of choosing an initialisation method from a suite of initialisation schemes that will lead to better solutions, depending on the nature of the problem encountered. This approach will also lead to the diversity of the population.

Author Contributions: Conceptualization, J.O.A. and A.E.E.; methodology, J.O.A. and A.E.E.; software, J.O.A. and A.E.E.; validation, J.O.A. and A.E.E.; formal analysis, J.O.A. and A.E.E.; investigation, J.O.A. and A.E.E.; writing—J.O.A.; writing—A.E.E.; supervision, A.E.E.; project administration, A.E.E. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Yang, X.S. Social Algorithms. In *Encyclopedia of Complexity and Systems Science*; Meyers, R.A., Ed.; Springer: Berlin/Heidelberg, Germany, 2017. [CrossRef]
2. Ezugwu, A.E.; Shukla, A.K.; Nath, R.; Akinyelu, A.A.; Agushaka, J.O.; Chiroma, H.; Muhuri, P.K. Metaheuristics: A comprehensive overview and classification along with bibliometric analysis. *Artif. Intell. Rev.* **2021**, *54*, 4237–4316. [CrossRef]
3. Ezugwu, A.E.; Adeleke, O.J.; Akinyelu, A.A.; Viriri, S. A conceptual comparison of several metaheuristic algorithms on continuous optimization problems. *Neural Comput. Appl.* **2020**, *32*, 6207–6251. [CrossRef]
4. Dokeroglu, T.; Sevinc, E.; Kucukyilmaz, T.; Cosar, A. A survey on new generation metaheuristic algorithms. *Comput. Ind. Eng.* **2019**, *137*, 106040. [CrossRef]
5. Kondamadugula, S.; Naidu, S.R. Accelerated evolutionary algorithms with parameter importance based population initialization for variation-aware analog yield optimization. In Proceedings of the 2016 IEEE 59th International Midwest Symposium on Circuits and Systems (MWSCAS), Abu Dhabi, United Arab Emirates, 16–19 October 2016.
6. Elsayed, S.; Sarker, R.; Coello, C.A.C. Sequence-based deterministic initialization for evolutionary algorithms. *IEEE Trans. Cybern.* **2016**, *47*, 2911–2923. [CrossRef] [PubMed]

7. Tzanetos, A.; Dounias, G. Nature inspired optimization algorithms or simply variations of metaheuristics. *Artif. Intell. Rev.* **2021**, *54*, 1841–1862. [CrossRef]
8. Pant, M.; Thangaraj, R.; Grosan, C.; Abraham, A. Improved particle swarm optimization with low-discrepancy sequences. In Proceedings of the 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), Hong Kong, China, 1–6 June 2008.
9. Gentle, J. *Random Number Generation and Monte Carlo Methods*; Springer Science & Business Media: Boston, MA, USA, 2006.
10. Agushaka, J.O.; Ezugwu, A.E. Advanced Arithmetic Optimization Algorithm for solving mechanical engineering design problems. *PLoS ONE* **2021**, *16*, e0255703. [CrossRef] [PubMed]
11. Imran, M.; Hashima, R.; Khalid, N.E.A. An overview of particle swarm optimization variants. *Procedia Eng.* **2013**, *53*, 491–496. [CrossRef]
12. Osaba, E.; Carballedo, R.; Diaz, F.; Onieva, E.; Lopez, P.; Perallos, A. On the influence of using initialization functions on genetic algorithms solving combinatorial optimization problems: A first study on the TSP. In Proceedings of the 2014 IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS), Linz, Austria, 2–4 June 2014.
13. Li, Q.; Liu, S.Y.; Yang, X.S. Influence of initialization on the performance of metaheuristic optimizers. *Appl. Soft Comput.* **2020**, *91*, 106193. [CrossRef]
14. Weidt Neiva, F.; de Souza da Silva, R.L. *Systematic Literature Review in Computer Science—A Practical Guide*; Technical Report of Computer Science Department DCC/UFJF RelaTeDCC 002/2016; Federal University of Juiz de Fora: Juiz de Fora, Brazil, 2016.
15. Jauro, F.; Chiroma, H.; Gital, A.; Almutairi, M.; Shafi'i, M.; Abawajy, J. Deep learning architectures in emerging cloud computing architectures: Recent development, challenges and next research trend. *Appl. Soft Comput.* **2020**, *96*, 106582. [CrossRef]
16. Cantú-Paz, E. On random numbers and the performance of genetic algorithms. *Comput. Sci. Prepr. Arch.* **2002**, *2002*, 203–210.
17. Daida, J.; Ross, S.; McClain, J.; Ampy, D.; Holczer, M. Challenges with verification, repeatability, and meaningful comparisons in genetic programming. In *Genetic Programming 1997: Proceedings of the Second Annual Conference*; Morgan Kaufmann Publishers: San Francisco, CA, USA, 1997.
18. Wang, X.; Hickernell, F. Randomized halton sequences. *Math. Comput. Model.* **2000**, *32*, 887–899. [CrossRef]
19. Niederreiter, H. *Random Number Generation and Quasi-Monte Carlo Methods*; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 1992.
20. Morokoff, W.; Caflisch, R. Quasirandom sequences and their discrepancies. *SIAM J. Sci. Comput.* **1994**, *15*, 1251–1279. [CrossRef]
21. Uy, N.Q.; Hoai, N.; McKay, R.; Tuan, P. Initialising PSO with randomized low-discrepancy sequences: The comparative results. In Proceedings of the 2007 IEEE Congress on Evolutionary Computation, Singapore, 25–28 September 2007.
22. Agushaka, J.; Ezugwu, A. Influence of Initializing Krill Herd Algorithm with Low-Discrepancy Sequences. *IEEE Access* **2020**, *8*, 210886–210909. [CrossRef]
23. Brits, R.; Engelbrecht, A.; van den Bergh, F. A niching particle swarm optimizer. In Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning, Singapore, 18–22 November 2002; Volume 2.
24. Covic, N.; Lacevic, B. Wingsuit flying search—A novel global optimization algorithm. *IEEE Access* **2020**, *8*, 53883–53900. [CrossRef]
25. Bangyal, W.H.; Ahmad, J.; Rauf, H.T.; Pervaiz, S. An improved bat algorithm based on novel initialization technique for global optimization problem. *Int. J. Adv. Comput. Sci. Appl.* **2018**, *9*, 158–166. [CrossRef]
26. Kimura, S.; Matsumura, K. Genetic algorithms using low-discrepancy sequences. In Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation, Washington, DC, USA, 25–29 June 2005.
27. Kucherenko, S.; Sytsko, Y. Application of deterministic low-discrepancy sequences in global optimization. *Comput. Optim. Appl.* **2005**, *30*, 297–318. [CrossRef]
28. Thangaraj, R.; Pant, M.; Abraham, A.; Badr, Y. Hybrid evolutionary algorithm for solving global optimization problems. In Proceedings of the International Conference on Hybrid Artificial Intelligence Systems, Salamanca, Spain, 10–12 June 2009; Springer: Berlin/Heidelberg, Germany, 2009.
29. Bangyal, W.H.; Ahmad, J.; Rauf, H.T. Comparison of Different Bat Initialization Techniques for Global Optimization Problems. *Int. J. Appl. Metaheuristic Comput.* **2021**, *12*, 157–184. [CrossRef]
30. Nakib, A.; Daachi, B.; Siarry, P. Hybrid differential evolution using low-discrepancy sequences for image segmentation. In Proceedings of the 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum, Shanghai, China, 21–25 May 2012.
31. Georgioudakis, M.; Lagaros, N.D.; Papadrakakis, M. Probabilistic shape design optimization of structural components under fatigue. *Comput. Struct.* **2017**, *182*, 252–266. [CrossRef]
32. Mosbah, H.; El-Hawary, M.E. Optimization of neural network parameters by Stochastic Fractal Search for dynamic state estimation under communication failure. *Electr. Power Syst. Res.* **2017**, *147*, 288–301. [CrossRef]
33. Wood, D.-A. Hybrid cuckoo search optimization algorithms applied to complex wellbore trajectories aided by dynamic, chaos-enhanced, fat-tailed distribution sampling and metaheuristic profiling. *J. Nat. Gas Sci. Eng.* **2016**, *34*, 236–252. [CrossRef]
34. Shanmugam, G.; Ganesan, P.; Vanathi, D.P. Meta heuristic algorithms for vehicle routing problem with stochastic demands. *J. Comput. Sci.* **2011**, *7*, 533. [CrossRef]
35. de Melo, V.V.; Delbem, A.C.B. Investigating smart sampling as a population initialization method for differential evolution in continuous problems. *Inf. Sci.* **2012**, *193*, 36–53. [CrossRef]

36. Rauf, H.T.; Bangyal, W.H.; Ahmad, J.; Bangyal, S.A. Training of artificial neural network using pso with novel initialization technique. In Proceedings of the 2018 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT), Zallaq, Bahrain, 18–20 November 2018.
37. Bui, D.T.; Pradhan, B.; Nampak, H.; Bui, Q.T.; Tran, Q.A.; Nguyen, Q.P. Hybrid artificial intelligence approach based on neural fuzzy inference model and metaheuristic optimization for flood susceptibilitgy modeling in a high-frequency tropical cyclone area using GIS. *J. Hydrol.* **2016**, *540*, 317–330.
38. Termeh, S.V.R.; Khosravi, K.; Sartaj, M.; Keesstra, S.D.; Tsai, F.T.C.; Dijksma, R.; Pham, B.T. Optimization of an adaptive neuro-fuzzy inference system for groundwater potential mapping. *Hydrogeol. J.* **2019**, *27*, 2511–2534. [CrossRef]
39. Lozano, M.; Duarte, A.; Gortázar, F.; Martí, R. A hybrid metaheuristic for the cyclic antibandwidth problem. *Knowl.-Based Syst.* **2013**, *54*, 103–113. [CrossRef]
40. Wang, G.G.; Hao, G.S.; Cheng, S.; Cui, Z. An improved monarch butterfly optimization with equal partition and f/t mutation. In Proceedings of the International Conference on Swarm Intelligence, Hong Kong, China, 25–27 March 2017.
41. Hodashinsky, I.A.; Filimonenko, I.V.; Sarin, K.S. Krill herd and piecewise-linear initialization algorithms for designing Takagi–Sugeno systems. *Optoelectron. Instrum. Data Process.* **2017**, *53*, 379–387. [CrossRef]
42. Jiang, Y.; Shao, Z.; Guo, Y.; Zhang, H.; Niu, K. Drscro: A metaheuristic algorithm for task scheduling on heterogeneous systems. *Math. Probl. Eng.* **2015**, *2015*, 396582. [CrossRef]
43. Kang, T.; Yao, J.; Jin, M.; Yang, S.; Duong, T. A novel improved cuckoo search algorithm for parameter estimation of photovoltaic (PV) models. *Energies* **2018**, *11*, 1060. [CrossRef]
44. Vlašić, I.; Đurasević, M.; Jakobović, D. Improving genetic algorithm performance by population initialization with dispatching rules. *Comput. Ind. Eng.* **2019**, *137*, 106030. [CrossRef]
45. Aminbakhsh, S.; Sonmez, R. Pareto front particle swarm optimizer for discrete time-cost trade-off problem. *J. Comput. Civ. Eng.* **2017**, *31*, 04016040. [CrossRef]
46. Wijayanto, A.W.; Purwarianti, A. Improvement design of fuzzy geo-demographic clustering using Artificial Bee Colony optimization. In Proceedings of the 2014 International Conference on Cyber and IT Service Management (CITSM), Bali, Indonesia, 8–10 August 2014.
47. Han, Y.Q.; Li, J.Q.; Liu, Z.; Liu, C.; Tian, J. Metaheuristic algorithm for solving the multiobjective vehicle routing problem with time window and drones. *Int. J. Adv. Robot. Syst.* **2020**, *17*, 1729881420920031. [CrossRef]
48. Xiang, W.L.; Meng, X.L.; An, M.Q.; Li, Y.Z.; Gao, M.X. An enhanced differential evolution algorithm based on multiple mutation strategies. *Comput. Intell. Neurosci.* **2015**, *2015*, 285730. [CrossRef]
49. Yeoh, J.M.; Caraffini, F.; Homapour, E.S.V.; Milani, A. A clustering system for dynamic data streams based on metaheuristic optimization. *Mathematics* **2019**, *7*, 1229. [CrossRef]
50. Carrizales-Turrubiates, O.; Rangel-Valdez, N.; Torres-Jiménez, J. Optimal shortening of covering arrays. In Proceedings of the Mexican International Conference on Artificial Intelligence, Puebla, Mexico, 26 November–4 December 2011; Springer: Berlin/Heidelberg, Germany, 2011.
51. Mandal, D.; Chatterjee, A.; Maitra, M. Robust medical image segmentation using particle swarm optimization aided level set based global fitting energy active contour approach. *Eng. Appl. Artif. Intell.* **2014**, *35*, 199–214. [CrossRef]
52. Benaichouche, A.N.; Oulhadj, H.; Siarry, P. Improved spatial fuzzy c-means clustering for image segmentation using PSO initialization, Mahalanobis distance and post-segmentation correction. *Digit. Signal Process.* **2013**, *23*, 1390–1400. [CrossRef]
53. Gallardo, J.E.; Cotta, C. A GRASP-based memetic algorithm with path relinking for the far from most string problem. *Eng. Appl. Artif. Intell.* **2015**, *41*, 183–194. [CrossRef]
54. Kohler, M.; Vellasco, M.M.; Tanscheit, R. PSO+: A new particle swarm optimization algorithm for constrained problems. *Appl. Soft Comput.* **2019**, *85*, 105865. [CrossRef]
55. Eshtay, M.; Faris, H.; Obeid, N. A competitive swarm optimizer with hybrid encoding for simultaneously optimizing the weights and structure of Extreme Learning Machines for classification problems. *Int. J. Mach. Learn. Cybern.* **2020**, *11*, 1801–1823. [CrossRef]
56. Sawant, S.S.; Prabukumar, M.; Samiappan, S. A band selection method for hyperspectral image classification based on cuckoo search algorithm with correlation based initialization. In Proceedings of the 2019 10th Workshop on Hyperspectral Imaging and Signal Processing: Evolution in Remote Sensing (WHISPERS), Amsterdam, The Netherlands, 24–26 September 2019.
57. Lin, L.; Ji, Z.; He, S.; Zhu, Z. A crown jewel defense strategy based particle swarm optimization. In Proceedings of the 2012 IEEE Congress on Evolutionary Computation, Brisbane, Australia, 10–15 June 2012.
58. Sun, Y.; Qi, X. A DE-LS Metaheuristic Algorithm for Hybrid Flow-Shop Scheduling Problem considering Multiple Requirements of Customers. *Sci. Program.* **2020**, *2020*, 8811391. [CrossRef]
59. El-Abd, M. Global-best brain storm optimization algorithm. *Swarm Evol. Comput.* **2017**, *37*, 27–44. [CrossRef]
60. Giuliani, D. A Grayscale Segmentation Approach Using the Firefly Algorithm and the Gaussian Mixture Model. *Int. J. Swarm Intell. Res.* **2018**, *9*, 39–57. [CrossRef]
61. Ivorra, B.; Mohammadi, B.; Ramos, A.M. A multi-layer line search method to improve the initialization of optimization algorithms. *Eur. J. Oper. Res.* **2015**, *247*, 711–720. [CrossRef]
62. Zainuddin, Z.; Ong, P. Optimization of wavelet neural networks with the firefly algorithm for approximation problems. *Neural Comput. Appl.* **2017**, *28*, 1715–1728. [CrossRef]

63. Li, W.; Özcan, E.; John, R. A learning automata-based multiobjective hyper-heuristic. *IEEE Trans. Evol. Comput.* **2017**, *23*, 59–73. [CrossRef]
64. Mehrmolaei, S.; Keyvanpour, M.R.; Savargiv, M. Metaheuristics on time series clustering problem: Theoretical and empirical evaluation. *Evol. Intell.* **2020**. [CrossRef]
65. Shen, X.N.; Yao, X. Mathematical modeling and multiobjective evolutionary algorithms applied to dynamic flexible job shop scheduling problems. *Inf. Sci.* **2015**, *298*, 198–224. [CrossRef]
66. Xiang, W.L.; An, M.Q.; Li, Y.Z.; He, R.C.; Zhang, J.F. An improved global-best harmony search algorithm for faster optimization. *Expert Syst. Appl.* **2014**, *41*, 5788–5803. [CrossRef]
67. Myszkowski, P.B.; Olech, Ł.P.; Laszczyk, M.; Skowroński, M.E. Hybrid differential evolution and greedy algorithm (DEGR) for solving multi-skill resource-constrained project scheduling problem. *Appl. Soft Comput.* **2018**, *62*, 1–14. [CrossRef]
68. Aqil, S.; Allali, K. Local search metaheuristic for solving hybrid flow shop problem in slabs and beams manufacturing. *Expert Syst. Appl.* **2020**, *162*, 113716. [CrossRef]
69. Tavazoei, M.S.; Haeri, M. Comparison of different one-dimensional maps as chaotic search pattern in chaos optimization algorithms. *Appl. Math. Comput.* **2007**, *187*, 1076–1085. [CrossRef]
70. Suresh, S.; Lal, S.; Reddy, C.S.; Kiran, M.S. A novel adaptive cuckoo search algorithm for contrast enhancement of satellite images. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2017**, *10*, 3665–3676. [CrossRef]
71. Afrabandpey, H.; Ghaffari, M.; Mirzaei, A.; Safayani, M. A novel bat algorithm based on chaos for optimization tasks. In Proceedings of the 2014 Iranian Conference on Intelligent Systems (ICIS), Bam, Iran, 4–6 February 2014.
72. Zhang, Q.; Chen, H.; Luo, J.; Xu, Y.; Wu, C.; Li, C. Chaos enhanced bacterial foraging optimization for global optimization. *IEEE Access* **2018**, *6*, 64905–64919. [CrossRef]
73. Li, B.; Liu, C.; Wu, H.; Zhao, Y.; Dong, Y. Chaotic adaptive butterfly mating optimization and its applications in synthesis and structure optimization of antenna arrays. *Int. J. Antennas Propag.* **2019**, *2019*, 1730868. [CrossRef]
74. Yuan, X.; Miao, Z.; Liu, Z.; Yan, Z.; Zhou, F. Multi-Strategy Ensemble Whale Optimization Algorithm and Its Application to Analog Circuits Intelligent Fault Diagnosis. *Appl. Sci.* **2020**, *10*, 3667. [CrossRef]
75. Wang, M.; Chen, H.; Yang, B.; Zhao, X.; Hu, L.; Cai, Z.; Huang, H.; Tong, C. Toward an optimal kernel extreme learning machine using a chaotic moth-flame optimization strategy with applications in medical diagnoses. *Neurocomputing* **2017**, *267*, 69–84. [CrossRef]
76. Gandomi, A.H.; Yang, X.S.; Talatahari, S.; Alavi, A.H. Firefly algorithm with chaos. *Commun. Nonlinear Sci. Numer. Simul.* **2013**, *18*, 89–98. [CrossRef]
77. Wu, B.; Fan, S.H. Improved artificial bee colony algorithm with chaos. In Proceedings of the International Workshop on Computer Science for Environmental Engineering and EcoInformatics, Kunming, China, 29–31 July 2011; Springer: Berlin/Heidelberg, Germany, 2011.
78. Saremi, S.; Mirjalili, S.; Lewis, A. Biogeography-based optimization with chaos. *Neural Comput. Appl.* **2014**, *25*, 1077–1097. [CrossRef]
79. Wang, G.G.; Guo, L.; Gandomi, A.H.; Hao, G.S.; Wang, H. Chaotic krill herd algorithm. *Inf. Sci.* **2014**, *274*, 17–34. [CrossRef]
80. Heidari, A.A.; Abbaspour, R.A.; Jordehi, A.R. An efficient chaotic water cycle algorithm for optimization tasks. *Neural Comput. Appl.* **2017**, *28*, 57–85. [CrossRef]
81. Kohli, M.; Arora, S. Chaotic grey wolf optimization algorithm for constrained optimization problems. *J. Comput. Des. Eng.* **2018**, *5*, 458–472. [CrossRef]
82. Sayed, G.I.; Khoriba, G.; Haggag, M.H. A novel chaotic salp swarm algorithm for global optimization and feature selection. *Appl. Intell.* **2018**, *48*, 3462–3481. [CrossRef]
83. Anter, A.M.; Ali, M. Feature selection strategy based on hybrid crow search optimization algorithm integrated with chaos theory and fuzzy c-means algorithm for medical diagnosis problems. *Soft Comput.* **2020**, *24*, 1565–1584. [CrossRef]
84. Kaveh, A.; Sheikholeslami, R.; Talatahari, S.; Keshvari-Ilkhichi, M. Chaotic swarming of particles: A new method for size optimization of truss structures. *Adv. Eng. Softw.* **2014**, *67*, 136–147. [CrossRef]
85. Liu, F.; Duan, H.; Deng, Y. A chaotic quantum-behaved particle swarm optimization based on lateral inhibition for image matching. *Optik* **2012**, *123*, 1955–1960. [CrossRef]
86. Faia, R.; Pinto, T.; Vale, Z.; Corchado, J.M. An ad-hoc initial solution heuristic for metaheuristic optimization of energy market participation portfolios. *Energies* **2017**, *10*, 883. [CrossRef]
87. Eltamaly, A.M.; Al-Saud, M.S.; Abokhalil, A.G. A Novel Bat Algorithm Strategy for Maximum Power Point Tracker of Photovoltaic Energy Systems under Dynamic Partial Shading. *IEEE Access* **2020**, *8*, 10048–10060. [CrossRef]
88. Yao, L.U.; You, S.U.N.; Xiaodong, L.I.U.; Bo, G.A.O. Control allocation for a class of morphing aircraft with integer constraints based on Lévy flight. *J. Syst. Eng. Electron.* **2020**, *31*, 826–840. [CrossRef]
89. Correia, S.D.; Beko, M.; Tomic, S.; Cruz, L.A.D.S. Energy-Based Acoustic Localization by Improved Elephant Herding Optimization. *IEEE Access* **2020**, *8*, 28548–28559. [CrossRef]
90. Eltamaly, A.M.; Al-Saud, M.S.; Abo-Khalil, A.G. Performance Improvement of PV Systems' Maximum Power Point Tracker Based on a Scanning PSO Particle Strategy. *Sustainability* **2020**, *12*, 1185. [CrossRef]
91. Abbas, A.; Hewahi, N.M. Imaging the search space: A nature-inspired metaheuristic extension. *Evol. Intell.* **2020**, *13*, 463–474. [CrossRef]

92. El-Sayed, W.T.; El-Saadany, E.F.; Zeineldin, H.H.; Al-Sumaiti, A.S. Fast initialization methods for the nonconvex economic dispatch problem. *Energy* **2020**, *201*, 117635. [CrossRef]
93. Hussein, W.A.; Sahran, S.; Abdullah, S.N.H.S. A new initialization algorithm for bees algorithm. In Proceedings of the International Multi-Conference on Artificial Intelligence Technology, Shah Alam, Malaysia, 28–29 August 2013; Springer: Berlin/Heidelberg, Germany, 2013.
94. Heidari, A.A.; Pahlavani, P. An efficient modified grey wolf optimizer with Lévy flight for optimization tasks. *Appl. Soft Comput.* **2017**, *60*, 115–134. [CrossRef]
95. Lin, J.H.; Chou, C.W.; Yang, C.H.; Tsai, H.L. A chaotic Levy flight bat algorithm for parameter estimation in nonlinear dynamic biological systems. *Comput. Inf. Technol.* **2012**, *2*, 56–63.
96. Aydoğdu, İ.; Akın, A.; Saka, M.P. Design optimization of real world steel space frames using artificial bee colony algorithm with Levy flight distribution. *Adv. Eng. Softw.* **2016**, *92*, 1–14. [CrossRef]
97. Amirsadri, S.; Mousavirad, S.J.; Ebrahimpour-Komleh, H. A Levy flight-based grey wolf optimizer combined with back-propagation algorithm for neural network training. *Neural Comput. Appl.* **2018**, *30*, 3707–3720. [CrossRef]
98. Barshandeh, S.; Haghzadeh, M. A new hybrid chaotic atom search optimization based on tree-seed algorithm and Levy flight for solving optimization problems. *Eng. Comput.* **2020**, *37*, 3079–3122. [CrossRef]
99. Chegini, S.N.; Bagheri, A.; Najafi, F. PSOSCALF: A new hybrid PSO based on Sine Cosine Algorithm and Levy flight for solving optimization problems. *Appl. Soft Comput.* **2018**, *73*, 697–726. [CrossRef]
100. Abdulwahab, H.A.; Noraziah, A.; Alsewari, A.A.; Salih, S.Q. An enhanced version of black hole algorithm via levy flight for optimization and data clustering problems. *IEEE Access* **2019**, *7*, 142085–142096. [CrossRef]
101. Jensi, R.; Jiji, G.W. An enhanced particle swarm optimization with levy flight for global optimization. *Appl. Soft Comput.* **2016**, *43*, 248–261. [CrossRef]
102. Parsopoulos, K.; Vrahatis, M. Initializing the particle swarm optimizer using the nonlinear simplex method. *Adv. Intell. Syst. Fuzzy Syst. Evol. Comput.* **2002**, *216*, 1–6.
103. Richards, M.; Ventura, D. Choosing a starting configuration for particle swarm optimization. *Neural Netw.* **2004**, *3*, 2309–2312.
104. Wang, P.; Zhou, Y.; Luo, Q.; Han, C.; Niu, Y.; Lei, M. Complex-valued encoding metaheuristic optimization algorithm: A comprehensive survey. *Neurocomputing* **2020**, *407*, 313–342. [CrossRef]
105. De Lima Corrêa, L.; Dorn, M. A multi-population memetic algorithm for the 3-D protein structure prediction problem. *Swarm Evol. Comput.* **2020**, *55*, 100677. [CrossRef]
106. Ahmed, Z.H. Genetic algorithm for the traveling salesman problem using sequential constructive crossover operator. *Int. J. Biom. Bioinform.* **2010**, *3*, 96.
107. Talbi, E.G. Combining metaheuristics with mathematical programming, constraint programming and machine learning. *Ann. Oper. Res.* **2016**, *240*, 171–215. [CrossRef]
108. Yang, X.S. A new metaheuristic bat-inspired algorithm. In *Nature Inspired Cooperative Strategies for Optimization (Nicso 2010)*; Studies in Computational, Intelligence; Gonzalez, J.R., Ed.; Springer: Berlin/Heidelberg, Germany, 2010.
109. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey wolf optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
110. Arora, S.; Singh, S. Butterfly optimization algorithm: A novel approach for global optimization. *Soft Comput.* **2019**, *23*, 715–734. [CrossRef]