algorithms

MDPI

Article

Agent State Flipping Based Hybridization of Heuristic Optimization Algorithms: A Case of Bat Algorithm and Krill Herd Hybrid Algorithm

Robertas Damaševičius [1,*] and Rytis Maskeliūnas [2]

[1] Faculty of Mathematics, Silesian University of Technology, 44-100 Gliwice, Poland
[2] Department of Multimedia Engineering, Kaunas University of Technology, 51368 Kaunas, Lithuania; rytis.maskeliunas@ktu.lt
* Correspondence: robertas.damasevicius@polsl.pl

Abstract: This paper describes a unique meta-heuristic technique for hybridizing bio-inspired heuristic algorithms. The technique is based on altering the state of agents using a logistic probability function that is dependent on an agent's fitness rank. An evaluation using two bio-inspired algorithms (bat algorithm (BA) and krill herd (KH)) and 12 optimization problems (cross-in-tray, rotated hyper-ellipsoid (RHE), sphere, sum of squares, sum of different powers, McCormick, Zakharov, Rosenbrock, De Jong No. 5, Easom, Branin, and Styblinski–Tang) is presented. Furthermore, an experimental evaluation of the proposed scheme using the industrial three-bar truss design problem is presented. The experimental results demonstrate that the hybrid scheme outperformed the baseline algorithms (mean rank for the hybrid BA-KH algorithm is 1.279 vs. 1.958 for KH and 2.763 for BA).

Keywords: hyper-heuristic; meta-heuristic; bio-inspired algorithms; heuristic optimization

check for updates

Citation: Damaševičius, R.; Maskeliūnas, R. Agent State Flipping Based Hybridization of Heuristic Optimization Algorithms: A Case of Bat Algorithm and Krill Herd Hybrid. *Algorithms* **2021**, *14*, 358. https://doi.org/10.3390/a14120358

Academic Editor: Antonio Della Cioppa

Received: 15 November 2021
Accepted: 8 December 2021
Published: 10 December 2021

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

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

In global optimization problems, the objective function often has a high computational complexity, high dimensionality, and non-trivial landscape. Such tasks are of high practical significance, and a lot of methods have been proposed as a basis to find their solutions. Localization capability with a high probability of sub-optimal (close to optimal) solutions is one of the main advantages of this class of algorithms, in addition to their versatility and simplicity of implementation. In practically important optimization problems, it is often sufficient to make such decisions. Some methods are specially developed or are best suited for solving optimal problems with mathematical models of a certain type. Thus, the mathematical apparatus of linear programming is specially designed to solve problems with linear optimality criteria and linear constraints on variables and leads to the solution of most of the problems formulated in this formulation. Dynamic programming is well suited for solving optimization problems of multi-stage processes, especially those in which the state of each stage is defined by a relatively small number of state variables. However, in the presence of a significant number of these variables, that is, with a high dimension in each stage, the use of the dynamic programming method is difficult due to the limited speed and memory capacity of computers. Perhaps the best way to choose the optimization method that is most suitable for solving the corresponding problem is to study the possibilities and experiences of using various optimization methods.

Bio-inspired algorithms are an actively developing area of heuristic optimization and decision-making methods [1]. Meta-heuristic optimization methods motivated by examples in nature are becoming more common in real-life applications. At the moment, the most promising direction can be considered the creation of new versions of bio-inspired algorithms, which take into account problem-oriented information about the search area for optimal solutions, as well as the history of the search. Bio-inspired algorithms exploit

the principles of the collective behavior of living organisms, such as insect swarms or bird flocks. Bio-inspired algorithms, in a broader sense, define the behavior of communicative agents in a multi-agent system. Whereas every agent has a limited processing capability, the system of agents (such as a swarm) as a whole can accomplish complicated tasks, such as searching for resources. Many current meta-heuristic algorithms have been presented that are influenced by natural or social events, such as grey wolf optimizer (GWO) [2], artificial bee colony (ABC) [3], particle swarm optimization (PSO) [4], firefly algorithm (FA) [5], krill herd (KH) [6], polar bear optimization [7], and red fox optimization [8]. Such algorithms are commonly used in various domains, including scientific computing [9], image recognition [10], software bug detection [11], industrial production optimization [12], and disease recognition from biomedical images [13,14].

The No Free Lunch theorem [15], which proved that, for an optimization algorithm, any improvement in performance for a single category of problems leads to a decrease in performance for another category, has motivated the comparison of bio-inspired algorithms for solving various optimization problems. As a rule, it is impossible to recommend the adoption of any one method without exception to solve all problems that arise in practice.

To create new, highly efficient bio-inspired algorithms, a method such as hybridization is used. Hybridization is an evolutionary meta-heuristic approach [16]. Meta-heuristics are flexible and easy to use since they replicate physical or biological events and only address inputs and outcomes. Meta-heuristics are also a type of probabilistic optimization approach. This trait enables them to efficiently avoid local optima, which are common in real-world issues. Meta-heuristic algorithms typically outpace heuristic optimization methods in solving numerous difficult and challenging optimization problems in real life due to their simplicity, flexibility, and ability to escape local optima. In the process of their development, hybrid bio-inspired algorithms can overcome the main disadvantages of classical bio-inspired algorithms.

A simple method of hybridization is to combine two algorithms to create a new one. There may be several heuristics from which to pick when addressing an issue, and each heuristic has its own set of strengths and limitations. The goal is to automatically create algorithms by combining the strengths and correcting the weaknesses of known heuristics. A broader concept is to automatically create new algorithms by combining the strengths and correcting the faults of existing heuristic algorithms. This approach is known as hyper-heuristic, and it aims to automate the selection, combination, or adaptation of numerous simpler heuristics (or parts thereof) to solve computational optimization problems quickly [17].

When compared to the conventional application of meta-heuristics for optimization issues, hyper-heuristics perform at a higher level. In other words, a hyper-heuristic is a higher-level heuristic that acts on lower-level heuristics [18], with the goal of intelligently selecting the best heuristic or technique for a particular situation. Hyper-heuristics frequently employ a "selection function", which leverages the exploitation and exploration operations in order to select the best heuristic to employ [19]. The selection function can be stochastic [20], lower-level heuristic performance based [18], evolutionary [21], or greedy [20]. A greedy selection function favors low-level heuristics with the highest current performance. The decision function, for example, can be described as a third-order tensor of the record of a hyper-heuristic [22]. The factorization of such a tensor shows hidden links between lower-level heuristics and the higher-level heuristic.

The co-algorithmic hybridization of population optimization algorithms is based on the following basic idea. Simultaneously, several sub-populations evolve in the search space, each of which is based on one of the population algorithms and solves the optimization problem. Each of the sub-populations "fights" for computational resources, which, at the end of a given number of iterations, are redistributed in favor of the more efficient sub-population. A co-algorithm can be built around various optimization algorithms, for example, artificial bee colony (ABC) [23] and particle swarm optimization (PSO) [24]. Researchers have also proposed new hybrid algorithms by merging two or more meta-

heuristic algorithms, such as the multistart hyper-heuristic [25], which combines ant colony optimization (ACO), robust tabu search (RTS), simulated annealing (SA), and breakout local search (BLS), and the hybridization of KH and ABC algorithms [26].

This study describes a unique meta-heuristic technique for hybridizing nature-inspired algorithms. The technique is based on altering the state of the agents by employing a logistic function that is dependent on an agent's fitness rank.

The remaining parts of the article are organized follows. Section 2 discusses previous studies. Section 3 presents the proposed methods. Section 4 analyzes the results. Section 5 presents the results of experiments on the industrial engineering design problem of the three-bar truss design. Finally, Section 6 reviews the main findings and concludes this work.

2. Related Works

Heuristic optimization methods can be categorized into evolutionary and behavioral (imitation) methods [27]. Behavioral methods are based on modeling the collective behavior of self-organizing living or non-living systems, the interacting elements of which are called agents. The key ideas of behavioral methods are decentralization, the interaction of agents, and the simplicity of their behavior [28]. To improve the efficiency of behavioral methods of global optimization, two main approaches are currently used—hybridization and meta-optimization. Hybridization combines either different methods or the same methods but with different values of the free parameters so that the effectiveness of one method compensates for the weakness of the other. There are several classifications of hybridization algorithms for behavioral methods of global optimization [29]. According to X. Wang, hybridization is distinguished according to the scheme of the embedded methods, the hybridization of the pre-processor/post-processor type, and the co-algorithm hybridization [30]. High-level embedded hybridization implies a loose coupling of the combined methods. In this case, combined methods retain significant autonomy, so it is relatively easy to single out each of them in the final method. In low-level embedded hybridization, the combined methods are so strongly integrated that it is usually impossible to isolate the components of the final method; i.e., low-level hybridization creates, in fact, a new method.

Behavioral methods of global optimization, particularly hybrid methods, have a large number of free parameters. The effectiveness of a method often depends on the values of these parameters. On the other hand, there are usually no rules for selecting the values of these parameters. Therefore, one of the main approaches to increase the effectiveness of behavioral methods is to develop algorithms for adapting the values of their free parameters to the features of a specific optimization problem or to the features of a certain class of optimization problems—the meta-optimization of behavioral methods. The most famous classifications of meta-optimization algorithms are presented in [31,32]. The idea of one-time parameter-tuning algorithms is that the program implementing the considered optimization method is executed for different values of its free parameters on a large number of optimization problems of a particular class. Based on the research results, the parameter values with the best efficiency indicators are selected, in accordance with which algorithms for parameter tuning and those for parameter control are distinguished [33].

Related to hyper-heuristics are evolutionary algorithms (EAs) [34], which can discover the optimal EA to optimize problem solutions. Self-modification Cartesian genetic programming [35] encodes low-level heuristics with self-modifying operators, and the exponentially expanding hyper-heuristic [36] employs a meta-hyper-heuristic algorithm to seek the heuristic space for better performance. The leader-following consensus problem can be characterized as a combination of heuristics, where the leader is the heuristic with the greatest fitness and is pursued by other heuristics.

We discuss other notable examples from the literature below. Alrassas [37] proposed a modified ANFIS (adaptive neuro-fuzzy inference system) model created using the Aquila Optimizer (AO), a novel optimization technique. The AO is a recently developed optimization algorithm influenced by the natural behavior of Aquila. AO-ANFIS, the created

model, was tested using real-world datasets given by local partners. Furthermore, thorough comparisons to the standard ANFIS model and numerous improved ANFIS models utilizing various optimization strategies were performed. The AO-ANFIS outperformed classic ANFIS and various modified models in terms of numerical results and statistics while also improving ANFIS prediction accuracy.

Helmi et al. [38] proposed a lightweight feature selection (FS) technique to improve classification for a real-world human activity recognition problem. The new FS approach, known as GBOGWO, intends to increase the performance of the gradient-based optimizer (GBO) algorithm by utilizing grey wolf optimizer's (GWO) operators. Support vector machine (SVM) is employed to categorize the activities once GBOGWO has selected suitable characteristics. Extensive tests were carried out to assess the efficiency of GBOGWO utilizing well-known UCI-HAR and WISDM datasets. The results suggested that GBOGWO enhanced the classification accuracy by an average of 98%.

Jouhari et al. [39] proposed a modified Harris hawks optimizer (HHO), providing an efficient method for addressing UPMSPs. The new approach, dubbed MHHO, employs the salp swarm algorithm (SSA) as a local search strategy to improve HHO's performance and reduce its computation time. Several tests were carried out to evaluate the characteristics of MHHO utilizing small and large problem scenarios. Furthermore, the proposed method was compared to other modern UPMSP methodologies. The MHHO outperformed the other methods in both small and large problem scenarios.

In Makhadmeh et al. [40], grey wolf optimizer (GWO), a swarm-based optimization method influenced by grey wolf behavior, was developed to address the power scheduling problem in a smart home (PSPSH). GWO features strong operations that are regulated by dynamic parameters that maintain exploration and exploitation actions in the search space. Simulations were performed using seven scenarios of power consumption and dynamic pricing schemes in order to assess the multi-objective PSPSH employing the SHB (BMO-PSPSH) approach. The performance of the BMO-PSPSH technique was compared to that of 17 other recent algorithms.

Neggaz et al. [41] developed a new salp swarm optimizer (SSA) version that, when paired with the sine cosine algorithm and the disruption operator, adjusts the location of followers in SSA using oscillating functions. This improvement aids in the discovery phase and avoids stagnation in a limited region. The disruption operator is used to increase variation in the population and maintain the balance between exploration and exploitation processes. The experimental outcomes were examined using 20 datasets, 4 of which had high complexity and a limited number of occurrences.

Ksiazek et al. [42] proposed using a modified ant lion optimizer for effective simulation and positioning models to assist human operators with radiation heat transfer in an electric furnace, which is supported by the results of tests that suggest that the proposed algorithm may efficiently settle the system.

Cruz et al. [43] provided an approach for customizing population-based meta-heuristics based on a heuristic model driven by simulated annealing. The method uses search operators from 10 meta-heuristic strategies as building blocks for new strategies. The method was tested on 107 benchmark high-dimensional functions under various experimental settings. The results show that it is feasible to create high-performing meta-heuristics with a variety of settings for each case study in an automated manner.

Turky et al. [44] presented a hyper-heuristic framework comprising numerous local optimization techniques and a pool of neighborhood structures. First, a two-stage hyper-heuristic framework is constructed for choosing a local search method and its operators. Second, an adaptive ranking system selects the best neighborhood constructs for the present local search method. It assesses the benefits of the local search in terms of quality and variety using entropy and adaptively updates the pool of potential neighborhood structures. Third, a population of solutions is employed to efficiently travel to various sections of the solution search space.

Wang et al. [45] proposed a hybrid AO and HHO technique merged with a non-linear escape energy parameter and stochastic opposition-based learning technique. First, integrating the key characteristics of AO and HHO preserves considerable exploration and exploitation opportunities. Second, in the exploitation phase, stochastic opposition-based learning is incorporated to increase the evasion of local optima. Finally, to leverage the exploration and exploitation stages, the non-linear escape energy parameter is used. The hybrid algorithm was assessed on 23 benchmark functions and 4 industrial engineering challenges to validate its optimization performance.

Abd Elaziz et al. [46] proposed a method that relies on the operators of six meta-heuristic algorithms (symbiotic organisms search (SOS), WOA, differential evolution (DE), GWO, SCA, and SSA), which allow solutions to be more diverse. Two benchmarks, the IEEE CEC 2014 and IEEE CEC 2017, were used to assess the technique.

In Dabba et al. [47], the quantum moth flame optimization algorithm (QMFOA), a unique swarm intelligence system for gene selection that is based on the hybridization of quantum computation with the moth flame optimization (MFO) algorithm, was proposed. The QMFOA is a straightforward two-phase technique. The first phase is a pre-processing phase that solves the challenge of high-dimensional data by measuring the redundancy and relevance of the gene in order to acquire the relevant gene set. In order to tackle the gene selection problem, the second phase involves the hybridization of MFOA, quantum computing, and support vector machine (SVM) with leave-one-out cross-validation, among other things. The authors employed quantum computing to provide a good trade-off between exploration and exploitation of the search space, while a new updated moth operation based on Hamming distance and the Archimedes spiral enables rapid exploration of all conceivable gene subsets.

In Huo et al. [48], a hybrid differential symbiotic organisms search (HDSOS) method was developed by merging the differential evolution (DE) mutation approach with modified symbiotic organism search techniques (SOS). The proposed method retains the SOS's local search capacity while also having a high global search capability. Furthermore, a perturbation method is used to improve the algorithm's resilience.

Shehadeh [49] proposed a combination of the gravitational search algorithm (GSA) and "sperm swarm optimization" (SSO), called HSSOGSA. The presented algorithm's basic principles and ideas are to combine the power of exploitation in SSO with the capability of exploration in GSA in order to combine the strengths of both algorithms. The proposed HSSOGSA method was compared to the regular GSA and SSO algorithms. These algorithms were evaluated using two mechanisms: qualitative and quantitative testing. The study used best fitness, standard deviation, and average metrics for the quantitative test, while for the qualitative test, the authors compared the convergence rates attained by the proposed algorithm to the convergence rates achieved by SSO and GSA.

Kundu and Garg [50] proposed an enhanced teaching–learning Harris hawks optimization (ITLHHO) based on teaching–learning-based optimization for tackling various types of engineering design and numerical optimization problems. ITLHHO's performance was demonstrated by using 33 well-known benchmark functions, including IEEE Congress on Evolutionary Computation's CEC-C06 2019 Competition Test Functions and 10 engineering optimization problems, and the significance of the results was demonstrating by statistical analysis with the Wilcoxon rank-sum test and multiple comparison test.

Chiu et al. [51] proposed a hybrid of the sine cosine algorithm and fitness-dependent optimizer (SC-FDO) for updating the velocity (pace) using the sine cosine scheme. SC-FDO, the proposed method, was evaluated on 19 classical and CEC-C06 2019 benchmark test functions. The results showed that SC-FDO outperformed the original FDO and well-known optimization techniques in the majority of scenarios. The proposed SC-FDO improved on the original FDO by obtaining a better exploit–explore trade-off while also attaining a quicker convergence speed.

In Alkhateeb et al. [52], a hybrid cuckoo search and simulated annealing (CSA) approach was proposed. The optimization operators of the simulated annealing technique

are included in the cuckoo search algorithm. They introduced discrete CSA (DCSA), a discrete version of CSA for solving the JSSP, in this study. DCSA introduces four changes to CSA. First, in the initialization stage, it employs the opposition-based learning approach to generate a varied set of candidate solutions. Second, it combines variable neighborhood search (VNS) and Lévy flight techniques to improve search space exploration. Third, it jumps out of local optima using the elite opposition-based learning approach prior to the abandonment stage of CSA. Finally, the lowest position value is used.

In this paper, we introduce a hyper-heuristic for combining the behavior of two independent heuristic algorithms. The combination of the BA [53] and the KH [6] is provided as a case study. Using 12 benchmark optimization challenges, the efficiency of the hyper-heuristic is compared to that of stand-alone BA and KH.

3. Hybridization Method

3.1. General Description

The presented hybridization approach designates the state (corresponding to the behavior) of each agent and proposes an algorithm for how this state evolves during algorithm execution. We presume that an agent's behavior is fully set by its state. If a swarm of agents is under-performing, their states can be adjusted in the hope that the change in behavior will lead to improved performance. As a result, the status of the agents is not consistent and might change from the previous iteration to the next iteration. The hybridization hyper-heuristic pseudo-code is shown below in Algorithm 1.

For simplicity, let there be two possible states (Agent_1 or Agent_2), while their appropriate actions are defined by Algorithms 2 and 3.

Algorithm 1 Proposed hybridization algorithm.

1: Set initial positions of swarming agents
2: Initialize algorithms Algorithms 2 and 3
3: **while** iterations not completed **do**
4: **if** iteration is first **then**
5: Set the agent state to $Agent_1$ or $Agent_2$ randomly
6: **else**
7: Use logistic probability function to flip the state of agents
8: **end if**
9: **for** agent $\in$ state of $Agent_1$ **do**
10: Compute new position using a single step of Algorithm 2
11: **end for**
12: **for** agent $\in$ state of $Agent_2$ **do**
13: Compute new position using a single step of Algorithm 3
14: **end for**
15: Combine both populations
16: **end while**
17: Return the performance of the best agent

The probability of switching the agent's state in Line 6 of the pseudo-code is computed using a common logistic function:

$$P(r) = \frac{1}{1 + e^{-(r-n/2)}} \tag{1}$$

where r is the performance rank of an operating agent (agents are arranged from best-performing to worst-performing agents by their fitness value), and n is the total count of agents in the population.

In our further experiments, Algorithm 1 is krill herd (KH) and Algorithm 2 is the bat algorithm (BA), which are explained below.

3.2. Krill Herd (KH)

Krill herd (KH) is a swarm intelligence optimization approach influenced by Antarctic krill herding behavior and takes into account krill's Lagrangian and evolutionary activities in their native habitat [6]. Krill can form enormous herds that can span hundreds of meters in length. Whenever predators (such as penguins or other birds) attack krill herds, they remove a krill, resulting in a lower krill density. Following a predator assault, krill formation is a multi-objective process aimed at increasing the krill density and accessing food. The krill swarm density, krill distance, and food density all influence the goal function in KH. The krill position is determined by the following factors: (i) movement caused by other krill, (ii) foraging movement, and (iii) physical diffusion. When looking for the maximum density and food, every individual krill migrates towards the best feasible solution. The smallest distances between individual krill and food, as well as the largest density of the herd, are regarded as the objective function for krill movement, which eventually leads to krill herding around the global minima.

The KH method uses the Lagrangian model in a d-dimensional solution space as follows:

$$\frac{dX_i}{dt} = N_i + F_i + D_i \tag{2}$$

where N_i, F_i, and D_i are the movement caused by other krill, foraging movement, and physical scattering of the ith krill.

The direction of motion, $alpha_i$, of the movement caused by other krill is approximated by the target swarm density, local swarm density, and repulsive swarm density. This movement may be characterized for a krill individual as shown below, where N_max is the greatest induced speed, $omega_n$ is the motion inertia weight in $[0, 1]$, and N_i^{old} is the final movement.

$$N_i^{new} = N^{max}\alpha_i + \omega_n N_i^{old} \tag{3}$$

The two basic components estimate the foraging motion. The first is the food position, and the second is prior knowledge of the food position. This movement may be approximated for the ith krill as:

$$F_i = V_f\beta_i + \omega_f F_i^{old} \tag{4}$$

where

$$\beta_i = \beta_i^{food} + \beta_i^{best} \tag{5}$$

and V_f is foraging velocity, ω_f is the inertia of foraging motion, and F^old_i is the final foraging movement.

In general, the random dispersal of krill individuals might be considered a random process. A maximum scattering velocity and a random directed vector can be used to define this motion. It is defined as:

$$D_i = D^{max}\delta \tag{6}$$

where D^max is the greatest scattering velocity, and δ is the stochastic vector with values from $[-1, 1]$.

The spatial vector of a krill throughout the interval t to $t + \Delta t$ is calculated based on the three above-mentioned motions and varying factors of the motion during this time, as follows:

$$X_i(t + \Delta t) = X_i(t) + \Delta t\frac{dX_i}{dt} \tag{7}$$

The KH algorithm is summarized in Algorithm 2.

The KH method has been demonstrated to be capable of effectively solving multiple numerical optimization problems [54], but it cannot avoid local optima, so it cannot be used to find a global optimum solution.

Algorithm 2 Pseudo-code for the KH Algorithm.

1: Define the bounds, parameter(s), and so forth.
2: Randomly create the starting population in search space.
3: Assess each krill based on its position.
4: Perform movement calculation.
5: Calculate movement influenced by other krill: foraging movement and physical scattering.
6: Update the krill location in the search space.
7: Go to Line 3 until the stopping criteria are reached.

3.3. Bat Algorithm (BA)

The bat algorithm was proposed by Yang in [53] and motivated by the echolocation behavior of bats. Microbat echolocation can be characterized as follows: each virtual bat flies at a random velocity v_i at position (solution) x_i with varying frequency or wavelength and loudness A_i. Its frequency, loudness, and pulse emission rate r fluctuate as it seeks for and finds its victim. A local random walk is used to enhance the search. The best ones are chosen until the specified stop requirements are reached. The dynamic behavior of a swarm of bats is controlled using a frequency-tuning approach, and the balance between exploration and exploitation can be managed by tweaking algorithm-dependent parameters in the bat algorithm.

The steps in BA are as follows:

- All bats employ echolocation to sense distance, and a bat's location x_i is regarded as a solution to a problem.
- Bats look for prey by flying randomly at location x_i with variable frequency (from the smallest frequency f_{min} to the largest value f_{max}) with varying wavelengths *lambda* and loudness A. They can automatically alter the wavelengths (or frequencies) of their emitted pulses as well as the rate of pulse emission r based on the target's distance.
- The value of loudness ranges from a large positive number A_0 to the smallest value A_{min}.

Each bat i should have its position (x_i) and velocity (v_i) defined in a d-dimensional solution space. Position x_i should be updated as the iterations progress.

$$f_k = f_{min} + (f_{max} - f_{min})rand \tag{8}$$

$$v_k^{t+1} = v_k^t + (x^* - x_k^t)f_k \tag{9}$$

$$x_k^{t+1} = x_k^t + v_K^{t+1} \tag{10}$$

where $rand \in [0, 1]$ is a stochastic vector drawn from a uniform distribution. In this case, x^* is the current global best position (solution) found after evaluating all solutions across all n bats. A new solution is produced locally for each bat using the random walk, defined as

$$X_{new} = X_{old} + \epsilon\langle A^{k+1}\rangle \tag{11}$$

where $\epsilon \in [-1, 1]$ is a stochastic number, while $\langle A_k^{t+1}\rangle$ is the mean loudness of bats at the current time step.

As iterations proceed, the loudness and rate of pulse emission are modified. As the bat comes closer to its meal, the volume decreases, and the pulse rate increases. The following equations are the formulas for loudness and puls e rate update:

$$A_k^{t+1} = \omega A_k^t \tag{12}$$

$$r^{t+1} = r_0(1 - exp(-\eta)) \tag{13}$$

where $1 < \omega < 1$ and $\eta < 1$ are constants. As $t \to \infty$, we have $A_k^t \to 0$ and $r_k^t \to r_k^0$. The initial parameters are defined as follows: $A_0 \in [1, 2]$ and emission rate $r_0 \in [0, 1]$.

The BA algorithm is summarized in Algorithm 3.

Algorithm 3 Bat algorithm.

```
Define objective function
Initialize the flock of bats
Define frequency of pulse
Set loudness and pulse rates
while t ≤ t_max do
    Adjust value of frequency
    Update velocities
    Update locations/solutions
    if (rand > r_k) then
        Select a solution from a set of best solutions
        Create a local solution nearby the selected best solution
    end if
    Create a new solution for a bat flying randomly
    if (rand < A_k & F(x_k) < F(x*)) then
        Adopt new solutions
        Increase r_k
        Decrease A_k
    end if
    Evaluate all bats and pick the current best bat
end while
```

3.4. Summary

The proposed hyper-heuristic hybridization approach, which is applied to the hybridization of the BA and KH algorithms, is visually presented in Figure 1. Whereas low-performing agents strive to copy the behavior of well-performing agents by flipping to their style of behavior, a fascinating collective behavior emerges as a result of the basic process of probabilistic state flipping. If, however, a single state applies to the bulk of agents, worse-performing agents begin to migrate to another state. Such opportunistic behavior is a trade-off between exploration and exploitation. The exploration process aids in getting near the global minimum or a local minimum, whereas the exploitation process aids in more precisely locating the global minimum. As a result, the hyper-heuristic strikes a balance between local and global qualities since the baseline algorithm may have a biased behavior towards global or local searching, thus allowing the diversification of the search to prevent becoming locked in a local optimum.

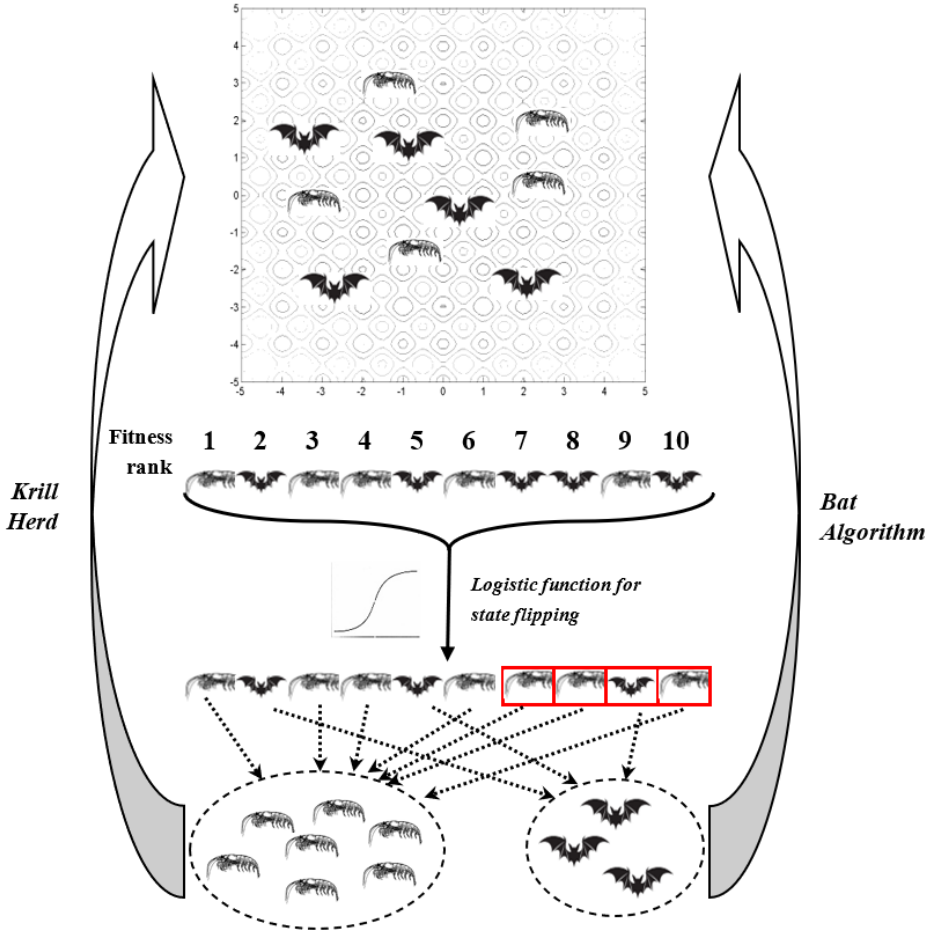


Figure 1. Abstract scheme of the introduced meta-heuristic hybridization approach.

4. Benchmarks

To experimentally validate the proposed approach, we employed 12 mathematical benchmark functions as follows: cross-in-tray (Equation (14)), rotated hyper-ellipsoid (RHE) (Equation (15)), sphere (Equation (16)), sum of different powers (Equation (17)), sum of squares (Equation (18)), McCormick (Equation (19)), Zakharov (Equation (20)), Rosenbrock (Equation (21)), De Jong function No. 5 (Equation (22)), Easom (Equation (23)), Branin (Equation (24)), and Styblinski–Tang (Equation (25)). These functions were recently used to evaluate other heuristic optimization methods [55,56].

$$f_1(x) = -0.0001(|\sin(x_1)\sin(x_2)\exp(|100 - \frac{\sqrt{x_1^2 + x_2^2}}{\pi}|)| + 1)^{0.1}, x_i \in [-10, 10], \forall i = 1, 2. \quad (14)$$

$$f_2(x) = \sum_{i=1}^{d} \sum_{j=1}^{i} x_j^2, x_i \in [-65.536, 65.536], \forall i = 1, \ldots, d. \quad (15)$$

$$f_3(x) = \sum_{i=1}^{d} x_i^2, x_i \in [-5.12, 5.12], \forall i = 1, \ldots, d. \quad (16)$$

$$f_4(x) = \sum_{i=1}^{d} |x_i|^{i+1}, x_i \in [-1, 1], \forall i = 1, \ldots, d. \quad (17)$$

$$f_5(x) = \sum_{i=1}^{d} i x_i^2, x_i \in [-5.12, 5.12], \forall i = 1, \ldots, d. \quad (18)$$

$$f_6(x) = \sin(x_1 + x_2) + (x_1 - x_2)^2 - 1.5x_1 + 2.5x_2 + 1, x_1 \in [-1.5, 4], x_2 \in [-3, 4]. \quad (19)$$

$$f_7(x) = \sum_{i=1}^{d} x_i^2 + (\sum_{i=1}^{d} 0.5ix_i)^2 + (\sum_{i=1}^{d} 0.5ix_i)^4, x_i \in [-5, 10], \forall i = 1, \ldots, d. \quad (20)$$

$$f_8(x) = \sum_{i=1}^{d-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2], x_i \in [-5, 10], \forall i = 1, \ldots, d. \quad (21)$$

$$f_9(x) = (0.002 + \sum_{i=1}^{25} \frac{1}{i + (x_1 - a_{1i})^6 + (x_2 - a_{2i})^6})^{-1}, x_i \in [-65.536, 65.536], \forall i = 1, 2. \quad (22)$$

$$f_{10}(x) = -\cos(x_i)\cos(x_2)\exp(-(x_1 - \pi)^2 - (x_2 - \pi)^2), x_i \in [-100, 100], \forall i = 1, 2. \quad (23)$$

$$f_{11}(x) = a(x_2 - bx_1^2 + cx_1 - r)^2 + s(1 - t)cos(x_1) + s, x_1 \in [-5, 10], x_2 \in [0, 15], \quad (24)$$

where $a = 1, b = 5.1/(4\pi^2), c = 5/\pi, r = 6, s = 10, t = 1/(8\pi)$.

$$f_{12}(x) = \frac{1}{2} \sum_{i=1}^{d} (x_i^4 + 16x_i^2 + 5x_i), x_i \in [-5, 5], \forall i = 1, \ldots, d, \quad (25)$$

where d denotes the dimensionality.

These functions are typically used to benchmark newly proposed evolutionary and bio-inspired optimization methods [57]. They are visualized in Figure 2.

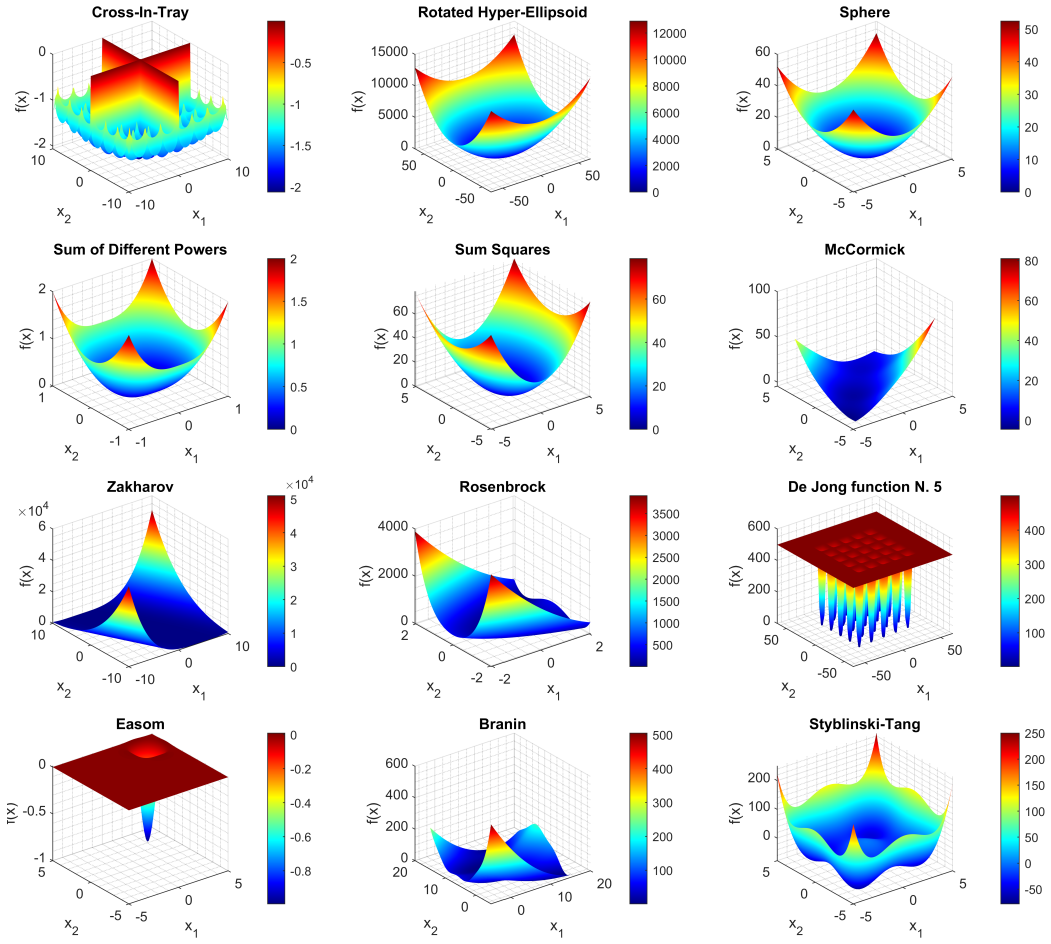


Figure 2. Visualization of optimization problems used in this study.

The benchmark functions are summarized in Table 1 according to their properties of continuity, differentiability, separability, scalability, and unimodality. A function is differentiable if its first derivative is defined at each point in its domain. A function is separable if it is defined as the sum of several functions, where at least one is a single-variable function. A function is unimodal if it only has one minimum or maximum in its landscape. A function is continuous if small changes in its output can be ensured by restricting to sufficiently small changes in its input. The hardest problems are considered to be non-differentiable, non-separable, and multi-modal.

Table 1. Benchmark functions used in this manuscript.

Function	Ref.	Continuous	Differentiable	Separable	Scalable	Unimodal
Cross-In-Tray	[58]	Yes	No	No	No	No
Rotated Hyper-Ellipsoid	[59]	Yes	Yes	Yes	Yes	Yes
Sphere	[58]	Yes	Yes	Yes	Yes	No
Sum of Different Powers	[60]	Yes	Yes	Yes	Yes	Yes
Sum of Squares	[58]	Yes	Yes	Yes	Yes	Yes
McCormick	[58]	Yes	Yes	No	No	No
Zakharov	[58]	Yes	Yes	No	Yes	No
Rosenbrock	[58]	Yes	Yes	No	Yes	Yes
De Jong No. 5	[60]	Yes	Yes	Yes	Yes	No
Easom	[58]	Yes	Yes	Yes	No	No
Branin	[60]	Yes	Yes	No	No	No
Styblinski–Tang	[58]	Yes	Yes	No	No	No

5. Results

5.1. Results with Benchmark Functions

The algorithms were written in MATLAB ver. 2020b (MathWorks, Nattick, MA) and run on a PC with a 3.9 GHz Quad Core CPU and 8 GB of RAM running Windows 10. For comparison, we developed and analyzed both the hybrid method, which merges KH and BA, and stand-alone KH and BA. We utilized 50 agents, and each algorithm was run for 100 iterations. We ran the technique 100 times each and statistically analyzed the findings.

Level curves of the 12 benchmark functions with optimal points (white dots) obtained by different simulations ($N = 50$) are presented in Figure 3.

Next, we evaluated the likelihood that the proposed strategy is the best (i.e., achieves the highest fitness) for all benchmark problems studied (Figure 4). We can observe that although the proposed strategy does not allow for high fitness in the beginning, its performance begins to improve after around 20–30 iterations.

The presented technique obtains top fitness for 11 benchmark problems, with the best performance ($p > 0.9$) obtained on RHE, sum of different powers, McCormick, and Easom functions, while it fails to attain top fitness for one (Branin) (see Figure 5).

The optimization process for 12 benchmark functions is illustrated in Figure 6, which illustrates the behaviour of the KH, BA, and hybrid KH-Bat algorithms as the process iterates.

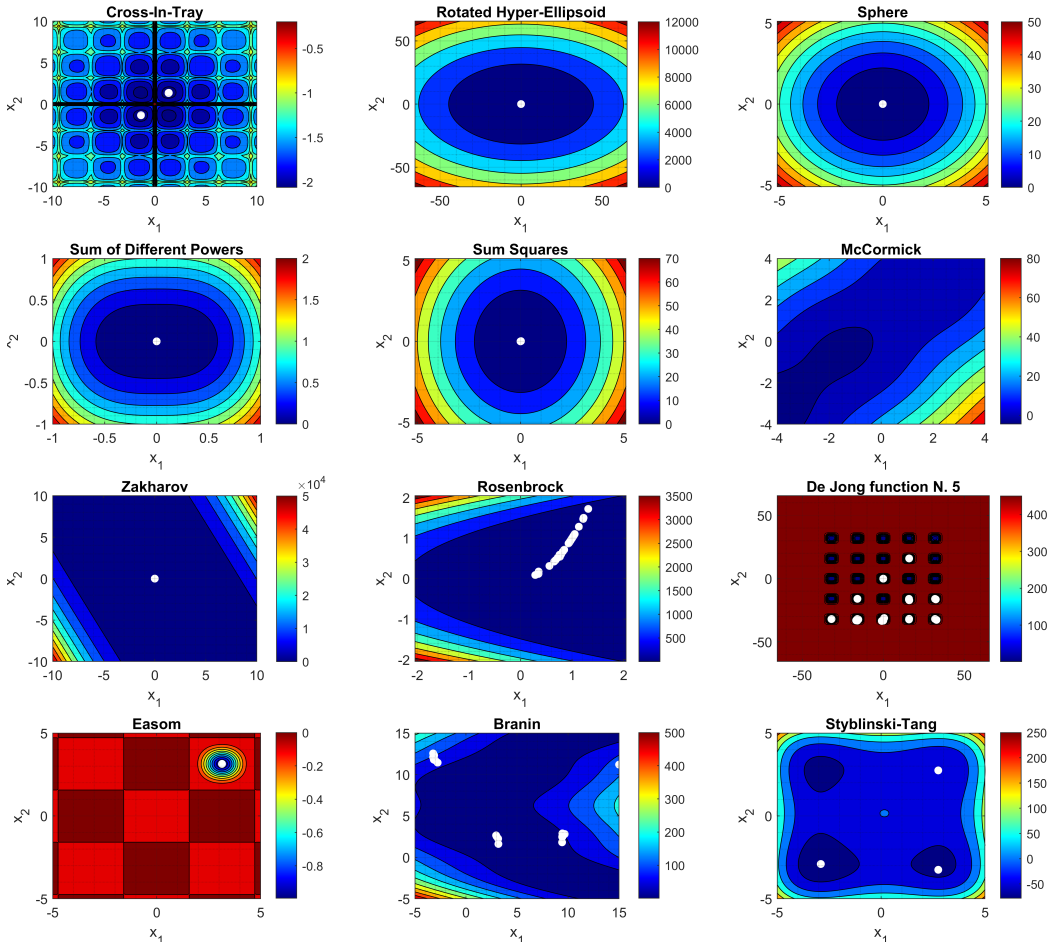


Figure 3. Benchmark function landscapes with solutions (marked by white dots) from different runs of the proposed hybrid algorithm.

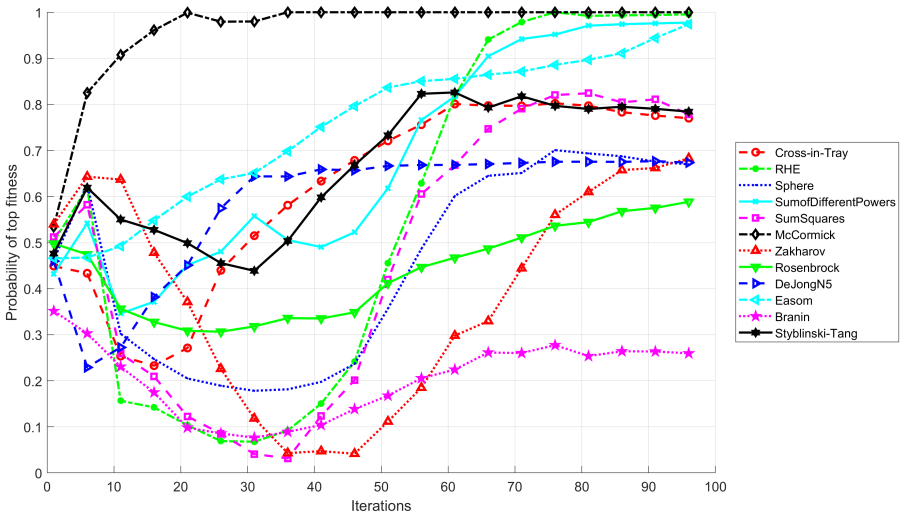


Figure 4. Probability of reaching the top fitness value for the proposed hybrid technique.

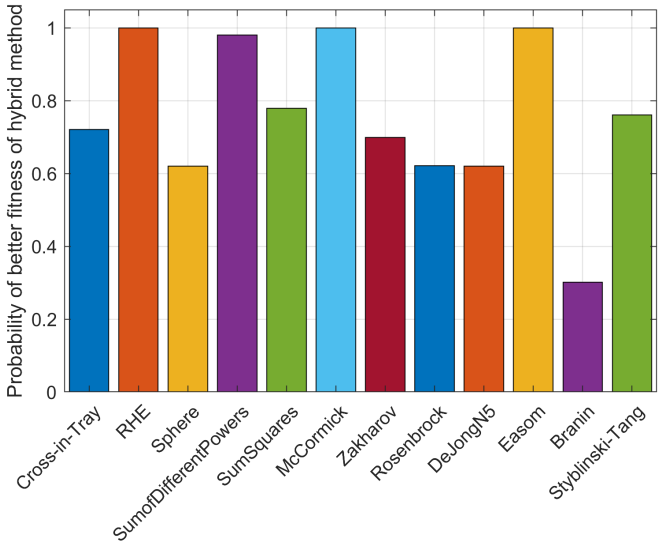


Figure 5. A plot of ranks vs. number of iterations for 12 selected optimization problems.

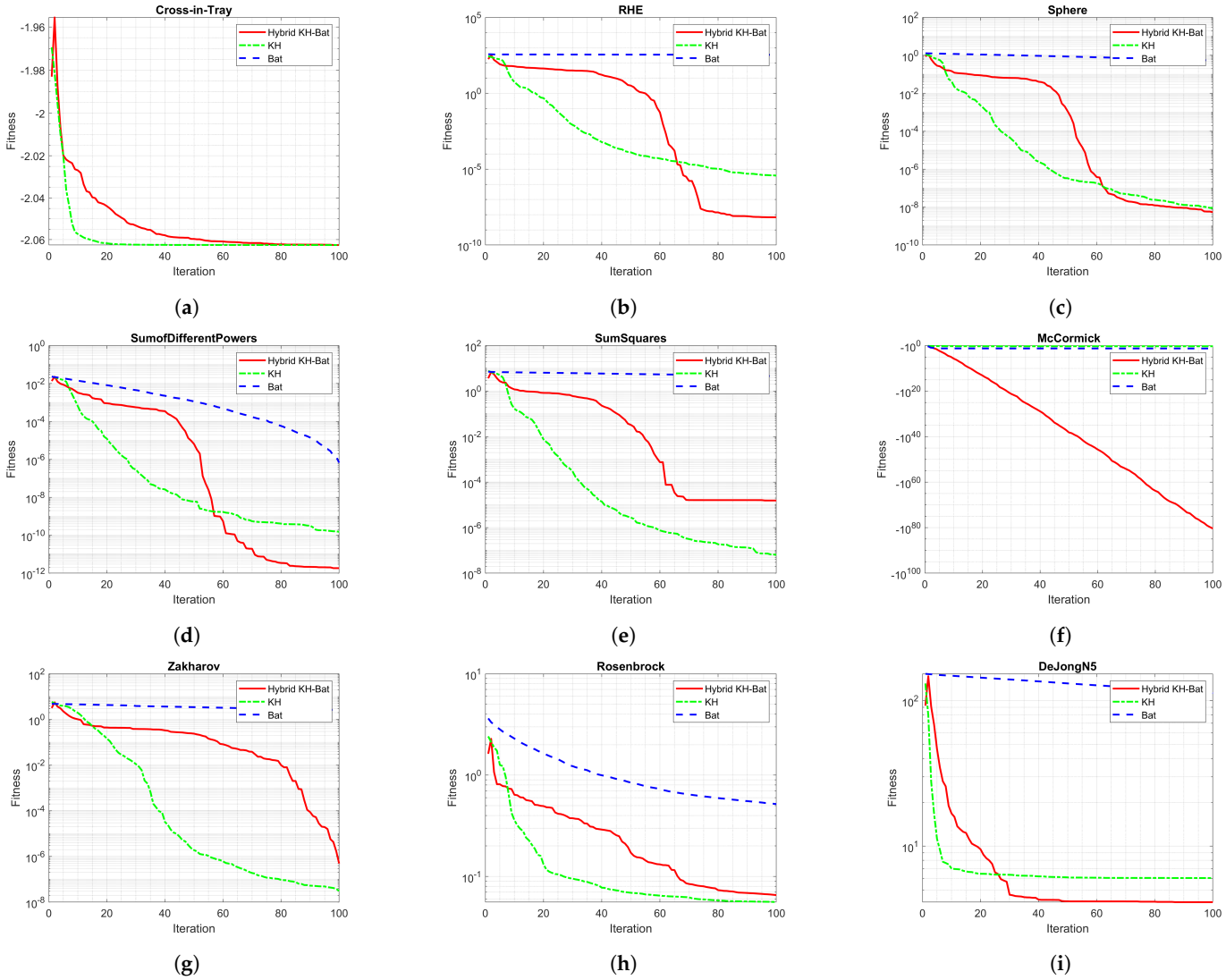
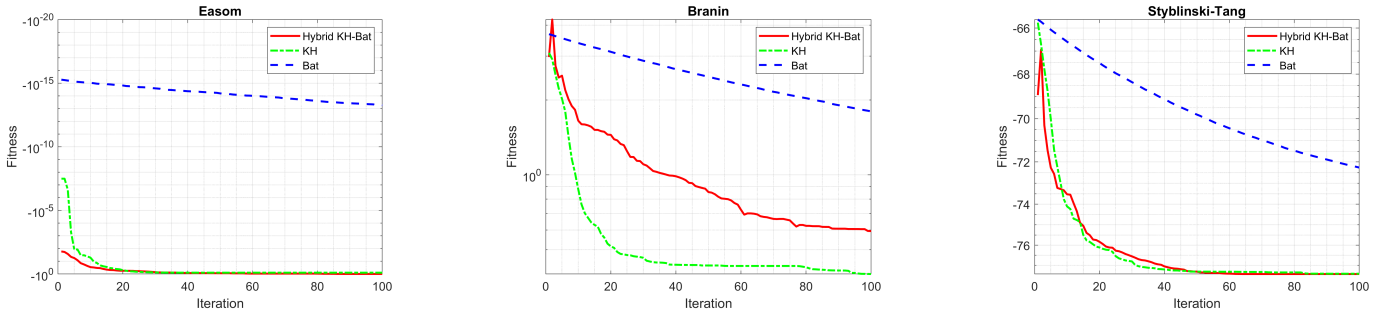


Figure 6. *Cont.*



(**j**) (**k**) (**l**)

Figure 6. Optimization process using KH, BA, and the proposed hybrid algorithm: (**a**) cross-in-tray, (**b**) rotated hyper-ellipsoid, (**c**) sphere, (**d**) sum of different powers, (**e**) sum of squares, (**f**) McCormick, (**g**) Zakharov, (**h**) Rosenbrock, (**i**) De Jong No. 5, (**j**) Easom, (**k**) Branin, (**l**) Styblinski–Tang.

To compare the hybridized algorithm with the original (BA and KH) algorithms, we performed the Friedman test and the post hoc Nemenyi test. All pairwise comparisons of algorithms were conducted using the non-parametric Nemenyi test with an α level of 0.05. The results (mean ranks) are visualized as a critical distance (CD) diagram in Figure 7.

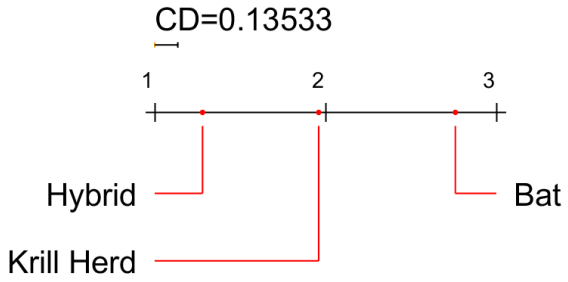


Figure 7. Critical difference (CD) diagram from the Friedman test and the post hoc Nemenyi test comparing baseline algorithms and the hybridized algorithm. High-to-low rankings are presented from left to right. A higher rank means the algorithm performed better.

The Friedman rank test was adopted to assess the algorithm's performance. The Friedman test is a powerful non-parametric statistical ranking-based test that does not require the assumption of normality. It has been previously used in numerous studies to analyze the performance of heuristic optimization algorithms [61–64]. The hypothesis of equal means, i.e., that there is no difference in ranks between the BA, KH, and hybrid BA-KH, is rejected ($p < 0.001$). The mean rankings of all compared approaches in the last (100th) iteration are shown in Figure 8, along with statistical confidence bounds. We can observe that the proposed technique is the best (mean rank 1.2791 ± 0.0195), KH is second (mean rank 1.9579 ± 0.0228), and BA is third (mean rank 2.7630 ± 0.0177).

We compared the average ranks of the algorithms under consideration in each iteration (see Figure 9). Note that from the 15th until about the 50th iteration, KH is the best strategy. The hybrid strategy then overtakes the top method in terms of rank. On the other hand, while BA is never the optimal approach, it does manage to outperform KH when the methods are performed using a greater number of iterations.

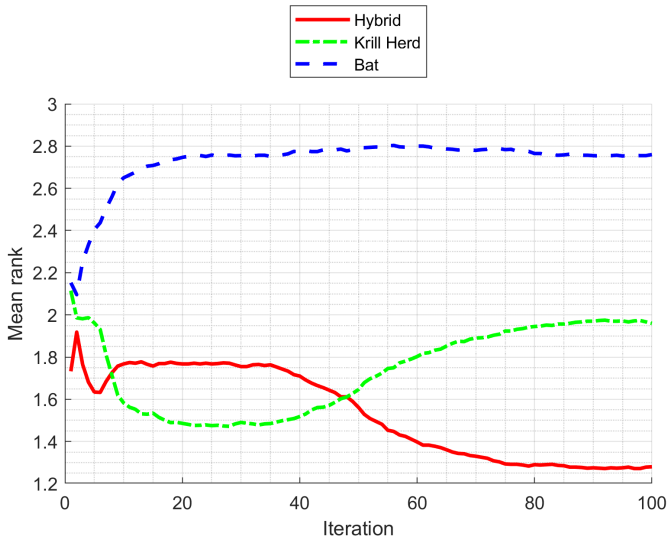


Figure 8. Mean ranks from the Friedman test during optimization.

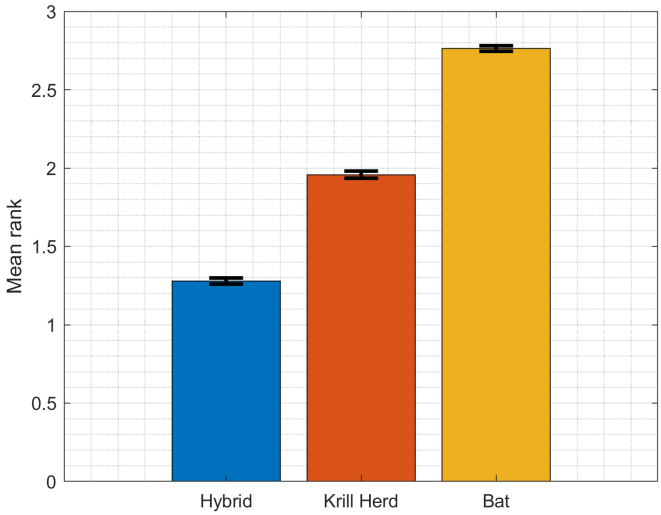


Figure 9. Average ranks for 12 optimization functions in the final iteration. A smaller rank is better.

For a more balanced evaluation, we adopted a different criterion to terminate iterations. We used the mean change in the coordinates of a swarm of krill or bats in subsequent iterations as a criterion. The process of optimization terminates if $|X_{curr} - X_{prev}|^2 < eps$, where $X_{curr}, X_{prev} \in (x_1, x_2, \ldots, x_D)$ are the function arguments (i.e., coordinates), D is the dimensionality of the problem, and eps is a small number. We repeated the process for 50 runs. To statistically evaluate the minimum value returned by the algorithms across 12 optimization problems, we also performed the Friedman test and the post hoc Nemenyi test (at an α level of 0.05). The results (mean ranks) are visualized as a critical distance (CD) diagram in Figure 10. These results show the statistically significant superiority of the hybridized algorithm.

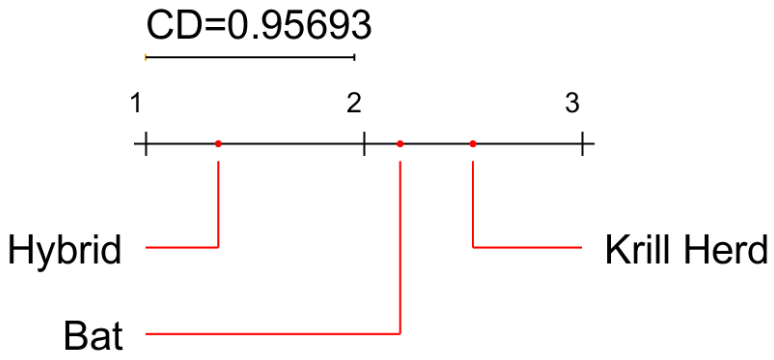


Figure 10. Critical difference (CD) diagram comparing baseline algorithms and hybridized algorithm using an alternative stopping criterion. High-to-low rankings are presented from left to right. A higher rank means that the algorithm performed better.

Finally, we analyzed the time performance for all 12 optimization functions. The results are presented in Figure 11. Since the hybrid algorithm uses the iterations from both KH and BA algorithms, its performance is better than the algorithm with the worst performance (KH) but much worse than the performance of the algorithm with the best performance (BA).

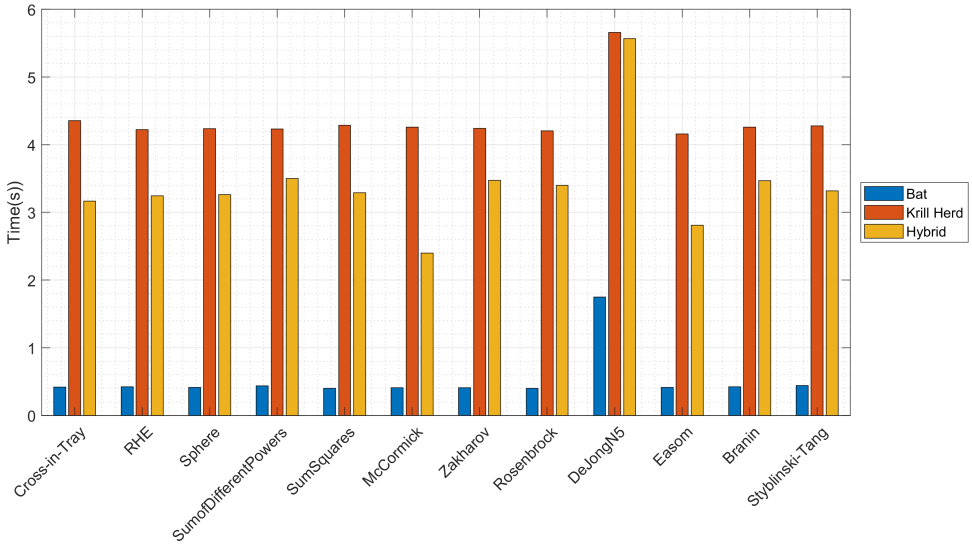


Figure 11. Time performance comparison for 12 optimization functions.

5.2. Three-Bar Truss Design Problem

The three-bar truss design is a well-known optimization problem in the realm of civil engineering (see Figure 12). The problem is very popular and has been used to benchmark optimization methods in numerous papers, most recently in [65–68]. The main goal of this problem is to decrease the weight of a three-bar truss by taking two structural characteristics into account. The three major constraints are deflection, stress, and buckling. This problem is defined as follows.

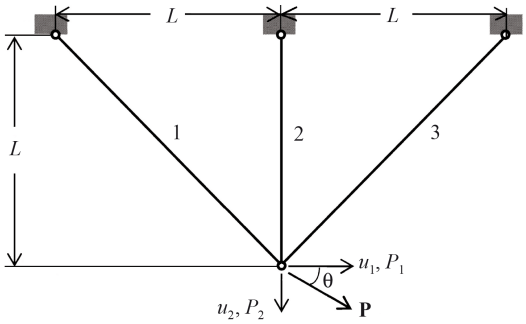


Figure 12. Schematic representation of the three-bar truss problem.

The three truss nodes are fully secured. The fourth node is free and has 2 degrees of freedom of movement u_1, u_2, and a force of magnitude P is applied to it at angle θ to the horizon. The angle is counted clockwise and can be changed in the range from 0 to 360°; L is the farm size; x_1, x_2, and x_3 are cross-sectional areas of rods; ρ, E, and d are density, the modulus of longitudinal elasticity, and the permissible tensile and compressive stresses of the truss material, respectively. The purpose of the design is to select the cross-sectional areas of the rods x_1, x_2, and x_3 (controlled parameters, design variables) such that the truss has the minimum mass and that stress and area constraints are met for the cross-sections of rods. Consider:

$$\overrightarrow{x} = [x_1 x_2] = [A_1 A_2] \tag{26}$$

Minimize:

$$f(\overrightarrow{x}) = (2\sqrt{2}x_1 + x_2) \times l \tag{27}$$

Subject to:

$$g_1(\overrightarrow{x}) = \frac{\sqrt{2}x_1 + x_2}{\sqrt{2}x_1^2 + 2x_1x_2} P - \sigma \leq 0 \tag{28}$$

$$g_2(\overrightarrow{x}) = \frac{x_2}{\sqrt{2}x_1^2 + 2x_1x_2} P - \sigma \leq 0 \tag{29}$$

$$g_3(\overrightarrow{x}) = \frac{1}{\sqrt{2}x_2 + x_1} P - \sigma \leq 0 \tag{30}$$

Variable range:

$$0 \leq x_1, x_2 \leq 1 \tag{31}$$

where $l = 100$ cm, $P = 2$ KN/cm^2, and $\sigma = 2$ KN/cm^2.

The results are presented in Table 2, which also compares the proposed approach with other methods. Table 2 was adopted from [69] and updated with the proposed hybrid algorithm results. The algorithms used for comparison are as follows: dingo optimization algorithm (DOA), mine blast algorithm (MBA), salp swarm algorithm (SSA), PSO with differential evolution (PSO-DE), and DE with dynamic stochastic selection (DEDS).

Table 2. Comparison of performance on the three-bar truss problem.

Algorithm	x_1	x_2	Optimum Weight
DOA	0.788675095	0.40824840	263.8958434
MBA	0.78856500	0.40855970	263.8958522
SSA	0.78866541	0.40827578	263.8958434
PSO-DE	0.78867510	0.40824820	263.8958433
DEDS	0.78867513	0.40824828	263.8958434
Proposed	0.78853476	0.40866456	263.8958434

6. Discussion and Conclusions

We investigated the proposed model and discovered its suitability for developing meta-heuristics to address mathematical problems of varying dimensions and features.

In the process of development, hybrid heuristic algorithms can overcome the major drawbacks of conventional optimization algorithms, such as becoming trapped in local optima and performance degradation in practical applications on complex problems. Combining two algorithms to generate a new one is a basic hybridization procedure. The purpose of this study was to provide a method for automatically constructing heuristic algorithms by combining the strengths and compensating for the shortcomings of existing heuristics. This method is known as hyper-heuristic, and it allows automation of the selection, combination, or modification of a large number of simpler heuristics to address complex computational and design engineering optimization issues.

We present a hybridization scheme of bio-inspired algorithms based on state flipping that sets each agent to a changeable state. The state is switched at random based on each agent's fitness rating following its current iteration. The proposed strategy seeks to combine the benefits of the krill herd (KH) and the bat algorithm (BA) in order to prevent all agents from becoming stuck in local optimum zones.

On numerous optimization functions with multiple local minima, the proposed approach is more successful at generating better solutions more frequently (i.e., achieves a higher rank and has a larger likelihood of reaching the top rank compared to to the stand-alone KH and BA).

We plan to carry out the following activities in the future:

- Assess the proposed scheme's efficiency, stability, and significance using other known unconstrained benchmark functions and several real-life problems.
- Hybridize the GA, PSO, GSA, or ACO algorithms and compare the hybrid method to these algorithms as baselines.
- Compare the proposed algorithm's resilience and efficiency to several state-of-the-art optimization algorithms.
- Apply the proposed hybrid approach to real-life applications, such as image segmentation, clustering, and feature selection, based on its promising results in finding the best solution for the challenges that we investigated. This discovery is also being looked at as a potential new research avenue for using meta-heuristic algorithms to handle issues such as image processing and segmentation, feature selection, and industrial process parameter estimation.

Author Contributions: Conceptualization, R.D.; methodology, R.D.; software, R.D.; validation, R.D. and R.M.; formal analysis, R.D. and R.M.; investigation, R.D. and R.M.; writing—original draft preparation, R.D.; writing—review and editing, R.D. and R.M.; visualization, R.D.; funding acquisition, R.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The implementation of the Krill Herd algorithm used in this study is available at https://se.mathworks.com/matlabcentral/fileexchange/55486-krill-herd-algorithm, accessed on 1 November 2021. The implementation of Bat Algorithm used in this study is available at: https://se.mathworks.com/matlabcentral/fileexchange/37582-bat-algorithm-demo, accessed on 1 November 2021.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Hanif, M.K.; Talib, R.; Awais, M.; Saeed, M.Y.; Sarwar, U. Comparison of bioinspired computation and optimization techniques. *Curr. Sci.* **2018**, *115*, 450–453. [CrossRef]
2. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey Wolf Optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
3. Karaboga, D.; Basturk, B. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *J. Glob. Optim.* **2007**, *39*, 459–471. [CrossRef]
4. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95—International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995. [CrossRef]
5. Fister Jr., I.; Yang, X..; Brest, J. A comprehensive review of firefly algorithms. *Swarm Evol. Comput.* **2013**, *13*, 34–46. [CrossRef]

6. Gandomi, A.H.; Alavi, A.H. Krill herd: A new bio-inspired optimization algorithm. *Commun. Nonlinear Sci. Numer. Simul.* **2012**, *17*, 4831–4845. [CrossRef]
7. Polap, D.; Woźniak, M. Polar bear optimization algorithm: Meta-heuristic with fast population movement and dynamic birth and death mechanism. *Symmetry* **2017**, *9*, 203. [CrossRef]
8. Połap, D.; Woźniak, M. Red fox optimization algorithm. *Expert Syst. Appl.* **2021**, *166*, 114107. [CrossRef]
9. Parpinelli, R.S.; Lopes, H.S. An eco-inspired evolutionary algorithm applied to numerical optimization. In Proceedings of the 2011 Third World Congress on Nature and Biologically Inspired Computing, Salamanca, Spain, 19–21 October 2011. [CrossRef]
10. Połap, D.; Woźniak, M.; Napoli, C.; Tramontana, E.; Damaševičius, R. Is the Colony of Ants Able to Recognize Graphic Objects? In *Communications in Computer and Information Science*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 376–387. [CrossRef]
11. Khurma, R.A.; Alsawalqah, H.; Aljarah, I.; Elaziz, M.A.; Damaševičius, R. An enhanced evolutionary software defect prediction method using island moth flame optimization. *Mathematics* **2021**, *9*, 1722. [CrossRef]
12. Woźniak, M.; Książek, K.; Marciniec, J.; Połap, D. Heat production optimization using bio-inspired algorithms. *Eng. Appl. Artif. Intell.* **2018**, *76*, 185–201. [CrossRef]
13. Khan, M.A.; Sharif, M.; Akram, T.; Damaševičius, R.; Maskeliūnas, R. Skin lesion segmentation and multiclass classification using deep learning features and improved moth flame optimization. *Diagnostics* **2021**, *11*, 811. [CrossRef] [PubMed]
14. Kadry, S.; Rajinikanth, V.; Damasevicius, R.; Taniar, D. Retinal Vessel Segmentation with Slime-Mould-Optimization based Multi-Scale-Matched-Filter. In Proceedings of the 2021 IEEE 7th International Conference on Bio Signals, Images and Instrumentation, Chennai, India, 25–27 March 2021.
15. Wolpert, D.H.; Macready, W.G. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1997**, *1*, 67–82. [CrossRef]
16. Ting, T.O.; Yang, X.S.; Cheng, S.; Huang, K. Hybrid Metaheuristic Algorithms: Past, Present, and Future. In *Studies in Computational Intelligence*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 71–83. [CrossRef]
17. Özcan, E.; Bilgin, B.; Korkmaz, E.E. A comprehensive analysis of hyper-heuristics. *Intell. Data Anal.* **2008**, *12*, 3–23. [CrossRef]
18. Burke, E.; Kendall, G.; Newall, J.; Hart, E.; Ross, P.; Schulenburg, S. Hyper-Heuristics: An Emerging Direction in Modern Search Technology. In *Handbook of Metaheuristics*; Kluwer Academic Publishers: Boston, MA, USA, 2003, pp. 457–474. [CrossRef]
19. Chakhlevitch, K.; Cowling, P. Hyperheuristics: Recent Developments. In *Studies in Computational Intelligence*; Springer: Berlin/Heidelberg, Germany, 2008; Volume 136, pp. 3–29.
20. Özcan, E.; Kheiri, A. A hyper-heuristic based on random gradient, greedy and dominance. In Proceedings of the Computer and Information Sciences II—26th International Symposium on Computer and Information Sciences, 2012; pp. 557–563.
21. Cowling, P.; Kendall, G.; Han, L. An investigation of a hyperheuristic genetic algorithm applied to a trainer scheduling problem. In Proceedings of the 2002 Congress on Evolutionary Computation, 2002; Volume 2, pp. 1185–1190.
22. Asta, S.; Özcan, E. A tensor-based selection hyper-heuristic for cross-domain heuristic search. *Inf. Sci.* **2015**, *299*, 412–432. [CrossRef]
23. Sun, Y.; Zhang, L.; Gu, X. A hybrid co-evolutionary cultural algorithm based on particle swarm optimization for solving global optimization problems. *Neurocomputing* **2012**, *98*, 76–89. [CrossRef]
24. Pan, Q. An effective co-evolutionary artificial bee colony algorithm for steelmaking-continuous casting scheduling. *Eur. J. Oper. Res.* **2016**, *250*, 702–714. [CrossRef]
25. Dokeroglu, T.; Cosar, A. A novel multistart hyper-heuristic algorithm on the grid for the quadratic assignment problem. *Eng. Appl. Artif. Intell.* **2016**, *52*, 10–25. [CrossRef]
26. Damaševičius, R.; Woźniak, M. State flipping based hyper-heuristic for hybridization of nature inspired algorithms. In Proceedings of the 16th International Conference, ICAISC 2017, Zakopane, Poland, 11–15 June 2017; Lecture Notes in Computer Science; 2017; Volume 10245 LNAI, pp. 337–346.
27. Gandomi, A.H.; Yang, X.S.; Talatahari, S.; Alavi, A.H. Metaheuristic Algorithms in Modeling and Optimization. *Metaheuristic Appl. Struct. Infrastructures* **2013**, 1–24. [CrossRef]
28. Molina, D.; Poyatos, J.; Ser, J.D.; García, S.; Hussain, A.; Herrera, F. Comprehensive Taxonomies of Nature- and Bio-inspired Optimization: Inspiration Versus Algorithmic Behavior, Critical Analysis Recommendations. *Cogn. Comput.* **2020**, *12*, 897–939. 2020. [CrossRef]
29. Raidl, G.R. *A Unified View on Hybrid Metaheuristics*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 1–12. [CrossRef]
30. Wang, X. Hybrid Nature-Inspired Computation Methods for Optimization. Ph.D. Thesis, Helsinki University of Technology, Espoo, Finland, 2009.
31. Stegherr, H.; Heider, M.; Hähner, J. Classifying Metaheuristics: Towards a unified multi-level classification system. *Nat. Comput.* **2020**, 1–17. [CrossRef]
32. Abd-Alsabour, N. *Hybrid Metaheuristics for Classification Problems*. In *Pattern Recognition—Analysis and Applications*; InTech: Rijeka, Croatia, 2016. [CrossRef]
33. Eiben, A.E.; Michalewicz, Z.; Schoenauer, M.; Smith, J.E. *Parameter Control in Evolutionary Algorithms*; Springer: Berlin/Heidelberg, Germany, pp. 19–46. [CrossRef]
34. Dioşan, L.; Oltean, M. Evolutionary design of Evolutionary Algorithms. *Genet. Program. Evolvable Mach.* **2009**, *10*, 263–306. [CrossRef]
35. Harding, S.; Miller, J.F.; Banzhaf, W. Developments in Cartesian Genetic Programming: Self-modifying CGP. *Genet. Program. Evolvable Mach.* **2010**, *11*, 397–439. [CrossRef]

36. Grobler, J.; Engelbrecht, A.P.; Kendall, G.; Yadavalli, V.S.S. Heuristic space diversity control for improved meta-hyper-heuristic performance. *Inf. Sci.* **2015**, *300*, 49–62. [CrossRef]
37. Alrassas, A.M.; Al-Qaness, M.A.A.; Ewees, A.A.; Ren, S.; Elaziz, M.A.; Damaševičius, R.; Krilavičius, T. Optimized anfis model using aquila optimizer for oil production forecasting. *Processes* **2021**, *9*, 1194. [CrossRef]
38. Helmi, A.M.; Al-Qaness, M.A.A.; Dahou, A.; Damaševičius, R.; Krilavičius, T.; Elaziz, M.A. A novel hybrid gradient-based optimizer and grey wolf optimizer feature selection method for human activity recognition using smartphone sensors. *Entropy* **2021**, *23*, 1065. [CrossRef] [PubMed]
39. Jouhari, H.; Lei, D.; Al-qaness, M.A.A.; Abd Elaziz, M.; Damaševičius, R.; Korytkowski, M.; Ewees, A.A. Modified Harris Hawks optimizer for solving machine scheduling problems. *Symmetry* **2020**, *12*, 1460. [CrossRef]
40. Makhadmeh, S.N.; Al-Betar, M.A.; Alyasseri, Z.A.A.; Abasi, A.K.; Khader, A.T.; Damaševičius, R.; Mohammed, M.A.; Abdulkareem, K.H. Smart home battery for the multi-objective power scheduling problem in a smart home using grey wolf optimizer. *Electronics* **2021**, *10*, 447. [CrossRef]
41. Neggaz, N.; Ewees, A.A.; Elaziz, M.A.; Mafarja, M. Boosting salp swarm algorithm by sine cosine algorithm and disrupt operator for feature selection. *Expert Syst. Appl.s* **2020**, *145*, 113103. [CrossRef]
42. Ksiazek, K.; Połap, D.; Woźniak, M.; Damaševičius, R. Radiation heat transfer optimization by the use of modified ant lion optimizer. In Proceedings of the 2017 IEEE Symposium Series on Computational Intelligence, Honolulu, HI, USA, 27 November–1 December 2018; pp. 1–7.
43. Cruz-Duarte, J.M.; Amaya, I.; Ortiz-Bayliss, J.C.; Conant-Pablos, S.E.; Terashima-Marín, H.; Shi, Y. Hyper-Heuristics to customise metaheuristics for continuous optimisation. *Swarm Evol. Comput.* **2021**, *66*, 100935. [CrossRef]
44. Turky, A.; Sabar, N.R.; Dunstall, S.; Song, A. Hyper-heuristic local search for combinatorial optimisation problems. *Knowl.-Based Syst.* **2020**, *205*, 106264. [CrossRef]
45. Wang, S.; Jia, H.; Abualigah, L.; Liu, Q.; Zheng, R. An improved hybrid aquila optimizer and harris hawks algorithm for solving industrial engineering optimization problems. *Processes* **2021**, *9*, 1551. [CrossRef]
46. Abd Elaziz, M.; Ewees, A.A.; Neggaz, N.; Ibrahim, R.A.; Al-qaness, M.A.A.; Lu, S. Cooperative meta-heuristic algorithms for global optimization problems. *Expert Syst. Appl.* **2021**, *176*, 114788. [CrossRef]
47. Dabba, A.; Tari, A.; Meftali, S. Hybridization of Moth flame optimization algorithm and quantum computing for gene selection in microarray data. *J. Ambient. Intell. Humaniz. Comput.* **2021**, *12*, 2731–2750. [CrossRef]
48. Huo, L.; Zhu, J.; Li, Z.; Ma, M. A hybrid differential symbiotic organisms search algorithm for UAV path planning. *Sensors* **2021**, *21*, 3037. [CrossRef] [PubMed]
49. Shehadeh, H.A. A hybrid sperm swarm optimization and gravitational search algorithm (HSSOGSA) for global optimization. *Neural Comput. Appl.* **2021**, *33*, 11739–11752. [CrossRef]
50. Kundu, T.; Garg, H. A hybrid ITLHHO algorithm for numerical and engineering optimization problems. *Int. J. Intell. Syst.* **2021**. [CrossRef]
51. Chiu, P.C.; Selamat, A.; Krejcar, O.; Kuok, K.K. Hybrid Sine Cosine and Fitness Dependent Optimizer for Global Optimization. *IEEE Access* **2021**, *9*, 128601–128622. [CrossRef]
52. Alkhateeb, F.; Abed-alguni, B.H.; Al-rousan, M.H. Discrete hybrid cuckoo search and simulated annealing algorithm for solving the job shop scheduling problem. *J. Supercomput.* **2021**, 1–28. [CrossRef]
53. Yang, X.S. *A New Metaheuristic Bat-Inspired Algorithm*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 65–74. [CrossRef]
54. Wang, G.G.; Gandomi, A.H.; Alavi, A.H.; Gong, D. A comprehensive review of krill herd algorithm: Variants, hybrids and applications. *Artif. Intell. Rev.* **2017**. *51*, 119–148. [CrossRef]
55. Kumar, A.; Misra, R.K.; Singh, D.; Mishra, S.; Das, S. The spherical search algorithm for bound-constrained global optimization problems. *Appl. Soft Comput. J.* **2019**, *85*, 105734. [CrossRef]
56. Liang, Y..; Cuevas Juarez, J.R. A self-adaptive virus optimization algorithm for continuous optimization problems. *Soft Comput.* **2020**, *24*, 13147–13166. [CrossRef]
57. Li, X.; Tang, K.; Omidvar, M.N.; Yang, Z.; Qin, K. Benchmark Functions for the CEC'2013 Special Session and Competition on Large-Scale Global Optimization. *Gene* **2013**, *7*, 8.
58. Jamil, M.; Yang, X.S. A literature survey of benchmark functions for global optimisation problems. *Int. J. Math. Model. Numer. Optim.* **2013**. *4*, 150. [CrossRef]
59. Al-Roomi, A.R. Unconstrained Single-Objective Benchmark Functions Repository. 2015. Available online: https://www.al-roomi.org/benchmarks/unconstrained (accessed on 9 December 2021).
60. Surjanovic, S.; Bingham, D. Virtual Library of Simulation Experiments: Test Functions and Datasets. Available online: http://www.sfu.ca/~ssurjano (accessed on 13 November 2021).
61. Chen, H.; Heidari, A.A.; Zhao, X.; Zhang, L.; Chen, H. Advanced orthogonal learning-driven multi-swarm sine cosine optimization: Framework and case studies. *Expert Syst. Appl.* **2020**, *144*, 113113. [CrossRef]
62. Chen, H.; Heidari, A.A.; Chen, H.; Wang, M.; Pan, Z.; Gandomi, A.H. Multi-population differential evolution-assisted Harris hawks optimization: Framework and case studies. *Future Gener. Comput. Syst.* **2020**, *111*, 175–198. [CrossRef]
63. Tawhid, M.A.; Ibrahim, A.M. A hybridization of grey wolf optimizer and differential evolution for solving nonlinear systems. *Evol. Syst.* **2020**, *11*, 65–87. [CrossRef]

64. Chakraborty, S.; Sharma, S.; Saha, A.K.; Chakraborty, S. SHADE–WOA: A metaheuristic algorithm for global optimization. *Appl. Soft Comput.* **2021**, *113*, 107866. [CrossRef]
65. Jan, M.A.; Mahmood, Y.; Khan, H.U.; Mashwani, W.K.; Uddin, M.I.; Mahmoud, M.; Khanum, R.A.; Ikramullah.; Mast, N. Feasibility-Guided Constraint-Handling Techniques for Engineering Optimization Problems. *Comput. Mater. Contin.* **2021**, *67*, 2845–2862.
66. Sattar, D.; Salim, R. A smart metaheuristic algorithm for solving engineering problems. *Eng. Comput.* **2021**, *37*, 2389–2417. [CrossRef]
67. Yildiz, B.S.; Pholdee, N.; Bureerat, S.; Yildiz, A.R.; Sait, S.M. Enhanced grasshopper optimization algorithm using elite opposition-based learning for solving real-world engineering problems. *Eng. Comput.* **2021**, 1–13. [CrossRef]
68. Massoudi, M.S.; Sarjamei, S.; Esfandi Sarafraz, M. Smell Bees Optimization algorithm for continuous engineering problem. *Asian J. Civ. Eng.* **2020**, *21*, 925–946. [CrossRef]
69. Peraza-Vázquez, H.; Peña-Delgado, A.F.; Echavarría-Castillo, G.; Morales-Cepeda, A.B.; Velasco-Álvarez, J.; Ruiz-Perez, F. A Bio-Inspired Method for Engineering Design Optimization Inspired by Dingoes Hunting Strategies. *Math. Probl. Eng.* **2021**. *2021*, 9107547. [CrossRef]