computers

MDPI

Article

"Optimizing the Optimization": A Hybrid Evolutionary-Based AI Scheme for Optimal Performance

Agathoklis A. Krimpenis [1,*] and Loukas Athanasakos [2]

[1] Department of Mechanical Engineering, Hellenic Mediterranean University, Estavromenos, 71410 Heraklion, Greece
[2] Core Department, National and Kapodistrian University of Athens, 34400 Psachna, Greece; lathanasakos@core.uoa.gr
* Correspondence: akrimpenis@hmu.gr

Abstract: Optimization algorithms for solving technological and scientific problems often face long convergence times and high computational costs due to numerous input/output parameters and complex calculations. This study focuses on proposing a method for minimizing response times for such algorithms across various scientific fields, including the design and manufacturing of high-performance, high-quality components. It introduces an innovative mixed-scheme optimization algorithm aimed at effective optimization with minimal objective function evaluations. Indicative key optimization algorithms—namely, the Genetic Algorithm, Firefly Algorithm, Harmony Search Algorithm, and Black Hole Algorithm—were analyzed as paradigms to standardize parameters for integration into the mixed scheme. The proposed scheme designates one algorithm as a "leader" to initiate optimization, guiding others in iterative evaluations and enforcing intermediate solution exchanges. This collaborative process seeks to achieve optimal solutions at reduced convergence costs. This mixed scheme was tested on challenging benchmark functions, demonstrating convergence speeds that were at least three times faster than the best-performing standalone algorithms while maintaining solution quality. These results highlight its potential as an efficient optimization approach for computationally intensive problems, regardless of the included algorithms and their standalone performance.

Keywords: metaheuristic algorithms; normalization; stochastic approach; distributed optimization; exploration; exploitation

check for updates

Academic Editor: Paolo Bellavista, Behrouz Zolfaghari, Amin Ramezani, Mohsen Hadian and Firooz Saghezchi

Received: 24 January 2025
Revised: 24 February 2025
Accepted: 6 March 2025
Published: 8 March 2025

Citation: Krimpenis, A.A.; Athanasakos, L. "Optimizing the Optimization": A Hybrid Evolutionary-Based AI Scheme for Optimal Performance. *Computers* **2025**, *14*, 97. https://doi.org/10.3390/computers14030097

Copyright: © 2025 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

Optimization is defined as the selection of the best possible solution based on predetermined criteria out of a set of available alternative solutions [1]. Optimization problems arise in all fields that require quantitative data management, predominantly in computer science, engineering [2], operational research, and economics. The development of optimization methods has been of interest in mathematics for centuries [3], and research interest in this area remains undiminished. In the simplest case, an optimization problem involves maximizing or minimizing a real-valued function by systematically selecting input values from a permissible set and calculating the output values of the function, attempting to approach the ideal solution. Generally, optimization entails finding the "best available" values of some objective function given a specified range of values (or inputs) [4]. The best values of a function may consist of a multitude of local optima, with the goal of the optimization process being to find the global optimum extremum. To find ideal minima or maxima in an optimization problem, heuristic methods such as Genetic Algorithms, the

Firefly Algorithm, and many other related evolutionary-based optimization algorithms are used. In most cases with optimization problems, the objective is to minimize or maximize scoring functions.

Many automated software tools for optimization generally apply either a maximization or a minimization task, not both [5]. Therefore, a maximization problem can be converted into a minimization one (and vice versa) by adding a negative sign in the objective function's formula. Since evolutionary-based algorithms operate stochastically, especially in multi-objective optimization schemes, finding optimum objective function values is expressed using the Pareto front theory. The Pareto principle (also known as the 80/20 rule) states that approximately 80% of the effects come from 20% of the causes [6]. Based on this, the Pareto front is a set of non-dominated solutions, chosen as optimal if no goal can be improved without sacrificing at least one other goal [7]. In an optimization problem, the entire solution space is simultaneously the output value domain of the function. In constrained optimization, only sub-sets of the output value space constitute the solution space. Additionally, the imposed constraints could be either expressed as equalities or inequalities.

Recently, significant interest has focused on finding solutions for a wide range of challenging optimization problems by scientists and researchers from various fields. This interest extends beyond academic and research purposes to address the expanding needs in real-life applications. The most widespread practice involves recommending algorithms based on elements observed in nature. This is due to the remarkable similarity between the way natural laws (such as natural phenomena) interact and the behavior of living entities (such as genes or a swarm of insects) in problem-solving. A comparison is made with how the human mind tackles similar problems.

Algorithms for global optimization are usually divided into two major categories: deterministic and stochastic [8]. In deterministic global optimization, the solution search space is the complete domain of all independent or input variables. At each step, the value range of one of the variables is chosen. Subsequently, the lower and upper bounds of the objective function values are calculated with regard to the current region. If these bounds are within a given distance from each other, the upper limit is accepted as the best optimum in the current region. If it is better than the previous overall best optimum, the current replaces the previous; otherwise, the region is discarded. If the bounds are farther apart, a variable for branching and a branching point are selected, and the region is divided into smaller sub-regions. The algorithm terminates when there are no more regions to examine. In the stochastic approach to the global optimization of a function, two phases are considered: global and local. In the 'global phase', the function is evaluated at a number of random points generated from a uniform distribution. In the 'local phase', the corresponding points are used as starting points for a local minimum search. The effectiveness of methods utilizing multiple stages depends on both the performance of the overall stochastic phase and the corresponding local minimization phase.

As the state-of-the-art progresses, and with the current popularity of Artificial Intelligence, the main scientific application fields that employ optimization algorithms are engineering [9] (e.g., optimizing aerodynamic shapes, designing lightweight yet robust structures); finance [10] (e.g., balancing risk and return or calibrating trading algorithms); business development [11] (e.g., optimizing customer clusters for personalized marketing); molecular modeling [12] (e.g., optimizing materials for catalytic reactions or simulating protein structures); biology [13] (e.g., designing synthetic gene circuits or accurately assembling fragmented DNA sequences into complete genomes); and machine learning [14] (e.g., optimizing communication and model aggregation for federated learning or optimizing neural network architecture).

Motivation

In metaheuristic algorithms, the term 'meta' characterizes algorithms that operate at a higher level than heuristics, generally performing better than simple heuristic schemes. All metaheuristic algorithms compensate for the focus on local searches of solutions by exploring the entire solution space [15]. Solution diversity is often achieved through randomization. Randomization is an effective way to limit local search compared to searching the entire solution space. This minimizes the chances of the algorithm getting trapped in local minima. Almost all metaheuristic algorithms are usually suitable for nonlinear modeling and global optimization. The primary components of any metaheuristic algorithm are intensification (exploitation) and diversification (exploration). Diversification means searching for different solutions to the extent that allows for exploration of as much as possible of the available value range, while intensification means focusing on nearby available values, utilizing the information that a good solution is in that region. Selection of the best ensures that the solutions eventually converge to the optimal one. These metaheuristic algorithms can also be used in hybrid schemes [16], commonly referred to as hyper-heuristics [17], where a combination of heuristic and metaheuristic algorithms can be used in tandem. These hybrid schemes, although generally more flexible and adaptable to problem-solving, still rely on selecting one algorithm that ultimately solves the optimization problem [18]. In this article, a thorough build and evaluation of a complex algorithmic scheme for the optimization of problems will be presented. The developed system constitutes a general tool for "Optimizing the Optimization", following an approach that is different from hyper-heuristics, but still applicable to a plethora of problems that require obtaining the global optimum for an analytically expressed objective function. In the literature, there appear to be numerous metaheuristic optimization algorithms that could be employed to that end [19] for design and parameter-tuning purposes. However, as this study encompasses introducing a new way of utilizing them, the purpose is to create a tool that optimizes not only the problem at hand but also the optimization process itself. Within this manuscript, Section 2 ("Materials and Methods") describes the methodology we followed, the quantization of different parameters that we performed, the structure of the individual algorithms and the hybrid scheme. This section also explains the benchmark objective functions that we selected. Section 3 ("Results") presents the outcomes of the conducted research in detailed tables and figures, while Section 4 ("Discussion") thoroughly explains the findings we made by conducting this research, and the breakthrough we identified. Finally, Section 5 ("Conclusions") provides the main points of the conducted research, as well as highlighting areas for improvement and the potential for future work.

2. Materials and Methods

Taking into consideration the review article by Stützle and López-Ibáñez [20], this study aims to create a novel hybrid evolution-based algorithmic scheme to optimize the optimization process, by employing metaheuristic algorithms in an advanced complex scheme. To achieve this goal, a set of optimization algorithms with different characteristics was selected to solve the problem. Note that these algorithms were selected indicatively, as they are well established and utilized in a vast field of applications; they were not selected because of their "better" performance over other more specialized modern ones. Subsequently, through the proposed innovative scheme, these algorithms collaborate to solve the optimization problem, aiming for the optimal solution with the fewest possible evaluations of the objective function. Simultaneously, the approach maximizes the exploration of the solution space and intensifies the search for optima. The architecture of the developed scheme is dynamic and provides the capability to use any combination of algorithms. However, for proof of concept, we selected the following four algorithms, as in

the literature these have been proven very efficient in coping with technological problems that involve large numbers of input and output parameters: the Genetic Algorithm [21], Firefly Algorithm [22], Harmony Search Algorithm [23], and Black Hole Algorithm [19].

2.1. Grouping and Normalization of Unique Initial Values

Optimization algorithms consist of a set of input parameters, which vary for each, depending on their principle and operational philosophies. To create a unified method that can call them simultaneously and cooperatively, these values must be recognized and normalized based on their correlation. As has been mentioned in the literature, all optimization algorithms are governed by two fundamental characteristics: the degree of exploration of the solution space (exploration) and the degree of intensification of specific local regions of the solution (exploitation). Based on this, the normalization of all individual algorithm parameters for the set of algorithms was done by focusing on which of these two fundamental characteristics they influence.

The process begins with the identification of parameters common to all of the selected algorithms:

- **Objective Function**: The function to be solved is common to all algorithms, as this is the base that all optimization algorithms utilize to compare solutions. The algorithms evaluate the quality of solutions based on their ability to minimize or maximize the objective function. In each iteration, all the algorithms' processes are assessed through the objective function.
- **Problem Dimensions**: The dimensions of the objective function are common to all algorithms. Any problem can have dimensions ranging from 1 to N, with each dimension representing an independent variable based on the function's form and the solving objective.
- **Value Range**: The solution space, according to the objective function, has a predetermined range within which algorithms should search for possible solutions. This range is determined by constraints imposed by the objective function itself or the additional constraints set and can vary from a very narrow value field to infinity. The range is characterized by a set of minimum and maximum values, with each dimension having its declared minimum and maximum. Depending on the problem type and constraints, the sets of minima or maxima can be characterized by different values for each independent variable.
- **Number of Iterations**: One of the essential optimization algorithm variables is the total number of iterations it will perform before stopping and returning the best values it has identified up to that point. Since these algorithms are stochastic, and the ideal solution is often unknown, the number of iterations is one of the most crucial factors for the success of an algorithm to converge. For this study, all algorithms must be applied for the same number of iterations. Due to the different architecture of each algorithm, to achieve this, the iterations are measured by the number of objective function evaluations.
- **Population of Solutions**: Each algorithm initializes and maintains a predetermined number of potential solutions. This number is predetermined, and despite changes made to facilitate convergence to an optimal solution, the size remains constant. The number of maintained solutions is equally important for an optimization algorithm because there is a correlation between the population and the speed and quality of convergence, especially as the difficulty level, which is imposed by the objective function, increases.

In addition to the initial input values, which are the same for all of the optimization algorithms used, each algorithm has a unique set of parameters shaped by its particular

characteristics. To enable these algorithms to be called under the same initial conditions, these specific parameters for each algorithm should be mapped based on their influence on either the degree of exploration or the degree of exploitation. Furthermore, for better uniformity, specific algorithm variables that characterize both the degree and the rate of parameterization have been grouped for their influence on the "mutation rate". These variables can take real values from 0 to 1, and for each algorithm, the mapping of these values is performed as follows:

Genetic Algorithm (GA):

- Beta: This value determines the rate of selecting individuals from the solution set for mutation. Due to its influence on expanding the search in the solution space, this value corresponds to "Exploration".
- PC: The term "PC" comes from "Population to Children", which translates to the correlation of the total population with the number of "offspring" or new values generated for each iteration. This variable determines the percentage of differentiation in the set of values, thus corresponding to "Exploitation".
- Gamma: The Gamma variable determines the intensity of crossover between the parameterized values. As this parameterization affects the way the solution space is explored, this term corresponds to "Exploration".
- MU: MU determines the percentage of features parameterized in each iteration for each chromosome. In other words, this value determines how much each chromosome to be parameterized changes. This value corresponds to the "Mutation Rate".
- Sigma: The last parameter that characterizes only GAs is Sigma. This parameter determines the size of parameterization for each chromosome to be parameterized, depending on MU. This variable also corresponds to the "Mutation Rate".

Firefly Algorithm (FA):

- Alpha: This variable determines the degree of randomization with which fireflies move that do not have the highest "intensity" or suitability as per the objective function. Thus, their degree of change is partly determined by this parameter, and for this reason, it corresponds to the "Mutation Rate".
- Beta: This term determines the degree of attraction with which fireflies are attracted to the "radiance" of the "brightest", or in other words, the better solution. The larger this term, the greater the attraction exhibited by the optimal fireflies toward the others, creating a tendency to gather the others towards them, thereby increasing the "Exploitation" characteristic of the algorithm.
- Gamma: This term collaborates with the Beta value and determines the degree to which the remaining fireflies react to this attraction, or how much light they allow to be absorbed and influence them. Conversely, this term determines the "Exploration".

Harmony Search Algorithm (HSA):

- Harmony Memory Considering Rate (HMRC): With this term, the algorithm decides how many of the stored values (melodies) in the form of a solution population it will choose to use per iteration compared to new ones that will be generated. Thus, the size of the solution space in which the algorithm will search for new solutions is determined, and it corresponds to "Exploration".
- Pitch Adjusting Rate (PAR): This variable characterizes the maximum allowable differentiation that an existing solution can have when it is deemed necessary to parameterize. In short, with the term PAR, it is determined how close a new solution will appear to the existing solution, thus determining the "Mutation Rate".
- Maximum Pitch Adjustment Proportion/Index (MPAP/MPAI): These two parameters essentially have the exact influence on a harmony search algorithm. One is used

when the function to be solved is continuous, while the other is used when we have a function with discrete values. In practice, these variables define the range of conversion that all parameterized values-melodies will undergo for each iteration. Consequently, it directly affects "Exploitation".

For the **Black Hole Algorithm (BHA),** there are no specific predefined variables beyond the common initial input values described earlier.

In summary, to enable the four selected algorithms to "run" in a mixed scheme with the same initial values, the individual unique variables governing the operation of each algorithm should be grouped based on some common features characterizing all of the optimization algorithms. More specifically:

- Exploration Degree:
  - Beta, Gamma (Genetic Algorithm)
  - Gamma (Firefly Algorithm)
  - HMRC (Harmony Search Algorithm)
- Exploitation Degree:
  - PC (Genetic Algorithm)
  - Beta (Firefly Algorithm)
  - MBAP/MBAI (Harmony Search Algorithm)
- Mutation Rate:
  - MU, Sigma (Genetic Algorithm)
  - Alpha (Firefly Algorithm)
  - PAR (Harmony Search Algorithm)

By grouping the above variables, in combination with the common parameters introduced in each algorithm, a mixed scheme can be designed with which they will be invoked.

2.2. Algorithmic Scheme

The goal is to create a hybrid optimization algorithm scheme that calls upon these algorithms with the aim of solving a common objective function. To achieve this goal, a novel algorithm will be developed, acting as the leader for this hybrid scheme. If all the algorithms are considered players in a cooperative problem-solving game, the leader algorithm will determine their initial values, input variables for each iteration, as well as when and how they should stop. To start, the selection of an objective function is crucial. Reference functions from the literature will be used for evaluating the performance of the scheme and for measuring the time required for the scheme to reach predefined accuracy levels for known solutions. By utilizing these functions, the search range of solutions, the function's elasticity for use with multiple independent variable values, the number of local optima, and the global optimum, along with the value extracted from this reference objective function, are known from the beginning. With the entire value domain known, it becomes possible to evaluate each optimization algorithm separately and the overall performance of the scheme, both individually and in comparison, to find a solution with each algorithm separately if the scheme is not utilized.

2.2.1. Objective Functions

For proof of concept, three different reference functions from the literature [24] will be utilized. Each one has its own value range and optimal value at its specific point, and all were selected to provide solutions for multiple choices of independent variables, depending on the preselection made during the initialization of values. This allows the same function to be tested within various dimensions, thus evaluating the solution quality as complexity increases. The functions used for this particular study appear in Table 1.

Table 1. Reference functions selected, their value range and optimized result.

Function	Objective Function	Value Range	Optimized Result
Sphere	$f(x) = \sum_{i=1}^{D} x_i^2$	$-\infty \leq x \leq +\infty$	$x^* = f(0, \ldots, 0),\ f(x^*) = 0$
Schwefel	$f(x) = 418.9829d - \sum_{i=1}^{d} x_i sin(\sqrt{\lvert x_i \rvert})$	$-50 \leq x_i \leq 50$	$f(420.9687, \ldots, 420.9687) = 0$
Alpine N.2	$f(x) = \prod_{i=1}^{D} \sqrt{x_i} sin(x_i)$	$0 \leq x_i \leq 10$	$x^* = f(7.917, \ldots, 7.917),$ $f(x^*) = 2.808^D$

2.2.2. Architecture of the Algorithmic Scheme

Having defined the objective function for optimization and prepared the algorithms regarding their input variables and their association with their respective influencing factors (Exploration, Exploitation, and Mutation Rate), the leader algorithm provides initial values to feed them. After the initial feeding, all algorithms attempt to solve the problem with equal treatment from the leader. For optimization of the optimization, the algorithms need to interact with each other from iteration to iteration. This is achieved by setting all algorithms to simultaneously optimize the same problem for a finite number of iterations, evaluating their results after each iteration. The algorithm that exhibits the best performance will feed all of the others with the optimal solution, while in the case of the worst-performing algorithm, it will be fed with the entire population of solutions from the best one, thus aiding through the elitism method to increase the chances of finding a better point in the next iteration (see Figure 1). For the others, apart from being fed with the optimal solution, the rest of the population remains constant from their previous iteration. In this way, by setting a small number of iterations for each individual algorithm at each step of the mixed scheme, convergence to the global optimum is sought with as few evaluations of the objective function as possible overall.

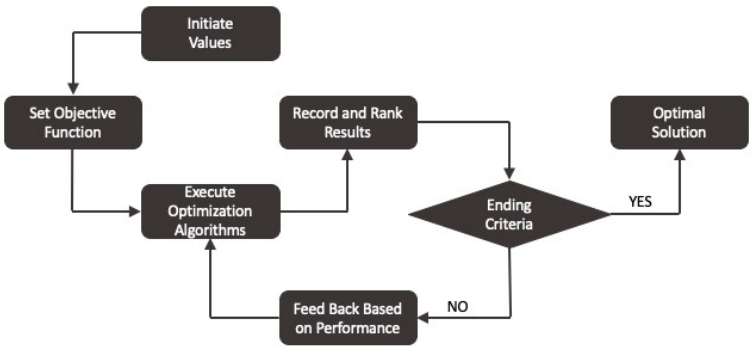


Figure 1. Overview flow chart of the hybrid algorithmic scheme.

By keeping the values for the exploration degree, exploitation degree, and mutation rate constant and in the middle of their range, it is ensured that the exploration and exploitation levels of the algorithms will remain at their average and equal for all players. The leader, by feeding all algorithms with information from the algorithm that has the best performance, indirectly attempts to maximize both the exploration and exploitation degrees simultaneously for the scheme. This is achieved without sacrificing time to evaluate the objective function more times, while at the same time minimizing the risk of the mixed scheme getting trapped in a local optimum.

The mixed optimization scheme guided by the leader will repeat the above process until a termination criterion is satisfied. This criterion may either concern the overall convergence degree (e.g., the order of magnitude of convergence relative to an ideal

termination value) or a predefined, finite number of iterations for a problem with an unknown ideal value.

2.2.3. Initial Values

These values, as defined by the optimization algorithms' literature, are the following:

> Problem dimensions (independent variables) (e.g., d = 2);
> Minimum value in the domain (where: Min = [value] * d, for creating a minimum for each dimension);
> Maximum value in the domain (where: Max = [value] * d, for creating a maximum for each dimension);
> Number of iterations (e.g., iterations = 20);
> Population size (e.g., population = 1000);
> Exploration rate (e.g., exploration = 0.5);
> Exploitation rate (e.g., exploitation = 0.5);
> Mutation rate (e.g., mutation_rate = 0.5).

These initial values are manually defined intuitively, compressed into a single variable with slave values (variable type: Structure), and fed as a 'package' to all of the optimization algorithms, which, in turn, distribute them to their corresponding specific parameters.

2.2.4. Special Values

To achieve the smooth operation of the scheme, some additional initial values must be determined, characterized as special for the leader. These values assist the leader algorithm in making decisions regarding the distribution of results per iteration step of the mixed scheme and determining the criterion by which it will terminate the process and extract the optimal result. These variables are being created through the results of each iteration within the leader algorithm, and are automatically allocated accordingly. These values are defined as follows:

> Optimal solution: The optimal solution is the best-performing value presented by the best algorithm compared to the others, e.g., for a problem with two independent variables, the point (1,1).
> Fitness of the optimal solution (Cost): Fitness is defined as the value given at the output of the objective function at the point of the optimal solution, e.g., Cost = 2, related to fitness for the point (1,1) and the function $f(x, y) = x + y$.
> Selection index: To allow the leader to determine which player performs better or worse, there must be a logical index (true/false) that, in each iteration of the mixed scheme, takes the appropriate value concerning the performance of the players. This selection index adjusts to the compressed single variable fed to the algorithms. Thus, the algorithms know whether they should consider either a better solution or an entire population for their next optimization attempt.
> Distributed performance list: For feedback to the user during the execution of the mixed scheme and for returning the optimal solution at all levels (optimal cost, optimal position), a distributed performance list is created. This list ranks the algorithms from best to worst and returns the corresponding index for each.

2.2.5. Objective Function Evaluations

In addition to the above parameters, it is crucial to align how each algorithm will attempt to improve the objective function. The aim of the overall study is the optimization of optimization; therefore, for the results to be reliable, there must be homogeneity among the players of the mixed scheme. Each player-algorithm, for a set number of optimization iterations, examines a possible optimal solution a predetermined number of times. This

means that, when assigning initial values to individual optimization algorithms, it must be considered that for one iteration of the mixed scheme, all algorithms will have evaluated the objective function an equal number of times, thus avoiding bias in favor of any algorithm performing more evaluations than the others in an iteration.

2.3. Programming Environment

For the implementation of the above, a programming language should be selected to transfer these relationships into a digital computational environment. In theory, any programming language can fulfill this need. For the purposes of this study, Python has been chosen. Especially in the field of optimization, Python is an indispensable tool due to its low complexity and flexibility in quickly transforming models and mathematical relationships into code.

For this study, all the code was implemented using Python version 3.7.10 within the Anaconda programming environment. Sublime Text software (Version 4126) was used for code processing, while the implementation and execution of the code were carried out using version 5.0.0 of the Spyder software. All development and testing were performed on an Apple MacBook Pro laptop with an M1 Pro processor and 16 GB of RAM, purchased in Greece.

2.4. Hybrid Scheme Rules and Programming

After structuring the player-algorithms and the mixed scheme at a theoretical level, the next step is to translate all this knowledge into a computational environment using Python. Starting with the optimization algorithms, all of them need to be implemented in code. As optimization algorithms have been a topic in the literature for several years, and Python is an open-source language, implementations are readily available for use by the programming community. These implementations are widely accessible but require extensive parameterization to adapt and integrate into the mixed scheme of this research.

For each algorithm, an implementation was chosen that aligns with the goals of the mixed scheme and was parameterized according to the following rules:

1. All algorithms should "accept" and "perceive" the values of the domain in the same way. This means that for each function $f(x_i)$ fed into the algorithms, where x_i represents the value from the domain for each time instance, it should have a common variable format for both the leader algorithm and the players. This format was chosen as "NumPy Array". In practice, NumPy Arrays are numeric lists created by the "NumPy" library, which is part of Python's source code libraries. This choice was made to ensure input uniformity based on an independent, standalone library from the programming language.
2. All algorithms should output the optimal value from the domain and the set of solution populations as a list, decoded by NumPy. This choice serves the leader algorithm to manage, compare, and present data independently of how they are generated in the individual player algorithms.
3. All algorithms should be able to assign initial and special values appropriately, according to the correspondences made in their theoretical study.
4. All algorithms should be able to extract the same parameters in the same format. Specifically, the output values for all of the algorithms are as follows: (i) optimal input value, (ii) optimal value of the objective function, and (iii) total population at the last iteration.
5. All algorithms should parameterize the value of the maximum iterations given to them, according to the number of evaluations of the objective function they perform

in total, so that all of the algorithms evaluate the objective function the same number of times in each "call".

6. All algorithms should be able to accept the objective function encoded in the same way.
7. The algorithms should be in the form of a library, which will be imported by the leader, who, in turn, can call on the execution method, providing the compressed initial and special values and the objective function as input variables.

Having parameterized the algorithms accordingly, the next step is to structure the leader algorithm. The structure of the leader is described in pseudocode as in Figure 2.

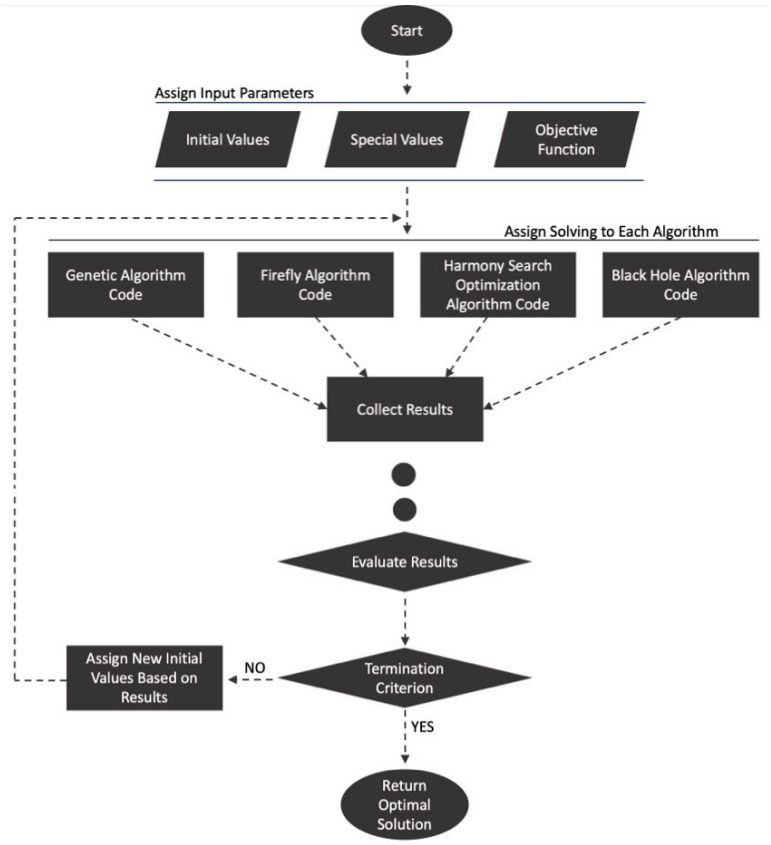


Figure 2. Pseudocode sequence of the hybrid scheme leader.

The form of the leader can either act as a coordinator of the mixed scheme or simply as a single controller of the individual algorithms without creating interaction between them for their isolated evaluation. Also, it must be clarified that in order to maximize the convergence speed, after each iteration the optimal results obtained by any of the algorithms, but not whole populations, are seeded to all the algorithms of the next iteration. The first stage of this study is to test the performance of the algorithms for each reference function separately, in order to draw conclusions regarding their optimization capability. Assessing the quality of solutions for each optimization algorithm separately, based on the aforementioned benchmark functions, is then applied to the mixed scheme with the same problem. The goal is to achieve comparable or better results with overall fewer evaluations of the objective function.

3. Results

3.1. Testing Optimization Algorithms Individually

For the evaluation of the algorithms, a problem-solving approach was applied to the three reference functions, where the optimal value is previously known from the

literature. This knowledge, however, was not introduced to the algorithms; thus, an objective assessment of the algorithms' ability to optimize can be performed. By varying the total number of iterations, problem dimensions, and solution population for each trial, the results of Tables 2–4 were obtained. For each evaluation, the algorithms executed a fixed number of iterations with a predefined population of solutions, and the solution values retrieved were the ones that achieved the best fitness at the end of each complete cycle of iterations. For each different set of parameters, a completely new trial was performed, generating new initial values and running independently of the others. Finally, for each reference function, a final trial with the maximum referenced population of solutions was perceived as indicative of the effectiveness of each algorithm in providing a solution that was within 10^{-2} of the desired one.

Table 2. Sphere reference function optimization results. Legend: ITE = Number of iterations with a complete population, POP = Population number, GA = Genetic Algorithm, DEV = Absolute deviation of returned solution to the ideal one, FA = Firefly Algorithm, HAS= Harmony Search Algorithm, BHA = Black Hole Algorithm.

Dimensions—Independent Values = 2									
ITE	**POP**	**GA**	**DEV-GA**	**FA**	**DEV-FA**	**HSA**	**DEV-HSA**	**BHA**	**DEV-BHA**
50	200	2.249×10^{-7}	2.249×10^{-7}	0.415	0.415	0.171	0.171	0.020	0.020
	400	2.990×10^{-7}	2.990×10^{-7}	0.340	0.340	0.018	0.018	0.343	0.343
	1000	2.599×10^{-8}	2.599×10^{-8}	0.318	0.318	0.096	0.096	0.082	0.082
240	200	1.868×10^{-21}	1.868×10^{-21}	0.010	0.010	0.407	0.407	0.247	0.247
	400	1.762×10^{-21}	1.762×10^{-21}	0.456	0.456	0.082	0.082	0.075	0.075
	1000	3.057×10^{-23}	3.057×10^{-23}	0.095	0.095	0.013	0.013	0.057	0.057
Comparison of each algorithm reaching an accuracy of 10^{-2}									
Algorithm		ITE		POP		Best Value Returned			
GA		20		1000		1.880×10^{-5}			
FA		1200		1000		2.464×10^{-4}			
HSA		1000		1000		4.654×10^{-3}			
BHA		700		1000		1.455×10^{-3}			
Dimensions—Independent Values = 3									
ITE	**POP**	**GA**	**DEV-GA**	**FA**	**DEV-FA**	**HSA**	**DEV-HSA**	**BHA**	**DEV-BHA**
50	200	7.241×10^{-5}	7.241×10^{-5}	2.209	2.209	1.702	1.702	0.488	0.488
	400	4.668×10^{-5}	4.668×10^{-5}	2.061	2.061	7.337	7.337	2.300	2.300
	1000	1.557×10^{-5}	1.557×10^{-5}	1.572	1.572	0.318	0.318	3.049	3.049
240	200	2.727×10^{-10}	2.727×10^{-10}	0.469	0.469	0.354	0.354	3.053	3.053
	400	1.009×10^{-10}	1.009×10^{-10}	1.600	1.600	0.635	0.635	3.828	3.828
	1000	5.415×10^{-11}	5.415×10^{-11}	0.512	0.512	3.337	3.337	3.256	3.256
Comparison of each algorithm reaching an accuracy of 10^{-2}									
Algorithm		ITE		POP		Best Value Returned			
GA		20		1000		1.694×10^{-3}			
FA		2400		1000		1.130×10^{-3}			
HSA		30,000		1000		7.109×10^{-3}			
BHA		-		-		No convergence for more than 10^{-1}			

Table 3. Schwefel reference function optimization results. Legend: ITE = Number of iterations with a complete population, POP = Population number, GA = Genetic Algorithm, DEV = Absolute deviation of returned solution to the ideal one, FA = Firefly Algorithm, HSA = Harmony Search Algorithm, BHA = Black Hole Algorithm.

Dimensions—Independent Values = 2									
ITE	**POP**	**GA**	**DEV-GA**	**FA**	**DEV-FA**	**HSA**	**DEV-HSA**	**BHA**	**DEV-BHA**
50	200	3.198×10^{-5}	3.198×10^{-5}	44.615	44.615	131.036	131.036	101.910	101.910
	400	2.634×10^{-5}	2.634×10^{-5}	7.410	7.410	53.005	53.005	44.155	44.155
	1000	2.589×10^{-5}	2.589×10^{-5}	41.289	41.289	8.540	8.540	118.362	118.362
240	200	2.545×10^{-5}	2.545×10^{-5}	4.721	4.721	0.720	0.720	3.762	3.762
	400	2.545×10^{-5}	2.545×10^{-5}	48.525	48.525	31.229	31.229	196.613	196.613
	1000	2.545×10^{-5}	2.545×10^{-5}	43.090	43.090	97.473	97.473	60.672	60.672
Comparison of each algorithm reaching an accuracy of 10^{-2}									
Algorithm		ITE		POP		Best Value Returned			
GA		30		1000		1.120×10^{-4}			
FA		10,000		1000		7.456×10^{-2}			
HSA		10,000		1000		3.748×10^{-2}			
BHA		-		-		No convergence			
Dimensions—Independent Values = 3									
ITE	**POP**	**GA**	**DEV-GA**	**FA**	**DEV-FA**	**HSA**	**DEV-HSA**	**BHA**	**DEV-BHA**
50	200	0.430	0.430	334.301	334.301	445.006	445.006	370.047	370.047
	400	0.013	0.013	101.121	101.121	114.760	114.760	270.679	270.679
	1000	0.001	0.001	338.982	338.982	153.056	153.056	152.497	152.497
240	200	3.818×10^{-5}	3.818×10^{-5}	128.789	128.789	136.208	136.208	306.345	306.345
	400	3.818×10^{-5}	3.818×10^{-5}	347.940	347.940	223.536	223.536	272.299	272.299
	1000	3.818×10^{-5}	3.818×10^{-5}	218.610	218.610	255.159	255.159	189.642	189.642
Comparison of each algorithm reaching an accuracy of 10^{-2}									
Algorithm		ITE		POP		Best Value Returned			
GA		50		1000		3.546×10^{-3}			
FA		-		-		No convergence			
HSA		150,000		1000		9.954×10^{-2}			
BHA		-		-		No convergence			

Table 4. Alpine N.2 reference function optimization results. Legend: ITE = Number of iterations with a complete population, POP = Population number, GA = Genetic Algorithm, DEV = Absolute deviation of returned solution to the ideal one, FA = Firefly Algorithm, HSA = Harmony Search Algorithm, BHA = Black Hole Algorithm.

Dimensions—Independent Values = 2									
ITE	**POP**	**GA**	**DEV-GA**	**FA**	**DEV-FA**	**HSA**	**DEV-HSA**	**BHA**	**DEV-BHA**
50	200	7.885	0.001	7.836	0.048	7.343	0.541	6.681	1.203
	400	7.885	0.001	7.795	0.089	7.872	0.012	7.759	0.125
	1000	7.885	0.001	7.710	0.174	7.846	0.038	7.865	0.019
240	200	7.885	0.001	7.871	0.013	7.753	0.131	7.678	0.206
	400	7.885	0.001	7.861	0.023	7.617	0.267	7.592	0.292
	1000	7.885	0.001	7.878	0.006	7.295	0.589	7.671	0.213
Comparison of each algorithm reaching an accuracy of 10^{-2}									
Algorithm		ITE		POP		Best Value Returned			
GA		3		1000		7.821			
FA		300		1000		7.847			
HSA		300		1000		7.463			
BHA		200		1000		7.803			
Dimensions—Independent Values = 3									
ITE	**POP**	**GA**	**DEV-GA**	**FA**	**DEV-FA**	**HSA**	**DEV-HSA**	**BHA**	**DEV-BHA**
50	200	22.130	0.013	14.552	7.588	16.963	5.171	10.118	12.022
	400	22.141	0.001	16.225	5.915	19.221	2.919	20.047	2.093
	1000	22.143	0.003	18.846	3.294	20.776	1.364	21.504	0.636
240	200	22.143	0.003	21.433	0.707	19.410	2.730	9.617	12.523
	400	22.143	0.003	20.100	2.040	20.087	2.053	20.170	1.970
	1000	22.143	0.003	16.885	5.255	18.975	3.165	20.260	1.880
Comparison of each algorithm reaching an accuracy of 10^{-2}									
Algorithm		ITE		POP		Best Value Returned			
GA		30		1000		22.042			
FA		3000		1000		22.075			
HSA		13,000		1000		22.055			
BHA		-		-		No convergence			

3.1.1. Sphere Reference Function

The optimal solution of the function is 0, at the point $f(0,0)$. We set a fixed range of values with a minimum of $[-10]$ and a maximum of [10] for each dimension, and we obtained the solutions as seen in Table 2, with a graphical representation in Figure 3.

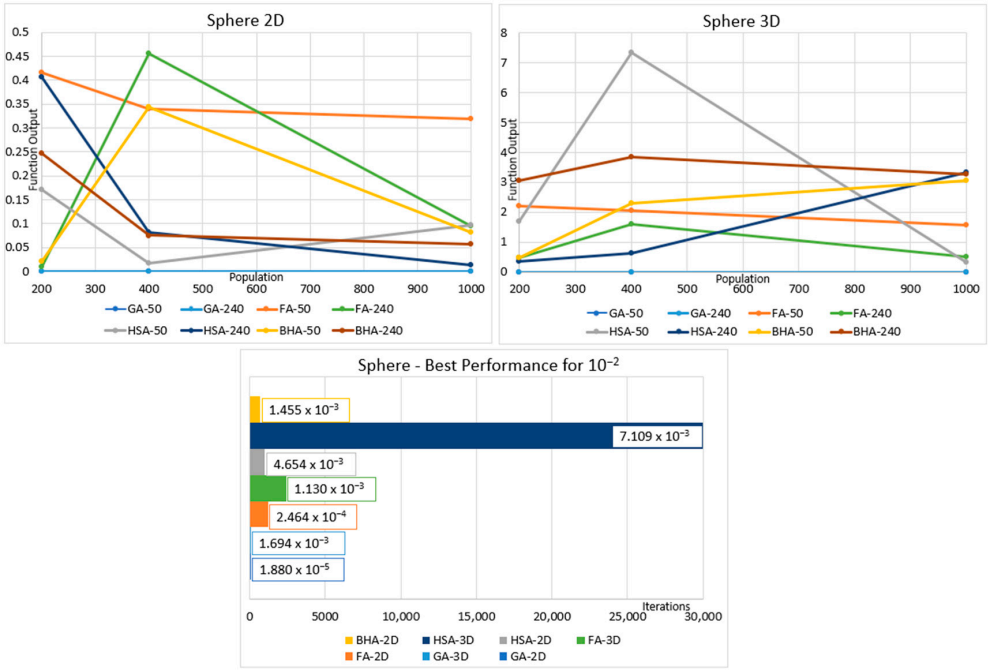


Figure 3. Representation of the results for the Sphere concerning 2D (**upper left**), 3D (**upper right**) and accuracy (**bottom**).

3.1.2. Schwefel Reference Function

The optimal solution of the function is 0, at the point $f(420.9687, \ldots, 420.9687)$. We set a fixed range of values with a minimum of $[-500]$ and a maximum of [500] for each dimension, and we obtained the solutions as seen in Table 3, with a graphical representation in Figure 4.

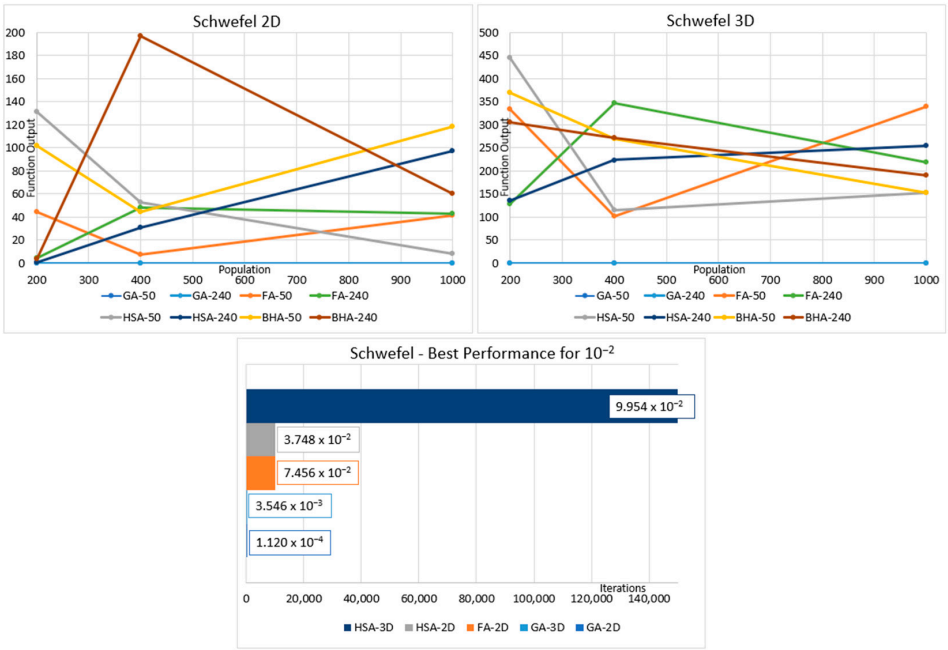


Figure 4. Representation of the results for the Schwefel concerning 2D (**upper left**), 3D (**upper right**) and accuracy (**bottom**).

3.1.3. Alpine n.2 Reference Function

The optimal solution of the function is 2.808^D, where D is the number of dimensions, at the point $f(7.917, \ldots, 7.917)$. For two dimensions, the optimal value is 7.884764, while

for three dimensions, it is 22.140698. We set a fixed range of values with a minimum of [0] and a maximum of [10] for each dimension, and we obtained the solutions as seen in Table 4, with a graphical representation in Figure 5.

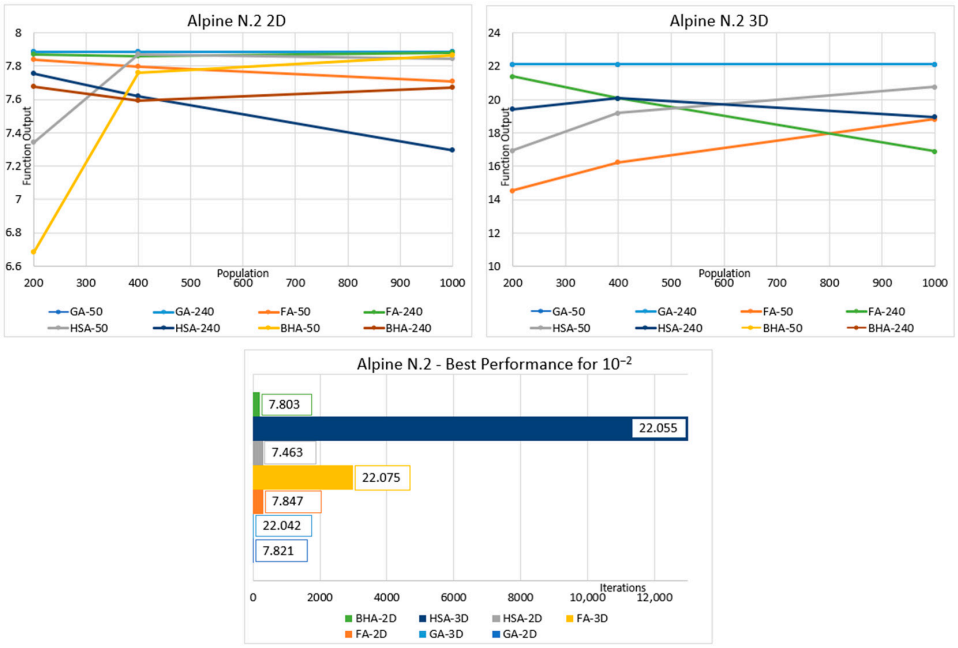


Figure 5. Representation of the results for the Alpine N.2 concerning 2D (**upper left**), 3D (**upper right**) and accuracy (**bottom**).

3.2. Testing the Hybrid Scheme

Having examined how each algorithm individually solves the optimization problem, it is necessary to test the solution of the same problem by the hybrid scheme described in Section 2.2. With the same initial values defined for all of the algorithms and having configured the way values will be fed, we obtained the solution arrays of Table 5 for each reference function, which are graphically represented in Figure 6.

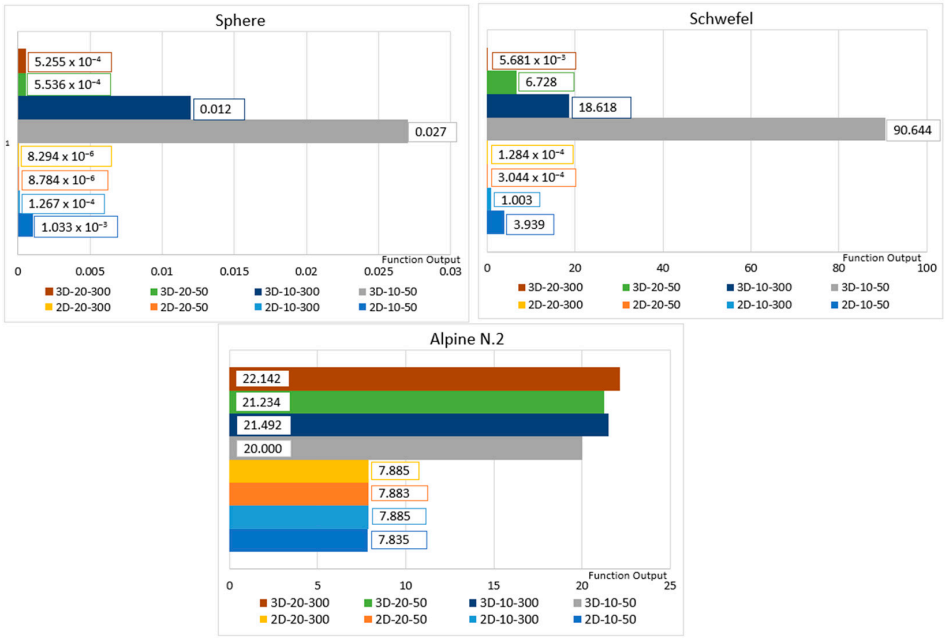


Figure 6. Representation of the results regarding the performance of the Hybrid scheme for Sphere (**left**), Schwefel (**right**), and Alpine N.2 (**center**).

Table 5. Hybrid scheme optimization results for all reference functions.

Sphere Reference Function				
Dimensions—Independent Values = 2				
Number of iterations with a complete population for each algorithm	**Population for each algorithm**	**Total runs of the hybrid scheme**	**Returned output value of objective function**	**Total hybrid scheme deviation**
10	50	2	1.033×10^{-3}	0.001
10	300	2	1.267×10^{-4}	0.0001
20	50	2	8.784×10^{-6}	10^{-6}
20	300	2	8.294×10^{-6}	10^{-6}
Total effort to reach an accuracy of 10^{-2}				
7	50	2	2.658×10^{-2}	
Dimensions—Independent Values = 3				
10	50	2	0.027	0.027
10	300	2	0.012	0.012
20	50	2	5.536×10^{-4}	10^{-4}
20	300	2	5.255×10^{-4}	10^{-4}
Total effort to reach an accuracy of 10^{-2}				
10	50	2	2.700×10^{-2}	
Schwefel reference function				
Dimensions—Independent Values = 2				
10	50	2	3.939	3.939
10	300	2	1.003	1.003
20	50	2	3.044×10^{-4}	10^{-4}
20	300	2	1.284×10^{-4}	10^{-4}
Total effort to reach an accuracy of 10^{-2}				
16	50	2	5.532×10^{-2}	
Dimensions—Independent Values = 3				
10	50	2	90.644	90.644
10	300	2	18.618	18.618
20	50	2	6.728	6.728
20	300	2	5.681×10^{-3}	10^{-3}
Total effort to reach an accuracy of 10^{-2}				
20	300	2	5.681×10^{-3}	
Alpine N.2 reference function				
Dimensions—Independent Values = 2				
10	50	2	7.835	0.049
10	300	2	7.885	0.001
20	50	2	7.883	−0.001
20	300	2	7.885	0.001
Total effort to reach an accuracy of 10^{-2}				
10	50	2	7.835	
Dimensions—Independent Values = 3				
10	50	2	20.000	2.140
10	300	2	21.492	0.647
20	50	2	21.234	0.906
20	300	2	22.142	0.002
Total effort to reach an accuracy of 10^{-2}				
20	300	2	22.137	

4. Discussion

4.1. Evaluation Summary for Individual Algorithms

All of the individual trials were performed for the same fixed number of iterations, number of population, and dimensions. This does not reflect the way each algorithm is "preferably" configured, but this has been selected on purpose. For example, since the Genetic Algorithms are inherently ideal for solving similar problems, they were expected to behave much better, across all of the trials. At the same time, a population of 1000 solutions per iteration is extreme, but was chosen to provide the rest of the algorithms an equalizing factor compared to the GA.

All algorithms behave differently in solving the same optimization problem. It was observed that the number of individuals in the population is extremely important compared to the number of objective function evaluations. In cases where the number is extremely low, the performance of the algorithms, except for the Genetic Algorithm, is very low and unstable, as there is not enough time to create a satisfactory convergence level. Similarly, as complexity increases in the dimensions and value ranges, things become even more challenging for the algorithms. Solutions have a very wide range, while their distance from the ideal solution fluctuates significantly depending on the initial parameter settings. In addition, since the nature of the selected optimization algorithms is stochastic, the initial values are generated at random. This randomness explains the large variability that all the algorithms have, apart from the better-suited GA. This variability was expected and desired, to highlight the different approach that each algorithm has, since this can become the strengthening factor when using these algorithms in a hybrid scheme. Finally, it was observed that, except for the Genetic Algorithms, when convergence concerns the optimal value at 0, the other algorithms significantly struggle to converge, and sometimes do not converge at all. This reinforces the argument that different algorithms are more suitable for different optimization problems. Some specific observations include:

- Genetic Algorithm: Exceptional ability to solve problems in a very short time for the majority of optimization problems.
- Particle Swarm Algorithm: Good problem-solving ability, but exhibits high instability depending on the combinations of initial values. Solution accuracy rarely exceeds 10^{-2}, and it struggles to converge for optimal values close to 0.
- Harmony Search Algorithm: The agnostic way, meaning the way it globally evaluates the entire domain value it maintains as a population, requires an exceptionally large number of iterations to solve a problem. Nevertheless, the solutions converge with very high accuracy to the desired target.
- Black Hole Algorithm: The inability to adjust its reaction method makes it extremely unstable, to the point of not being able to converge in some cases. Its results have a high degree of randomness, regardless of the initial parameter values provided. As the domain of definition increases, so does the instability it exhibits.

4.2. Comparing the Hybrid Scheme to Individual Algorithms

From the above process, we observed that the mixed-solution approach to optimization problems demonstrates several benefits. When comparing the individual performance of algorithms, our main observation is that algorithms require a certain number of evaluations of the objective function to achieve convergences of reasonable accuracy. The best performance was observed from the Genetic Algorithm, which consistently behaved well in solving all reference problems. Other algorithms required significantly more time to solve the same problem, especially when the dimensions (independent variables of the problem) were more than two. In such cases, many algorithms struggled to converge to

a global optimum, and in some cases, as in the Black Hole Algorithm, convergence was not achievable.

For minimization of the run time of the mixed optimization scheme, each algorithm was assigned to a different thread of a multi-thread CPU. This is easily applied, as each algorithm in each iteration is independent. The mixed optimization system that was introduced and implemented in this study seems to solve all the individual problems presented by the algorithms without sacrificing performance in any area. In all cases, the system achieves convergence with significantly fewer evaluations of the objective function, as shown in Figure 7. Based on individual evaluations for each algorithm, the difference in performance is at least three times better, meaning the mixed approach needs to evaluate the objective function by only one-third of the number needed by the best-performing standalone algorithm. This can be translated by multiplying the number of iterations to the number of populations selected to retrieve a total "effort" made by the processor. For example, for the best performing GA, in solving the Schwefel function in 3D, the processor running 50 ITE * 1000 POP returned with a best value within 10^{-3} from the target, while with the hybrid scheme, the same calculation only cost the processor 20 ITE * 300 POP, for very similar accuracy. This translates to roughly eight times less computational cost for the same accuracy. If we compare this to HSA, for example, on the same reference function it required some significant 150,000 ITE * 1000 POP to achieve a solution with an accuracy of 10^{-2}, thus requiring a 250 times higher computational cost! Additionally, during the use of the mixed approach, an index was used to indicate the algorithm that achieved the optimal solution. This was done to evaluate the system itself, as a scheme that always takes the optimal solution from the same algorithm has limitations in its use. According to this logic, while individual evaluation showed that the performance of the Genetic Algorithm was stable and better than the others, during the use of the mixed approach, it was observed that the Genetic Algorithm did not always provide the optimal solution, making the scheme flexible against the diversity of optimization problems.

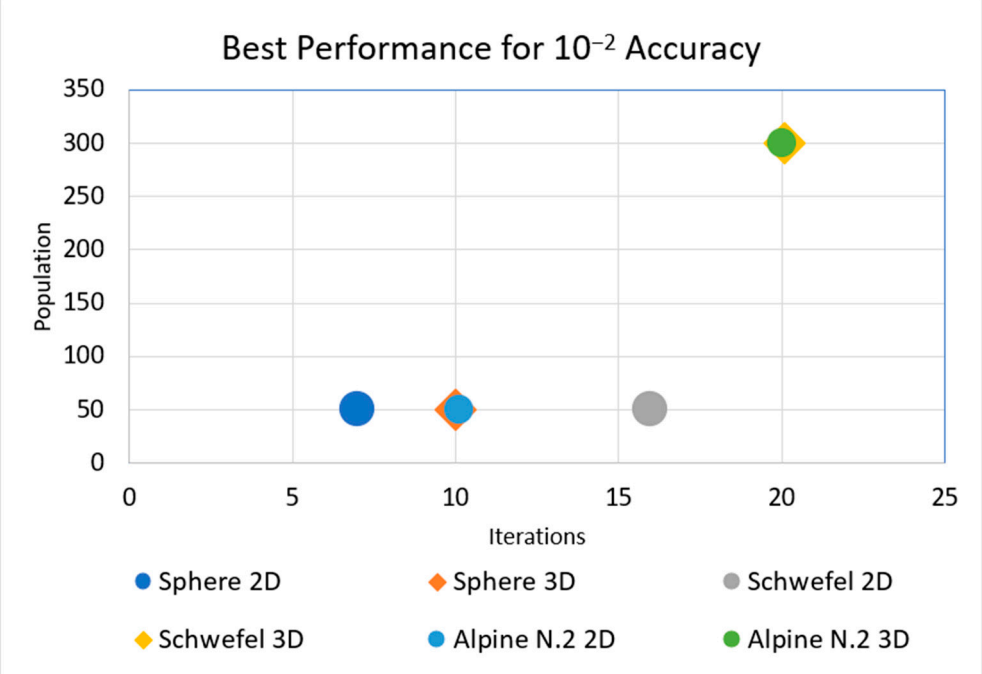


Figure 7. Performance graph for solution accuracy of the hybrid scheme. Lower is better.

4.3. Identified Breakthrough

Although only few other similar approaches have been published, such as in [25] and [26], the significant advantage observed in the proposed mixed approach is that, with very few iterations during its initial call, the algorithm with the best performance dramatically accelerates the others, giving them a significant boost towards the global optimum without trapping them in a local optimum. Thus, with a form of elitism, the best-performing algorithm in the scheme collaboratively shares its findings with the team, which often surpasses it in performance in subsequent steps. Therefore, by leveraging the capabilities of different algorithms, the global optimum in an optimization problem is found much more quickly, without the scheme being affected by the poor performance of a specific algorithm-player. For this to happen, all algorithms must perform poorly on the same optimization problem. The scheme's diversity, capitalizing on each algorithm's performance quality for a different problem, helps the entire optimization process converge to a global optimum in a better time frame at a significantly lower calculation cost, regardless of the specific form of the problem's objective function. Practically, this means that any optimization algorithm can be utilized, and the scheme has dynamic length, meaning it can be functional with two algorithms up to N, where N is the total number of optimization algorithms in the literature.

5. Conclusions

Through this research, an attempt was made to create an innovative mixed solving scheme for optimization problems. The goal was for this scheme to be able to solve optimization problems by performing the fewest possible evaluations of the objective function. Observing the scheme, we conclude that such a method could evolve into a powerful solving tool applicable to all areas facing optimization problems. Through calculation trials using these algorithms to optimize predefined reference functions with known best solutions, we realized that by using a hybrid scheme with multiple optimization algorithms, the entire process of optimization gets optimized, and the best solution can be found at least three times faster than by using the standalone best performing algorithm and in some cases at two or three magnitudes faster. Even though the selected reference functions are relatively easy to solve using GA, they pose as a powerful tool to demonstrate the potential of the hybrid scheme by merging more modern algorithms that are not optimized for solving such functions, but perform better in other applications. Following this logic, by using a hybrid tool like the one proposed in this manuscript, complex problems that require a lot of time for selecting and fine-tuning the most appropriate optimization algorithm can be solved just by employing a group of algorithms with uniform input values, competing with each other and sharing their best performers after each iteration.

While the present study focuses on demonstrating the feasibility of applying such a scheme by solving problems with known optima, the prospects for the future development of the scheme are excellent. A tremendous advantage is the dynamic use of algorithms. This scheme can exploit as few as two algorithms in its basic form and up to as many as the user wishes to introduce. Thus, in theory, a user could solve all possible problems for all possible objective functions. As long as any of the involved algorithms converges for a specific problem with a well-formulated objective function in a well-formulated solution space, the proposed method converges. Moreover, the proposed mixed algorithm will converge at least as fast, but in reality, multiple times faster, than the fastest of the involved algorithms. The scheme would automatically promote the best algorithm for solving, obtaining results in the shortest possible time, at a fraction of the calculation time needed for any individual optimization algorithm, and requiring the fewest parameter adjustments for each different problem.

There are a few identified areas in which the scheme can be improved and the research can be progressed. Firstly, for optimization algorithms themselves, the scheme can dynamically evolve along with the development of optimization algorithms by systematically evaluating the literature and considering various optimization schemes. Furthermore, the scheme and how it feeds algorithms in each iteration can be improved, with researchers looking to improve the scheme by experimenting with different feeding criteria for algorithms between each iteration of the mixed scheme. The scheme itself could even take the form of an optimization algorithm, making the solving process more efficient. Another point of improvement could be the selected programming language. Even with the use of the same programming language (Python), an experienced software engineer could dramatically optimize the code, from writing style to resource allocation for solving, significantly improving the time required to complete the process. Additionally, lower-level programming languages, generally much faster than Python and with greater accuracy in resource distribution for optimization, could be explored, such as C++. Finally, the specialization of the scheme regarding the field of interest could be an improvement. Depending on the field to which an optimization problem belongs (e.g., mechanical engineering, physics, or biology), the scheme could be parameterized by domain experts to better fit the problems associated with each research field.

Author Contributions: A.A.K.: Conceptualization, Methodology, Writing—original draft preparation, Writing—review and editing, Supervision. L.A.: Conceptualization, Formal analysis and investigation, Resources, Writing—original draft preparation, Software creation, Algorithm integration. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: No data were used for this research, only mathematical modeling.

Conflicts of Interest: The authors declare no conflicts of interest. All authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript.

References

1. Dantzig, G. *The Nature of Mathematical Programming*; Mathematical Programming Glossary; INFORMS Computing Society: Catonsville, MD, USA, 2014.
2. Martins, J.R.R.A.; Ning, A. Engineering Design Optimization. Available online: https://www.cambridge.org/highereducation/books/engineering-design-optimization/B1B23D00AF79E45502C4649A0E43135B#overview (accessed on 16 September 2024).
3. Du, D.-Z.; Pardalos, P.M.; Wu, W. History of Optimization. In *Encyclopedia of Optimization*; Floudas, C.A., Pardalos, P.M., Eds.; Springer US: Boston, MA, USA, 2008; pp. 1538–1542. ISBN 978-0-387-74758-3.
4. El-Omari, N.K.T. Sea Lion Optimization Algorithm for Solving the Maximum Flow Problem. *Int. J. Comput. Sci. Netw. Secur.* **2020**, *20*, 30–68. [CrossRef]
5. Weir, M.D.; Hass, J.; Giordano, F.R. *Thomas' Calculus*; Pearson Addison Wesley: Boston, MA, USA, 2005; ISBN 978-0-321-18558-7.
6. Bunkley, N. Joseph Juran, 103, Pioneer in Quality Control, Dies. *The New York Times*. Available online: https://www.nytimes.com/2008/03/03/business/03juran.html (accessed on 16 September 2024).
7. Mohanty, R.; Suman, S.; Das, S.K. Modeling the Axial Capacity of Bored Piles Using Multi-Objective Feature Selection, Functional Network and Multivariate Adaptive Regression Spline. In *Handbook of Neural Computation*; Elsevier: Amsterdam, The Netherlands, 2017; pp. 295–309. ISBN 978-0-12-811318-9.
8. Liberti, L.; Kucherenko, S. Comparison of Deterministic and Stochastic Approaches to Global Optimization. *Int. Trans. Oper. Res.* **2005**, *12*, 263–285. [CrossRef]
9. Haggag, S.; Desokey, F.; Ramadan, M. A Cosmological Inflationary Model Using Optimal Control. *Gravit. Cosmol.* **2017**, *23*, 236–239. [CrossRef]
10. Index. In *An Introduction to Algorithmic Finance, Algorithmic Trading and Blockchain*; Emerald Publishing Limited: Leeds, UK, 2020; pp. 185–191. ISBN 978-1-78973-894-0.
11. Sieja, M.; Wach, K. The Use of Evolutionary Algorithms for Optimization in the Modern Entrepreneurial Economy: Interdisciplinary Perspective. *EBER* **2019**, *7*, 117–130. [CrossRef]

12. Guimaraes, G.L.; Sanchez-Lengeling, B.; Outeiral, C.; Farias, P.L.C.; Aspuru-Guzik, A. Objective-Reinforced Generative Adversarial Networks (ORGAN) for Sequence Generation Models. *arXiv* **2017**, arXiv:1705.10843.
13. He, J.; Mattsson, F.; Forsberg, M.; Bjerrum, E.J.; Engkvist, O.; Nittinger, E.; Tyrchan, C.; Czechtizky, W. Transformer Neural Network for Structure Constrained Molecular Optimization. *ChemRxiv* **2021**. [CrossRef]
14. Brownlee, J. Why Optimization Is Important in Machine Learning. 2021. Available online: https://MachineLearningMastery.com (accessed on 16 September 2024).
15. Gandomi, A.H.; Yang, X.-S.; Talatahari, S.; Alavi, A.H. Metaheuristic Algorithms in Modeling and Optimization. In *Metaheuristic Applications in Structures and Infrastructures*; Elsevier: Amsterdam, The Netherlands, 2013; pp. 1–24. ISBN 978-0-12-398364-0.
16. Blum, C.; Puchinger, J.; Raidl, G.; Roli, A. Hybrid Metaheuristics. In *Hybrid Optimization. Springer Optimization and Its Applications*; van Hentenryck, P., Milano, M., Eds.; Springer: New York, NY, USA, 2011; Volume 45. [CrossRef]
17. Burke, E.; Gendreau, M.; Hyde, M.; Kendall, G.; Ochoa, G.; Özcan, E.; Qu, R. Hyper-heuristics: A survey of the state of the art. *J. Oper. Res. Soc.* **2013**, *64*, 1695–1724. [CrossRef]
18. Farag, A.A.; Ali, Z.M.; Zaki, A.M.; Rizk, F.H.; Eid, M.M.; El-Kenawy, E.M. Exploring Optimization Algorithms: A Review of Methods and Applications. *J. Artif. Intell. Metaheuristics* **2024**, *7*, 8–17. [CrossRef]
19. Kumar, S.; Datta, D.; Singh, S.K. Black Hole Algorithm and Its Applications. In *Computational Intelligence Applications in Modeling and Control*; Azar, A.T., Vaidyanathan, S., Eds.; Studies in Computational Intelligence; Springer International Publishing: Cham, Switzerland, 2015; Volume 575, pp. 147–170. ISBN 978-3-319-11016-5.
20. Stützle, T.; López-Ibáñez, M. Automated Design of Metaheuristic Algorithms. In *Handbook of Metaheuristics. International Series in Operations Research & Management Science*; Gendreau, M., Potvin, J.Y., Eds.; Springer: Cham, Switzerland, 2019; Volume 272. [CrossRef]
21. Lambora, A.; Gupta, K.; Chopra, K. Genetic Algorithm—A Literature Review. In Proceedings of the 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon), Faridabad, Indi, 14–16 February 2019; IEEE: Faridabad, India; pp. 380–384.
22. Yang, X.-S. Firefly Algorithms for Multimodal Optimization. In *Stochastic Algorithms: Foundations and Applications*; Watanabe, O., Zeugmann, T., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2009; Volume 5792, pp. 169–178. ISBN 978-3-642-04943-9.
23. Gao, X.Z.; Govindasamy, V.; Xu, H.; Wang, X.; Zenger, K. Harmony Search Method: Theory and Applications. *Comput. Intell. Neurosci.* **2015**. [CrossRef] [PubMed]
24. Jamil, M.; Yang, X.-S. A Literature Survey of Benchmark Functions For Global Optimization Problems. *arXiv* **2013**, arXiv:1308.4008.
25. Kerschke, P.; Trautmann, H. Automated Algorithm Selection on Continuous Black-Box Problems by Combining Exploratory Landscape Analysis and Machine Learning. *Evol. Comput.* **2019**, *27*, 99–127. [CrossRef] [PubMed]
26. Liu, J.; Moreau, A.; Preuss, M.; Rapin, J.; Roziere, B.; Teytaud, F.; Teytaud, O. Versatile black-box optimization. In Proceedings of the 2020 Genetic and Evolutionary Computation Conference (GECCO'20), Cancún, Mexico, 8–12 July 2020; Association for Computing Machinery: New York, NY, USA; pp. 620–628. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.