Enhancing Continuous Optimization with a Hybrid History-Driven Firefly and Simulated Annealing Approach

Sina Alizadeh and Malek Mouhoub[a]
Department of Computer Science, University of Regina, Regina SK, Canada
{*sau564, mouhoubm*}*@@uregina.ca*

Keywords: Feature Landscaping, Metaheuristics, Firefly, Simulated Annealing.

Abstract: In this study, we propose a hybrid History-driven approach through collaboration between Firefly (FA) and Simulated Annealing (SA) algorithms, to improve the hybrid framework performance in finding the global optima in continuous optimization problems in less time. A Self-Adaptive Binary Space Partitioning (SA-BSP) tree is used to partition the search space of a continuous problem and guide the hybrid framework towards the most promising sub-region. To solve the premature convergence challenge of FA a "*Finder – Tracker agents*" mechanism is introduced. The hybrid framework progresses through three main stages. Initially, in the first phase, the SA-BSP tree is utilized within the FA algorithm as a unit of memory. The SA-BSP tree stores significant information of the explored regions of the search space, creates the fitness landscape, and divides the search space during exploration. Moving on to the second phase, a smart controller is introduced to maintain a balance between exploration and exploitation using HdFA and SA. During the third step, the search is limited to the most promising sub-region discovered. Subsequently, the SA algorithm employs the best solution's information, including its fitness value and position, to efficiently exploit the limited search space. The proposed HdFA-SA technique is then compared against different metaheuristics across ten well-known unimodal and multimodal continuous optimization benchmarks. The results demonstrate HdFA-SA's exceptional performance in finding the global optima solution while simultaneously reducing execution time.

1 INTRODUCTION

Solving optimization problems can present significant challenges due to their frequently non-linear nature, existing numerous local optima, and dealing with large search spaces (Zhang et al., 2016). Optimization problems range from scheduling tasks, and balancing loads in telecommunication networks (Jafarian et al., 2014) to Deep Learning Training (Kingma and Ba, 2014), Distributed Data Processing (Zaharia et al., 2012), Cluster Scheduling (Schwarzkopf et al., 2013), and robust combinatorial optimization in the form of minimax optimization(Shao et al., 2022). Many metaheuristics have been proposed over the past decades to find the optimal solution (Talbi, 2009; Korani and Mouhoub, 2021; Hmer and Mouhoub, 2016; Bidar and Mouhoub, 2022).

Metaheuristics represent strategies that are inspired by nature and can be classified as bio-inspired, population-based, or physical and chemical-based approaches. Population-based algorithms include evolutionary methods such as Genetic Algorithms (GAs) (Holland, 1992), and Swarm Intelligence (SI) methods such as Particle Swarm Optimization(PSO) (Kennedy and Eberhart, 1995), Ant Colony Optimizer(ACO) (Dorigo et al., 2006), Artificial Bee Colony (ABC), and the Firefly Algorithm (FA) (Yang, 2009). Physical-chemical-based algorithms include Simulating Annealing (SA)(Kirkpatrick et al., 1983) which is a single solution algorithm that focuses on modifying and improving a single candidate solution. Other single solution algorithms include Iterated Local Search (ILS) (Lourenço et al., 2003), and Variable Neighborhood Search (VNS) (Mladenović and Hansen, 1997). While metaheuristics have resulted in positive outcomes and benefits, they confront challenges such as the inability to discover the global optimum, premature convergence, and lengthy execution times. Moreover, based on the No Free Lunch (NFL) theorem (Wolpert and Macready, 1997), no algorithm can be the best for all optimization problems. The challenges encountered have led researchers to design hybrid cooperative framework strategies. These techniques aim to increase the accuracy of metaheuristics through a collaboration of several algorithms to

[a] https://orcid.org/0000-0001-7381-1064

Alizadeh, S. and Mouhoub, M.
Enhancing Continuous Optimization with a Hybrid History-Driven Firefly and Simulated Annealing Approach.
DOI: 10.5220/0012812900003758

In *Proceedings of the 14th International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH 2024)*, pages 120-127
ISBN: 978-989-758-708-5; ISSN: 2184-2841

develop efficient hybrid methodologies for optimization problems (Blum and Roli, 2008). In the field of hybrid optimization algorithms, numerous combinations of popular optimization methods have been developed, such as CS/PSO by Ghodrati and Lotfi (Ghodrati and Lotfi, 2012), HEPSO (Mahmoodabadi et al., 2014), HAP (Kıran et al., 2012), and HP-CRO (hybrid of PSO and CRO) (Nguyen et al., 2014). Kao and Zahara applied a combination of GAs and PSO to solve 17 test problems (Kao and Zahara, 2008). They observed that despite this strategy requiring more time for extensive function evaluations, it resulted in fewer average errors when compared with (Chelouah and Siarry, 2000) and CHA(Chelouah and Siarry, 2003) approaches. Zhang and Li developed a hybrid algorithm HP-CRO, based on a combination of PSO with chemical reaction optimization (Nguyen et al., 2014). They utilized this method to solve 30-dimensional Rosenbrock and Schwefel 2.22 functions, attaining superior outcomes relative to the RCCRO algorithm, with errors diminishing to as minimal as $5.52E+00$ and $4.22E-06$. Farnad et al. (Farnad et al., 2018), developed a hybrid algorithm combining PSO, GA, and SOS for improved global search in complex search spaces. Their HPG-SOS method optimized functions more rapidly, and they introduced a new PSO variant, HEPSO, outperforming other algorithms (Mahmoodabadi et al., 2014). In a previous work, we introduced a self-adaptive hybrid framework to enhance the cooperation between PSO and SA based on a self-adaptive BSP tree and maturity condition to find the best iteration to switch from HdPSO to SA. The proposed HdPSO-SA was tested on 10 well-known continuous benchmarks. The findings demonstrated that the hybrid HdPSO-SA algorithm achieved superior performance by identifying the global optima more efficiently with less running time than traditional metaheuristics, including FA, PSO, DE, SA, and GA (Alizadeh and Mouhoub, 2023). Following on this previous research about cooperation between heuristics based on a history-driven method, in this study, we will show how the proposed Self-Adaptive BSP tree approach is applicable in different metaheuristics to enhance the collaboration between them and finding the global optima in less time. In this study, we use FA instead of PSO as the exploration algorithm due to the method's efficiency in exploration and SA for exploitation. According to the negative effect of premature challenge on metaheuristics especially in FA and potentially increasing the running time due to less effective exploration of the solution space (Khan et al., 2016), In this context, we propose an approach called "$Finder-Tracker$ agents" to deal with this challenge. Besides, we mention the destructive effects of this challenge on the proposed SA-BSP tree and the approximated fitness landscape. An innovation of this work is the proposed "$Finder-Tracker$ agents" method, which maintains population diversity during the exploration and increases the accuracy of the SA-BSP tree in finding the most promising subregion in the given search space.

2 BACKGROUND

Binary Space Partitioning (BSP) tree is a basic data structure introduced in computer graphics and computational geometry, utilized to enhance the efficiency of metaheuristic algorithms through a historical data-driven approach (Yuen and Chow, 2008). In continuous optimization problems, the Binary Space Partitioning (BSP) tree is employed to partition the search space S. This method additionally keeps track of the valuable gathered information during the exploration of search space and the position and fitness value of newly generated solution $[s_i, f(x_i)]$, acting as a storage component for metaheuristics. In (Alizadeh and Mouhoub, 2023), we indicated that the fitness approximation method based on the BSP tree has two limitations. Consequently, HdPSO calculates the fitness values for all solutions generated, both in unpromising and the most promising sub-regions. implementing the BSP tree to identify the most promising region before switching from HdPSO to SA leads to computing all generated solutions, which causes extra partitioning (unnecessary nodes), extra running time and a deep BSP tree with lower accuracy. To address BSP tree limitations, we offered a novel Self-adaptive BSP tree for an improved fitness landscape prediction and node insertion technique. Besides, a novel approach to continuous optimization through the development of the hybrid HdPSO-SA method and the incorporation of a self-adaptive Binary Space Partitioning (BSP) tree to guide the HdPSO-SA algorithm to identify the most promising subregion introduced. This self-adaptive mechanism, referred to as the SA-BSP tree, serves a dual function: it partitions the search space into manageable sub-regions and stores critical information about these segments, such as fitness values and their spatial coordinates. Initially, the hybrid model starts with limited information about the search space. However, as HdPSO progresses, accumulating data in its long-term memory, the framework's knowledge of the search space expands significantly. The maturation of the SA-BSP tree is crucial for enhancing the precision of the fitness landscape evaluation. A primary focus of HdPSO within this hybrid model is to explore the search space, continuously up-

dating the SA-BSP tree with new fitness values and positional information to accurately form the fitness landscape. This process is instrumental in guiding the algorithm towards promising sub-regions, thereby optimizing the efficiency of the search by avoiding less promising areas. This strategic approach not only enhances the search efficiency but also significantly reduces the computational time required. The exploration-exploitation (E-E) trade-off, a fundamental challenge in optimization algorithms, is adeptly managed within this framework by the introduction of a smart maturity condition, denoted as α_{ave}. Upon meeting this maturity criterion, the algorithm transitions to a phase where the optimal solution, along with its specific sub-region coordinates, is passed for exploitation by the Simulated Annealing (SA) process. This phase limits the search to a specific sub-region, leveraging the detailed coordinates provided by the SA-BSP tree to achieve significant exploitation. Furthermore, the publication explores the transformation process of the BSP tree into a self-adaptive entity through the development of the innovative SA-BSP maturity condition. This transition is pivotal for the approximation of fitness values, aimed at enhancing time efficiency without compromising the quality of the search results. The steps involved in the implementation of the HdPSO-SA method are outlined in detail, highlighting the innovations and the results in 10 well-known continuous benchmarks (Alizadeh and Mouhoub, 2023).

3 HISTORY-DRIVEN FIREFLY-SIMULATED ANNEALING (HDFA-SA)

The primary purpose of this study is to demonstrate how the proposed SA-BSP tree in (Alizadeh and Mouhoub, 2023) applies to other metaheuristics (Firefly in this study). Another aspect of this study is to show the negative effects of premature convergence on the proposed SA-BSP tree. To achieve our objectives, we utilized the Firefly Algorithm (FA) because of its proficiency in global search optimization and Simulated Annealing (SA) for its effectiveness in exploitation tasks. Since premature convergence is the main cause of the FA algorithm getting stuck in local optima a method known as "*Finder_Tracker agents*" is introduced to manage premature convergence. In the subsequent sections, we delve into the steps of HdFA-SA, as illustrated in Figure 1. Following this, we examine the adverse impacts of premature convergence on the SA-BSP tree utilizing the provided

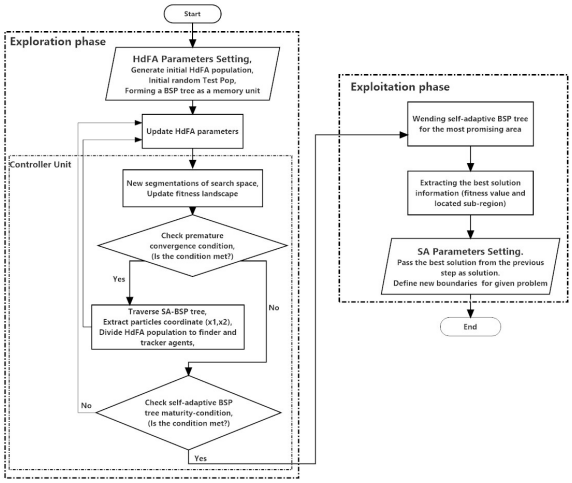

Figure 1: HdFA-SA flowchart.

multi-modal Michalewicz function. Subsequently, we introduce the 'Finder Tracker Agents' approach. Finally, we present the outcomes of HdFA-SA across 10 continuous benchmarks.

Figure 1 shows the hybrid HdFA-SA approach using a self-adaptive BSP tree as HdFA's memory to accumulate search progress benefited "*Finder_Tracker agents*" approach. Figure 1 describes HdFAb integrated with a self-adaptive Binary Space Partitioning (BSP) tree structure and SA. The process is divided into two main phases: exploration and exploitation. During the exploration phase, the algorithm sets up the parameters for the HdFA and generates an initial population alongside an SA-BSP tree to serve as a memory unit. This is a dynamic phase where the algorithm continuously updates the parameters and segments the search space to create the fitness landscape and approximate the fitness values for the next iterations. It checks for premature convergence, which could indicate that the algorithm is not exploring the search space adequately and is instead settling on local optima. If this condition is met, the algorithm proceeds to navigate through the HdFA, adjusting its population into finder and tracker agents to diversify the search. This process is iterative and continues until a maturity condition for the self-adaptive BSP tree is satisfied, suggesting that the exploration has been thorough. In the exploitation phase, the focus shifts to intensifying the search around the most promising area identified in the exploration phase. The algorithm focuses on the best solution identified, as determined by the fitness value and a specific sub-region of the search space. At this stage, the parameters for SA are configured, leveraging the best solution obtained from the exploration as a starting point for further exploitation. New boundaries for the given problem are

established based on the findings, which means limiting the search to only the most promising subregion that is explored from the SA-BSP tree.

To show the ability of the SA-BSP tree (proposed in (Alizadeh and Mouhoub, 2023)) to enhance the collaboration between metaheuristics, in this study, we use FA instead of PSO for the exploration phase. The proposed HdFA-SA (algorithm 1) is based on the behavior of fireflies attracted to each other's brightness, combined with simulated annealing to exploit the limited subregion only. In contrast, the HdPSO-SA uses particle swarm principles, guiding particles towards optimal solutions based on individual and collective experience, also enhanced by simulated annealing. Another difference between HdFA-SA and HdPSO-SA is the HdFA-SA algorithm is equipped with "$Finder-Tracker$ agents" (algorithm 2)to prevent early convergence if happens as shown as an update in figure1. In other words, In this study "$Finder-Tracker$ agents" approach is added to the controller unit of HdFA-SA figure 1 and is used to update the Tracker agents differently if the condition τ is met.

Algorithm 1 shows the proposed HdFA-SA algorithm starts by taking input parameters based on provided steps in figure 1, including SA-BSP tree parameters size pop_{test}, HdFA fireflies (pop_{real}), number of neighbors n, number of HdFA and SA iterations, HdFA parameters (γ, β_0), and SA parameters (T, k, c). From lines 2 to 10, the algorithm applies the exploration steps based on HdFA and updates the formula. In line 10 The SA-BSP tree is traversed for every generated new solution s_i to extract the location of s_i. line 12 indicates that If the extracted sub-region includes the lowest fitness value, it will be partitioned and the new solution's exact fitness value will be calculated. Otherwise, the region is not divided into new segments and the previous fitness value of the segment is assigned to s_i. α_i represents the difference of two values estimated fitness of sub-region h_j extracted from the BSP tree for m_i and the actual fitness of m_i. Lines 24 - 26 determine that the maturity condition will be met if α_{ave} has a downward trend or remains stable for l iterations or reaches zero. α_i and α_{ave} are calculated for each iteration. According to maturity condition α_{ave}, if the SA-BSP tree has matured, then the algorithm will switch from HdFA to SA, and the fitness value and position of a solution s and the coordinates of a sub-region containing solution s are returned (lines 27 - 29). once the maturity condition is met, The algorithm passes the HdFA best solution to SA and limits the search space to the sub-region containing the best solution s. If the SA-BSP tree is immature, the algorithm iterates for better

Algorithm 1: Pseudocode of HdFA-SA Algorithm.

1: **Input:** SA-BSP tree as memory, size pop_{test}, number of fireflies (pop_{real}), HdFA iterations, light absorption coefficient γ, attractiveness β_0, SA iterations, reduction factor c, temperature T, Boltzmann's constant k,
2: Initialize pop_{test} and HdFA fireflies pop_{real} with random positions the SA-BSP tree maturity condition α_{ave} is not met $i = 1$; $i \leq pop_{real}$ $j = 1$;$i \leq pop_{real}$ $f(s_i) < f(s_j)$
3: Calculate attractiveness $\beta = \beta_0 \cdot \exp(-\gamma \cdot |s_i - s_j|^2)$
4: Move firefly i towards j based on attractiveness and randomness
5: $s_i = s_i + \beta \cdot (s_j - s_i) + \alpha \cdot (\text{rand} - 0.5)$
6: Evaluate new solutions and update light intensity
7: Traverse SA-BSP tree and extract the s_i sub-regions s_i is located in the most promising region among n neighbors
8: Accept s_i and evaluate objective function $f(s_i)$
9: Segment sub-region h_i
10: Update the SA-BSP tree child and parent nodes information $f(s_i) < \tilde{f}(s_i)$
11: Update the value of $f(s_i)$ for $\tilde{f}(s_i)$
12: Reject s_i and avoid evaluating objective function $f(s_i)$
13: Accept the value of $\tilde{f}(s_i)$ for $f(s_i)$
14: No segmentation
15: $\alpha_i = |f(m_{i_{Pop\_test}}) - \tilde{f}(m_{i_{Pop\_test}})|$
16: $\alpha_{ave} = \frac{\sum_1^i(\alpha)}{i}$ α_{ave} has a downward continual trend for l iterations or
17: α_{ave} remains stable in l iterations or
18: $\alpha_{ave} = 0$
19: BSP tree maturity condition is set to True
20: Return fitness value and position of solution s by traversing BSP tree
21: Return the coordinates of the sub-region containing solution s
22: Go to step 3
23: $iter = iter + 1$
24: Pass the optimal solution s position to SA for exploitation.
25: Limit the search space to the most promising sub-region.
SA iteration number is not reached
solution s
$f(s + \Delta x) > f(s)$
26: $f_{new} = f(s + \Delta x)$; $s = s + \Delta x$
27: $\Delta f = f(s + \Delta x) - f(s)$
28: random $r \in (0,1)$ $r > exp(-\Delta f / kT)$
29: $f_{new} = f(s + \Delta x)$, $s = s + \Delta x$
30: $f_{new=f(s)}$
31: **end For**
32: $f = f_{new}$
33: $T = c \times T$ (Decrease the temperature periodically)
34: **end While**

solutions and sub-regions. SA employs the solution s and starts the search progress with a probability of accepting a less optimal solution to address the local optima challenge. Lines 35 to 48 indicate the SA method to exploit the limited search space. (Rere et al., 2015)

3.1 Premature Convergence and Finder-Tracker Agents Method

Premature convergence represents a significant challenge in metaheuristic algorithms. These algorithms may suffer from premature convergence and poor global exploration when implemented to optimize

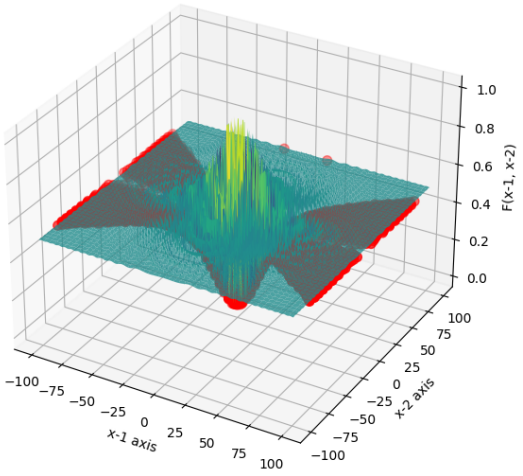

Figure 2: 3-D Schaffer function.

complex and high-dimension continuous optimization problems.

To show the negative impacts of premature convergence in multi-local problems we utilize the Schaffer function as an example. Figure 2 shows the Schaffer function (f_4 in Table 1 of (Alizadeh and Mouhoub, 2023)), particularly with the highlighting of local optima (or traps) with red. The red points are where the function value is lower compared to the immediate surroundings, making them "traps" for optimization algorithms. In a minimization problem, metaheuristic algorithms might converge to these points believing in reaching the global optima. While the Schaffer function is a theoretical construct used primarily for benchmarking and testing, its challenges reflect real-world optimization problems in various fields such as engineering design, financial modeling, logistics, and machine learning. The Firefly algorithm might suffer from premature convergence when the members settle into a local optimum due to the very close fitness values.

In this work, early convergence severely limits the Firefly Algorithm's (FA) exploration capabilities, resulting in an incomplete search history and an underdeveloped Binary Space Partitioning (BSP) tree. This constraint not only prevents the algorithm from thoroughly exploring viable solutions but also reduces the BSP tree's ability to predict the best exploration-exploitation (E-E) trade-off point between FA and Simulated Annealing (SA). Addressing premature convergence is critical for improving the optimization process, guaranteeing a balanced E-E trade-off, and generating optimal solutions that properly leverage both FA and SA characteristics.

Different methods have been developed to address the early convergence challenge. A common approach involves restarting the population upon detecting early convergence. However, this method has critical drawbacks. Firstly, regenerating the population will erode valuable search history, like the gathered information about the most promising subregions. Secondly, using this method leads to missing the created fitness landscape. Therefore, storing crucial information before resetting the FA's population is recommended. An approach named "Finder-Tracker agents" is proposed in this study to assign distinct roles within the HdFA population to efficiently counter the problem of early convergence with an approach of storing the valuable information of the best members. *Finder_Tracker agents* mechanism will divide and label the particles of the applied HdFA into two different types of particles with different tasks if premature convergence happens. A portion of the population is labelled as *finder*, while another is labelled as a *tracker*. In this approach for each iteration, the average Euclidean distance between particles located in the n regions is calculated and compared to a predefined threshold (τ). τ represents a threshold value. The choice of τ is critical; it must be set appropriately to balance the convergence speed against the risk of missing the global optimum. Line 9 of Algorithm 2, illustrates the mechanism by which the algorithm detects premature convergence, by comparing the average of calculated Euclidean distance to a threshold τ, which it concludes early convergence is happening. If convergence occurs and the convergence condition is met, the particles that store the best solutions will be labelled as trackers, and the rest labelled as finders. The tracker solutions will be saved, and the rest of the solutions that are finders will be regenerated. By making new finder solutions, the algorithm can explore different regions and find a variety of different solutions.

Algorithm 2: Finder-Tracker Agents Algorithm.

1: **Initialize**:
2: Set distance threshold τ,
3: HdFA Population = $Particles_{Finder} + Particles_{Tracker}$,
4: $Particles_{Finder} \geq Particles_{Tracker}$.
5: **Main Loop**: m Iterations
6: Travers SA-BSP tee $particle_1$ to $particle_i$ located in same region $\frac{\sum_1^i (Euclideandistance)}{i} < \tau$
7: Flag convergence condition
8: Label HdFA particles with the best fitness values as *tracker*.
9: Label the remaining particle as *finder*
10: Regenerate *finder* particles.
11: **end for**
12: **end for**
13: end for

Figure 3 shows the E-E trade-off between HdFA and SA from epoch 1 to 4 on the 2d Schaffer function. The blue Crosses are HdFA particles and SA

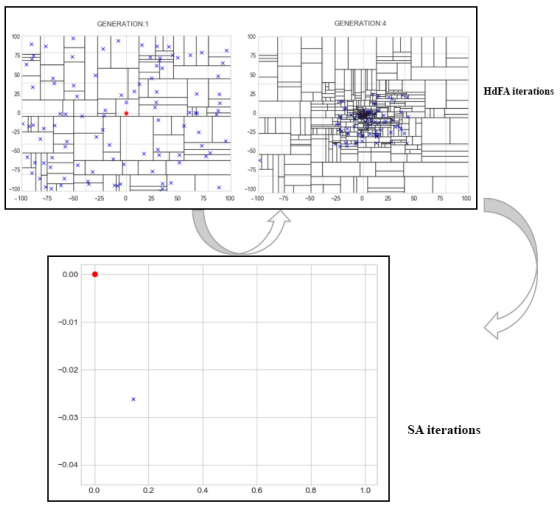

Figure 3: Switching from HdFA to SA for Schaffer function.

solution at the top and below figures, respectively, and the red dot is the global optima. During the exploration process, the HdFA algorithm could find the best local value. Furthermore, using the BSP tree memory mechanism, the valuable information was saved and passed to the SA for the remainder of the search process. After checking premature and maturity conditions (controller unit in figure 1), the search space limited from S= $[x_1, x_2] \in [-100, 100] \times [-100, 100]$ in exploration phase to sub-region h_i = $[x_1, x_2] \in [-0.05, 1.05] \times [-0.044, 0.002]$ for exploitation phase. In this example, the number of solutions of HdFA equals 100 (the blue crosses). Applying a self-adaptive E-E trade-off based on history driven, the algorithm will be successful in limiting the search space S to the sub-region h_i with the best chance of finding the global optima solution. In other words, sub-region h_i is a leaf node of the SA-BSP tree with the best fitness value.

4 EXPERIMENTATION

4.1 Experimentation Environment

To evaluate HdFA-SA, benchmark test functions from (Alizadeh and Mouhoub, 2023) are employed that list the continuous functions, their search ranges, and global optima. Functions $f_1 - f_3$ are unimodal and non-separable. $f_4 - f_{10}$ are multimodal functions, in which $f_4 - f_8$ are non-separable while f_9 and f_{10} are separable. Experiments run on Python 3.7, an Intel Core i7 CPU (2.00 GHz) and 16 GB RAM.

4.2 Comparison with Other Nature-Inspired Techniques

Table 1 compares the performance of different history-driven methods HdFA-SA and HdPSO-SA with the traditional methods FA, GA, PSO, and SA across ten different unimodal and multimodal test functions $f_1 - f_{10}$. The performance is measured in terms of the "Best" fitness value achieved by the algorithms and the running time(s). The population size for all the methods is equal to 100 and SA is equal to 1. For HdFA $\alpha = 0.2$, $\gamma = 1$, and $\beta_0 = 1$ and for SA, $T = 100$, $k = 0.8$, and $c = 0.95$. The other requirement parameters for HdFA-SA are $Pop_{test} = 300$, $l >= 3$, and $n = 3$. For the rest of the algorithms, we tune the parameters to their best values. For PSO, $w = 0.9$, $c_1 = 2.05$, and $c_2 = 2.05$, For GA, the mutation probability is 0.05 and the crossover probability is 0.95. For the HdPSO-SA and other measures refer to (Alizadeh and Mouhoub, 2023).

As can be seen from table 1, two history-driven approaches HdPSO-SA and HdFA-SA have better results not only in finding the global optima but also in running time compared with the other algorithms that shows the ability of SA-BSP tree in collaboration between metaheuristics and guiding the hybrid framework to exploit the most promising subregion based on fitness landscape approximation. PSO and SA can find the global optima in only five and one benchmark problems, respectively. GA exclusively finds the global optimum for the function f_3. FA exhibits a competitive performance in locating the global optima, though with a longer computational time than HdFA-SA and HdPSO-SA are needed. HdFA-SA could address the most complex functions with many local optima like Michalewicz2 (f_{10}), and Schaffer (f_4) functions with the "$Finder - Tracker$ agents" algorithm to deal with premature convergence to find the global optima in lower time compared with FA. Table 1 illustrates that HdPSO-SA successfully located the global optimum for the Easom function (f_2), a feat not achieved by either HdFA-SA or FA. Furthermore, while FA successfully located the global optima for all benchmark functions except for f_2, HdFA-SA achieved these results more efficiently, in terms of computational time. This improvement was made possible by employing the fitness landscape approximation strategy introduced in this study, which enhanced the search efficiency across all tested functions, excluding f_2.

Table 1: HdFA-SA comparison with HdPSO-SA, FA, GA, PSO, and SA with the running time(s) of algorithms.

		HdFA-SA	HdPSO-SA	FA	GA	PSO	SA
f_1	Best	**0**	**0**	0	0.75	0	9.3593E-07
	Time(s)	**0.62**	**0.75**	32.16	2.31	0.90	0.03
f_2	Best	-0.999	**-1**	-0.9997	0	0.9999	0
	Time(s)	10	**0.75**	17.61	17.96	1.11	0.007
f_3	Best	**0**	**0**	0	0.01	0	22.65
	Time(s)	**0.28**	**0.43**	8.46	9.6	0.78	0.008
f_4	Best	**0**	**0**	0	0.03	0	0.3854
	Time(s)	**0.21**	**0.72**	158.8	2.64	1.03	0.007
f_5	Best	**-1.0316**	**-1.0316**	-1.0316	0.07	-1.0316	-1.0316
	Time(s)	**0.46**	**0.41**	0.94	2.36	0.79	0.42
f_6	Best	**0**	**0**	0	0.03	2.16	3510.20512
	Time(s)	**0.76**	**0.65**	41.66	2.35	2.1697E-07	0.007
f_7	Best	**0**	**0**	0	0.03	3.5393E-05	12860.23353
	Time(s)	**0.72**	**0.44**	25.55	2.38	1.25	0.007
f_8	Best	**-186.73**	**-186.73**	-186.73	0.0004	-177.6542	-8.5682
	Time(s)	**4.68**	**1.09**	70.91	2.35	5.38	0.008
f_9	Best	**0**	**0**	0	0.05	0	7652.4848
	Time(s)	**0.9**	**0.64**	17.46	2.39	2.09	0.008
f_{10}	Best	**-1.8013**	**-1.8013**	-1.8013	0.0004	-1.8012	-1.8012
	Time(s)	**0.7**	**1.1**	2.7	2.38	1.88	0.19

5 CONCLUSIONS

We propose a cooperative hybrid approach based on a history-driven method, namely HdFA-SA, for creating feature landscapes and efficiently finding the global optima in continuous optimization problems. A self-adaptive BSP tree is used to store valuable information about the search space to create the landscape of approximated fitness values and to partition the search space accordingly during the exploration phase. HdFA and SA are implemented for exploration and exploitation, respectively. Since early convergence leads to a deep SA-BSP tree and an inaccurate fitness landscape, the HdFA is equipped with a "$Finder - Tracker$ agents" approach in its controller unit compared with the previous study (HdPSO-SA) to identify and deal with this challenge. Finally, a smart controlling mechanism is implemented in HdFA-SA for determining the best time (iteration) for switching from HdFA to SA (following the E-E tradeoff) to take advantage of the strengths of both algorithms. Besides, to decrease the running time in exploitation steps, the search space is limited to only the most promising subregion. The "$Finder - Tracker$ agents" approach is proposed to maintain population diversity in the face of early convergence while the gathered valuable data during the exploration iterations will be stored. We evaluate the proposed method on 10 unimodal and multimodal continuous benchmarks and compare the results with state-of-the-art metaheuristics. The results make it clear that for nine out of the benchmarks, HdFA-SA located the global optima faster than the traditional methods. The comparison illustrates both hybrid methods HdPSO-SA and HdFA-SA could find the global optima in 10 and 9 continuous benchmarks, respectively with less running time compared with the other methods. Fitness landscape approximation is an aspect of the research that has a crucial role in decreasing the running time of both hybrid approaches HdFA-SA and HdPSO-SA.

REFERENCES

Alizadeh, S. and Mouhoub, M. (2023). A new self-adaptive hybrid approach based on history-driven methods for improving metaheuristics. In *Proceedings of the IEEE 2023 International Conference on Machine Learning and Applications (ICMLA 2023), Jacksonville, FL, USA, December 15-17, 2023*, pages 762–7670. IEEE.

Bidar, M. and Mouhoub, M. (2022). Nature-inspired techniques for dynamic constraint satisfaction problems. *Operations Research Forum*, 3(2):1–33.

Blum, C. and Roli, A. (2008). Hybrid metaheuristics: an introduction. In *Hybrid metaheuristics*, pages 1–30. Springer.

Chelouah, R. and Siarry, P. (2000). A continuous genetic algorithm designed for the global optimization of multimodal functions. *Journal of Heuristics*, 6(2):191–213.

Chelouah, R. and Siarry, P. (2003). Genetic and nelder–mead algorithms hybridized for a more accurate global optimization of continuous multiminima functions. *European Journal of operational research*, 148(2):335–348.

Dorigo, M., Birattari, M., and Stutzle, T. (2006). Ant colony optimization. *IEEE computational intelligence magazine*, 1(4):28–39.

Farnad, B., Jafarian, A., and Baleanu, D. (2018). A new hybrid algorithm for continuous optimization problem. *Applied Mathematical Modelling*, 55:652–673.

Ghodrati, A. and Lotfi, S. (2012). A hybrid cs/pso algorithm for global optimization. In *Intelligent Information and Database Systems: 4th Asian Conference, ACIIDS 2012, Kaohsiung, Taiwan, March 19-21, 2012, Proceedings, Part III 4*, pages 89–98. Springer.

Hmer, A. and Mouhoub, M. (2016). A multi-phase hybrid metaheuristics approach for the exam timetabling. *International Journal of Computational Intelligence and Applications*, 15(04):1–22.

Holland, J. H. (1992). Genetic algorithms. *Scientific american*, 267(1):66–73.

Jafarian, A., Measoomy, S., Golmankhaneh, A. K., and Baleanu, D. (2014). A numerical solution of the urysohn-type fredholm integral equations. *Rom. J. Phys*, 59(7-8):625–635.

Kao, Y.-T. and Zahara, E. (2008). A hybrid genetic algorithm and particle swarm optimization for multimodal functions. *Applied soft computing*, 8(2):849–857.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, volume 4, pages 1942–1948. IEEE.

Khan, W. A., Hamadneh, N. N., Tilahun, S. L., and Ngnotchouye, J. (2016). A review and comparative study of firefly algorithm and its modified versions. *Optimization Algorithms-Methods and Applications*, 45:281–313.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kıran, M. S., Gündüz, M., and Baykan, Ö. K. (2012). A novel hybrid algorithm based on particle swarm and ant colony optimization for finding the global minimum. *Applied Mathematics and Computation*, 219(4):1515–1521.

Kirkpatrick, S., Gelatt Jr, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *science*, 220(4598):671–680.

Korani, W. and Mouhoub, M. (2021). Review on nature-inspired algorithms. In *Operations research forum*, volume 2, pages 1–26. Springer.

Lourenço, H. R., Martin, O. C., and Stützle, T. (2003). Iterated local search. In *Handbook of metaheuristics*, pages 320–353. Springer.

Mahmoodabadi, M. J., Mottaghi, Z. S., and Bagheri, A. (2014). Hepso: high exploration particle swarm optimization. *Information Sciences*, 273:101–111.

Mladenović, N. and Hansen, P. (1997). Variable neighborhood search. *Computers & operations research*, 24(11):1097–1100.

Nguyen, T. T., Li, Z., Zhang, S., and Truong, T. K. (2014). A hybrid algorithm based on particle swarm and chemical reaction optimization. *Expert systems with applications*, 41(5):2134–2143.

Rere, L. R., Fanany, M. I., and Arymurthy, A. M. (2015). Simulated annealing algorithm for deep learning. *Procedia Computer Science*, 72:137–144.

Schwarzkopf, M., Konwinski, A., Abd-El-Malek, M., and Wilkes, J. (2013). Omega: flexible, scalable schedulers for large compute clusters. In *Proceedings of the 8th ACM European Conference on Computer Systems*, pages 351–364.

Shao, Z., Yang, J., Shen, C., and Ren, S. (2022). Learning for robust combinatorial optimization: Algorithm and application. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*, pages 930–939. IEEE.

Talbi, E.-G. (2009). *Metaheuristics: from design to implementation*, volume 74. John Wiley & Sons.

Wolpert, D. H. and Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82.

Yang, X.-S. (2009). Firefly algorithms for multimodal optimization. In *International symposium on stochastic algorithms*, pages 169–178. Springer.

Yuen, S. Y. and Chow, C. K. (2008). A genetic algorithm that adaptively mutates and never revisits. *IEEE transactions on evolutionary computation*, 13(2):454–472.

Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauly, M., Franklin, M. J., Shenker, S., and Stoica, I. (2012). Resilient distributed datasets: A {Fault-Tolerant} abstraction for {In-Memory} cluster computing. In *9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*, pages 15–28.

Zhang, L., Liu, L., Yang, X.-S., and Dai, Y. (2016). A novel hybrid firefly algorithm for global optimization. *PloS one*, 11(9):e0163230.