applied sciences

IMPACT FACTOR 2.838

CITESCORE 3.7

Article

A Spider Monkey Optimization Based on Beta-Hill Climbing Optimizer for Unmanned Combat Aerial Vehicle (UCAV)

Fouad Allouani, Abdelaziz Abboudi, Xiao-Zhi Gao, Sofiane Bououden, Ilyes Boulkaibet, Nadhira Khezami and Fatma Lajmi

MDPI

https://doi.org/10.3390/app13053273

applied sciences

MDPI

Article

A Spider Monkey Optimization Based on Beta-Hill Climbing Optimizer for Unmanned Combat Aerial Vehicle (UCAV)

Fouad Allouani [1,*], Abdelaziz Abboudi [2], Xiao-Zhi Gao [3], Sofiane Bououden [1], Ilyes Boulkaibet [4], Nadhira Khezami [4] and Fatma Lajmi [5]

[1] Laboratory of SATIT, Department of Industrial Engineering, Abbes Laghrour University, Khenchela 40004, Algeria
[2] Department of Mechanical Engineering, Abbes Laghrour University, Khenchela 40004, Algeria
[3] School of Computing, University of Eastern Finland, 70210 Kuopio, Finland
[4] College of Engineering and Technology, American University of the Middle East, Egaila 54200, Kuwait
[5] ENISO Laboratory: Networked Objectives, Control, and Communication Systems, National Engineering School of Sousse, Sousse 4023, Tunisia
* Correspondence: allouani_fouad@univ-khenchela.dz

Abstract: Unmanned Combat Aerial Vehicle (UCAV) path planning is a challenging optimization problem that seeks the optimal or near-optimal flight path for military operations. The problem is further complicated by the need to operate in a complex battlefield environment with minimal military risk and fewer constraints. To address these challenges, highly sophisticated control methods are required, and Swarm Intelligence (SI) algorithms have proven to be one of the most effective approaches. In this context, a study has been conducted to improve the existing Spider Monkey Optimization (SMO) algorithm by integrating a new explorative local search algorithm called Beta-Hill Climbing Optimizer (BHC) into the three main phases of SMO. The result is a novel SMO variant called SMOBHC, which offers improved performance in terms of intensification, exploration, avoiding local minima, and convergence speed. Specifically, BHC is integrated into the main SMO algorithmic structure for three purposes: to improve the new Spider Monkey solution generated in the SMO Local Leader Phase (LLP), to enhance the new Spider Monkey solution produced in the SMO Global Leader Phase (GLP), and to update the positions of all Local Leader members of each local group under a specific condition in the SMO Local Leader Decision (LLD) phase. To demonstrate the effectiveness of the proposed algorithm, SMOBHC is applied to UCAV path planning in 2D space on three different complex battlefields with ten, thirty, and twenty randomly distributed threats under various conditions. Experimental results show that SMOBHC outperforms the original SMO algorithm and a large set of twenty-six powerful and recent evolutionary algorithms. The proposed method shows better results in terms of the best, worst, mean, and standard deviation outcomes obtained from twenty independent runs on small-scale (D = 30), medium-scale (D = 60), and large-scale (D = 90) battlefields. Statistically, SMOBHC performs better on the three battlefields, except in the case of SMO, where there is no significant difference between them. Overall, the proposed SMO variant significantly improves the obstacle avoidance capability of the SMO algorithm and enhances the stability of the final results. The study provides an effective approach to UCAV path planning that can be useful in military operations with complex battlefield environments.

Keywords: unmanned combat aerial vehicle (UCAV); path planning; spider monkey optimization (SMO); beta-hill climbing optimizer (BHC)

Citation: Allouani, F.; Abboudi, A.; Gao, X.-Z.; Bououden, S.; Boulkaibet, I.; Khezami, N.; Lajmi, F. A Spider Monkey Optimization Based on Beta-Hill Climbing Optimizer for Unmanned Combat Aerial Vehicle (UCAV). *Appl. Sci.* **2023**, *13*, 3273. https://doi.org/10.3390/app13053273

Academic Editors: Yosoon Choi and Dimitris Mourtzis

Received: 14 January 2023
Revised: 20 February 2023
Accepted: 27 February 2023
Published: 3 March 2023

1. Introduction

Unmanned Aerial Vehicles (UAVs) have been increasingly used for a wide range of applications, including surveillance, search and rescue, and delivery. A crucial aspect of UAV operations is path planning, which involves finding the optimal path for the UAV to achieve its objectives. The Unmanned Combat Aerial Vehicle (UCAV) is a specific type of

UAV that is designed for combat missions. However, path planning for UCAVs presents unique challenges compared to other UAVs, mainly due to their operation in a contested environment where enemy air defense systems pose significant threats. Effective path planning for UCAVs involves minimizing exposure to these threats by considering enemy air defense systems and planning the route accordingly.

UCAVs are equipped with a variety of sensors, including radar and cameras, that assist in the path planning process. The development of advanced path planning algorithms for UCAVs is critical to ensuring their effective use in military operations. Navigation systems-based planning and optimization techniques-based planning are two widely used approaches to UAV path planning.

Navigation systems-based planning involves using existing maps and sensor data, such as GPS and LIDAR, to plan the UAV's path. This approach is typically used in applications where the UAV has a pre-existing map of the environment and the primary goal is to navigate from point A to point B. Navigation systems-based planning algorithms use various techniques, such as A* search, Dijkstra's algorithm, and RRT, to generate the path that the UAV will follow. Although this approach is limited by the accuracy of the maps and sensor data, it is more accessible for beginners and those without extensive programming or mathematical backgrounds [1,2].

Optimization techniques-based planning, on the other hand, uses mathematical optimization to generate the most efficient path for the UAV. This approach involves developing a mathematical model that considers various constraints, such as the UAV's maximum speed, minimum turning radius, and minimum altitude. Common optimization algorithms used for UAV path planning include Genetic Algorithms (GAs), Particle Swarm Optimization (PSO), and Simulated Annealing (SA). Although this approach can generate more efficient paths than navigation systems-based planning, it is computationally expensive, which can make it challenging to implement in real-time applications [3,4].

The choice of approach in path planning depends on the specific application and the available resources. Navigation systems-based planning may be more appropriate for simpler environments with accurate maps and sensor data, while optimization techniques-based planning may be better for more complex environments. Regardless of the approach chosen, path planning is a critical component of UAV operations and can significantly impact the effectiveness of the mission.

In recent years, optimization algorithms based on natural phenomena have been extensively studied and developed to solve various problems in science and engineering. The Spider Monkey Optimization (SMO) algorithm, proposed by Bansal et al. [5], is one such optimization algorithm based on the intelligent foraging behavior of animals with Fission–Fusion social structures. The SMO algorithm has shown promising results in solving optimization problems in various fields, including the Unmanned Combat Aerial Vehicle (UCAV) path planning problem in 2D space.

The SMO algorithm uses the concept of population splitting or merging depending on the search situation, which is widely used in various fields. To handle the UCAV path planning problem with obstacle avoidance, Zhu et al. [6] proposed a Cooperative Co-Evolution (CE) based Spider Monkey Optimization algorithm (CESMO) that was tested and compared with ten other Swarm Intelligence (SI) algorithms using thirty-six test cases. The results showed that CESMO was robust and had a competitive performance for UCAV path planning. In another study, Zhu et al. [7] compared the basic version of SMO with eleven other SI algorithms for UCAV path planning in a complex 2D environment. The results showed that SMO had better performance in finding a safe path compared to the other algorithms.

However, there is always room for improvement, and to enhance the performance of the SMO algorithm, an improved version called SMOBHC has been proposed. The main contributions of this study are threefold. Firstly, the Beta-Hill Climbing Optimizer (BHC), a new exploratory local search algorithm, has been integrated into the three main phases of the SMO algorithm to create SMOBHC. This new algorithm has the advantage

of being better than the SMO in terms of intensification, exploration, avoidance of local minima, and speed of convergence. The BHC is implemented in the SMO algorithm for three main purposes: to improve each new Spider Monkey solution generated in the Local Leader Phase (LLP), to enhance each new Spider Monkey solution produced in the Global Leader Phase (GLP), and finally to update the positions of all Local Leader members under a specific condition in the Local Leader Decision (LLD) phase.

Secondly, the SMOBHC algorithm has been adapted and applied to generate an optimal path for the UCAV in 2D space in complex battlefields under different conditions. Finally, to demonstrate the effectiveness and efficiency of the proposed algorithm, SMOBHC has been compared with the SMO and a large set of powerful and recent algorithms. The results showed that SMOBHC outperformed SMO and other algorithms in terms of finding an optimal path for the UCAV in a complex 2D environment.

Overall, it can be seen that the SMO algorithm and its improved version, SMOBHC, are highly effective in solving optimization problems, particularly in the area of UCAV path planning. The successful integration of the BHC algorithm into the SMO algorithm has resulted in significant improvements in its performance, which is demonstrated through extensive experimentation and comparison with other algorithms in this study.

A list of the meanings of variables, abbreviations, and adjustable control variables utilized in this paper can be found in Table A1, which is located in Appendix A.

The rest of this paper is organized as follows: Section two presents a review of related works. Section three discusses the 2D mathematical model of the UCAV path planning problem. Section four provides details on the SMO, the Beta-Hill Climbing Optimizer (BHC), and the SMOBHC. Section five presents the experimental results. Finally, Section six contains our concluding remarks.

2. Related Review

The purpose of this review is to provide a comprehensive overview of recent research on Navigation Systems-based UAV Path Planning methods and Optimization Techniques-based UAV Path Planning methods. As emphasized in the introduction section, these methods are crucial in the field of Unmanned Aerial Vehicles (UAVs).

In this review, we present two tables that showcase the latest research on these path planning methods, highlighting their significance and relevance in the UAV industry. By examining these examples, readers will gain a deeper understanding of the current state of research in this field and the advancements made in recent years.

Tables 1 and 2 outline the advantages and limitations of each method. Navigation Systems-based UAV Path Planning methods utilize GPS and sensor-based technologies to plan and optimize paths for UAVs, while Optimization Techniques-based UAV Path Planning methods rely on mathematical algorithms and models to find the most efficient path for UAVs.

Navigation Systems-based UAV Path Planning methods offer several benefits, including real-time capabilities that enable adjustments to the UAV's path while in-flight. Additionally, these methods are relatively easy to implement and applicable to a wide range of UAV applications. However, their reliance on sensors and GPS could limit their effectiveness in areas with poor signal coverage.

On the other hand, Optimization Techniques-based UAV Path Planning methods can provide optimized paths for UAVs in complex environments. These methods can consider various constraints and objectives, making them suitable for a wide range of applications. However, they may require significant computational resources, making them less suitable for real-time applications.

To sum up, Navigation Systems-based UAV Path Planning and Optimization Techniques-based UAV Path Planning methods each have their own strengths and weaknesses, and the selection of the most suitable method for a particular UAV application will depend on specific requirements. This review has shed light on the capabilities and

limitations of each approach, which will be helpful for researchers and practitioners in making an informed decision about the path planning method that best meets their needs.

Table 1. A detailed study of UAV path planning using Navigation Systems-based UAV Path Planning.

Navigation Systems-Based UAV Path Planning			
Ref	**Proposed Algorithm**	**Advantages**	**Limitations**
[8]	A deep reinforcement learning approach for UAV navigation in high dynamic environments. The method uses a neural network to learn control policies from sensor inputs and reward signals.	Can handle high-dynamic environments, is adaptable to different environments, can learn from experience, and requires minimal human intervention.	Needs a large amount of data for training, may suffer from safety issues if deployed without sufficient training and can be affected by sensor noise and disturbances in the environment.
[9]	Autonomous navigation method for a solar-powered UAV that can provide secure communication in urban environments while avoiding eavesdropping. The method utilizes an onboard camera and a GPS receiver to create a real-time map of the environment and uses machine learning algorithms to detect and avoid potential eavesdropping areas.	Provides secure communication in urban areas, is environmentally friendly and cost-effective, and effective tool for surveillance and reconnaissance.	Limited flight time, GPS signal disruption, the accuracy of machine learning algorithms affected by the complexity of urban environments and multiple signals, and privacy and security concerns.
[10]	A 3D navigation method for a solar-powered UAV that provides secure communication in the presence of eavesdroppers and no-fly zones while being energy efficient. The method uses a dynamic programming algorithm to optimize the UAV's flight path and speed based on the available solar energy and communication requirements.	Provides secure communication in challenging environments, is energy efficient, can avoid no-fly zones, and is adaptable to different communication requirements.	May not be suitable for environments with low solar energy availability; requires accurate solar energy prediction; may not be able to handle sudden changes in the environment; may require a more complex communication system for larger distances.
[11]	The paper proposes a novel method for ensuring the integrity of GPS-based navigation for Unmanned Aerial Vehicles (UAVs) in an urban environment using an integrity map and a weighted sum of the navigation solution residuals.	The advantages of the proposed method are that it provides a means to detect and mitigate GPS signal errors and can improve the reliability and safety of UAV navigation in urban environments.	The limitations of the proposed method are that it relies on the assumption that the GPS errors are independent and identically distributed, which may not always be the case in practice. Additionally, the method may not be effective in environments where the GPS signal is severely obstructed or jammed.
[12]	The paper proposes a deep reinforcement learning approach for the autonomous navigation of Unmanned Aerial Vehicles (UAVs) in multi-obstacle environments. The method involves training a Deep Q-Network (DQN) agent to select actions that enable the UAV to navigate through an environment with multiple obstacles while maximizing its reward.	The proposed method offers several advantages, including the ability to navigate complex environments without prior knowledge of the environment, the ability to handle dynamic obstacles, and the ability to handle multiple obstacles simultaneously. Additionally, the method offers the potential to reduce human intervention in UAV navigation, which could lead to increased efficiency and reduced costs.	One potential limitation of the proposed method is the need for significant computational resources for training the DQN agent. Additionally, the method may not be well-suited for environments with high levels of uncertainty or unpredictability, as the agent's performance is dependent on the quality of the reward function and the training data. Finally, the method may not be able to handle certain types of obstacles or environments, such as those with complex or irregular shapes.

Table 1. *Cont.*

Navigation Systems-Based UAV Path Planning			
Ref	**Proposed Algorithm**	**Advantages**	**Limitations**
[13]	A safe path planning algorithm for UAVs that uses a grid-based approach and considers the risk of GNSS signal occlusion in urban environments.	The algorithm takes into account the risk of GNSS signal occlusion and is able to plan safe paths in urban environments with high accuracy and efficiency.	The algorithm relies on accurate and up-to-date maps of the environment and may not be able to handle unforeseen obstacles or changes in the environment in real-time.
[14]	The paper proposes an Explainable Deep Reinforcement Learning (X-DRL) algorithm for UAV autonomous path planning.	The X-DRL algorithm provides a way to interpret the decision-making process of the UAV, allowing for greater transparency and accountability. It also achieves higher accuracy and faster convergence compared to other methods.	The proposed algorithm relies on a pre-defined reward function and may not be suitable for complex environments where the optimal reward function is unknown. Additionally, the X-DRL algorithm requires a large amount of training data and computational resources.
[15]	The paper proposes a new UAV navigation method called DNCS, which considers the no-fly zone and efficiently selects the charging station to optimize the UAV's energy consumption.	The DNCS method can effectively avoid no-fly zones and choose the best charging station based on the UAV's energy level, resulting in longer flight time and reduced energy consumption.	The proposed method is evaluated through simulations and may not reflect real-world scenarios. The accuracy of the no-fly zone map and the availability of charging stations may also affect the performance of the method.
[16]	The paper proposes a highly reliable relative navigation method for multi-UAV formation flight in urban environments using a combination of vision-based and sensor-based techniques.	The proposed method achieves high reliability in relative navigation, even in challenging urban environments with obstacles and GPS-denied conditions. It also allows for flexible formation reconfiguration and can be applied to a variety of multi-UAV formations.	The proposed method relies on the availability of multiple UAVs and may not be suitable for single UAV missions. The accuracy of the method is also affected by the quality of sensor data and the complexity of the environment.
[17]	Deep Q-Network (DQN)-based path planning algorithm for multiple UAVs for energy-efficient data collection in UAV-assisted IoT.	The proposed algorithm can efficiently plan optimal paths for multiple UAVs, reducing energy consumption and enabling more efficient data collection.	The algorithm relies on accurate and timely communication between the UAVs and the ground station, which can be challenging in certain scenarios. Additionally, the algorithm assumes that the UAVs have unlimited battery capacity, which may not be practical in real-world scenarios.
[18]	Optimized belief propagation-based cooperative navigation algorithm for multiple UAVs in GNSS-denied areas using only onboard sensors.	The proposed algorithm can achieve high positioning accuracy and reliability in GNSS-denied areas by leveraging the cooperation of multiple UAVs and the optimization of the belief propagation algorithm.	The proposed algorithm relies on the availability of multiple UAVs and assumes that they can communicate with each other in real-time, which may not always be feasible in practical scenarios. Additionally, the algorithm may require significant computational resources, which may limit its real-time applicability.

Table 1. *Cont.*

Navigation Systems-Based UAV Path Planning			
Ref	**Proposed Algorithm**	**Advantages**	**Limitations**
[19]	Soft actor-critic with Hindsight Experience Replay (HER) algorithm based on deep reinforcement learning for model-free path planning and collision avoidance for UAVs.	The proposed algorithm can learn effective collision-free paths for UAVs in complex environments using only raw sensory input, without relying on a pre-defined map or model of the environment.	The training of the algorithm can be computationally expensive and time-consuming, and the resulting policies may not always generalize well to unseen environments. Additionally, the algorithm may require a large amount of data to achieve good performance, which can be challenging to obtain in real-world scenarios.
[20]	Barrier Lyapunov Function (BLF)-based control algorithm for safe navigation of quadrotor UAVs in the presence of uncertain dynamics and with guaranteed collision avoidance.	The proposed algorithm can guarantee safe and collision-free navigation of quadrotor UAVs even in the presence of model uncertainties and external disturbances.	The algorithm assumes that the UAV's state and environment information are accurately known, which may not always be the case in real-world scenarios. Additionally, the algorithm may require tuning of certain parameters, which can be challenging for users without a deep understanding of control theory.
[21]	Real-time obstacle identification and avoidance algorithm based on point cloud data for UAV navigation in environments with simultaneous static and dynamic obstacles.	The proposed algorithm can identify and avoid both static and dynamic obstacles in real-time, enabling safe and efficient UAV navigation in complex environments.	The algorithm relies on accurate and reliable point cloud data, which may not always be available or may be affected by environmental factors such as weather. Additionally, the algorithm may struggle with identifying and avoiding small or transparent obstacles that are not easily detectable in point cloud data.
[22]	Mission-based planning, tasking, and re-tasking (PTR) triangle framework for flight planning of multi-UAV systems in complex missions.	The proposed framework can facilitate efficient mission planning, tasking, and re-tasking of multi-UAV systems, enabling improved mission performance and adaptability.	The framework may require significant communication and computational resources to achieve real-time re-tasking and adaptation of the UAVs, which can be challenging in certain scenarios. Additionally, the framework assumes that the UAVs have unlimited battery capacity and does not consider the impact of energy consumption on mission performance.
[23]	Cooperative marine search and rescue framework based on visual navigation and reinforcement learning-based control for USV-UAV systems.	The proposed framework enables efficient and effective search and rescue missions in marine environments by combining the advantages of both USVs and UAVs. The use of visual navigation and reinforcement learning-based control can improve the adaptability and robustness of the system.	The framework may require significant computational resources to process a large amount of sensory data and perform real-time control of the USV-UAV system. Additionally, the framework assumes that the system has accurate and reliable sensors and communication capabilities, which may not always be the case in real-world scenarios.

Table 1. *Cont.*

Navigation Systems-Based UAV Path Planning			
Ref	**Proposed Algorithm**	**Advantages**	**Limitations**
[24]	Velocity-adaptive 3D local path planning method for collision-avoided tracking control of UAVs.	The proposed method can enable efficient and safe tracking control of UAVs in complex environments by adapting the velocity of the UAV to avoid obstacles and reduce the risk of collisions. Path planning in 3D can provide more flexibility and maneuverability for the system.	The proposed method relies on accurate and reliable sensor data for obstacle detection and path planning, which may not always be available or may be affected by environmental factors such as weather. Additionally, the method may not always guarantee optimal or globally optimal paths for the UAV.
[25]	Spatiotemporal motion planner for quadrotors in unknown environments with unbending and consistency awareness referred to as STUNS-Planner.	The STUNS-Planner can enable quadrotors to navigate safely and efficiently in unknown environments by taking into account the dynamics of the quadrotor and the surrounding obstacles. The unbending and consistency awareness can improve the stability and robustness of the system.	The proposed method may be computationally intensive and require significant computing resources, limiting its real-time applicability. Additionally, the method may not always guarantee globally optimal paths or may be affected by sensor noise or inaccuracies in the estimation of the quadrotor dynamics.

Table 2. A detailed study of UAV path planning using Optimization Techniques-based UAV Path Planning.

Optimization Techniques-Based UAV Path Planning			
Ref	**Proposed Algorithm**	**Advantages**	**Limitations**
[26]	A novel Swarm Intelligence algorithm called the Anas Platyrhynchos Optimizer (APO) for global optimization and multi-UAV cooperative path planning.	The APO algorithm can optimize multiple objectives, such as energy consumption and path length while taking into account the coordination and cooperation among multiple UAVs. The method has been shown to achieve good performance and convergence speed in simulation experiments.	The effectiveness of the APO algorithm may depend on the choice of parameters and may be affected by the complexity of the optimization problem. The algorithm may also require significant computational resources for large-scale problems, limiting its practical applicability.
[27]	Multiple Swarm Fruit Fly Optimization Algorithm-Based Path Planning method for multi-UAVs, which utilizes a hybrid of a Fruit Fly Optimization Algorithm and Swarm Intelligence for path planning of multiple Unmanned Aerial Vehicles (UAVs).	The proposed method provides a balance between exploration and exploitation and improves the efficiency and effectiveness of path planning for multi-UAVs.	The proposed method does not consider dynamic environments and obstacles, which may limit its applicability in real-world scenarios. Additionally, the computational complexity may increase with an increasing number of UAVs.
[28]	A modified Particle Swarm Optimization (PSO) algorithm for autonomous path planning of Unmanned Aerial Vehicles (UAVs) in 3D environments that takes into account obstacles and the need for safe separation distances.	The proposed method achieves a higher success rate and a lower computational time than other PSO-based path planning methods. It can also handle complex environments and be used for both single and multiple UAVs.	The proposed method assumes that the UAV has complete and accurate knowledge of the environment, which may not be the case in real-world scenarios. Additionally, the method may require tuning its parameters to achieve optimal performance.

Table 2. *Cont.*

Optimization Techniques-Based UAV Path Planning			
Ref	**Proposed Algorithm**	**Advantages**	**Limitations**
[29]	An improved Pigeon-Inspired Optimization algorithm (IPIO) for path planning of Unmanned Aerial Vehicles (UAVs) in dynamic 3D environments with moving obstacles, specifically designed for oilfield inspections.	The proposed IPIO algorithm outperforms other popular optimization algorithms in terms of convergence rate and solution quality. It can effectively handle the complexities of a dynamic oilfield environment while optimizing multiple objectives, such as the shortest path and safe separation distances.	The proposed method assumes perfect information about the environment and obstacles, which may not always be available in real-world scenarios. Additionally, the computational time may increase with larger search spaces and more UAVs.
[30]	A Multi-Population Ensemble Differential Evolution (MP-EDE) algorithm for path planning of Unmanned Aerial Vehicles (UAVs) in 3D environments that optimizes multiple objectives such as collision avoidance and path smoothness.	The proposed MP-EDE algorithm shows a better convergence rate, diversity, and robustness compared to other state-of-the-art algorithms for UAV path planning. It can also handle complex 3D environments with multiple UAVs.	The proposed method assumes a static environment and may require parameter tuning to achieve optimal performance. Additionally, the computational time may increase with larger search spaces and more UAVs.
[31]	The paper proposes a path planning optimization algorithm based on Particle Swarm Optimization (PSO) for Unmanned Aerial Vehicles (UAVs) for bird monitoring and repelling.	The proposed algorithm can optimize the path planning for UAVs to effectively monitor and repel birds, which can reduce the damage caused by birds to crops and reduce the risk of bird-aircraft collisions.	The paper does not provide extensive experimental results, and the proposed algorithm may not be applicable to other scenarios beyond bird monitoring and repelling.
[32]	The paper proposes a modified Multi-Objective Particle Swarm Optimization (MOPSO) algorithm for path planning of Unmanned Aerial Vehicles (UAVs) in 3D terrain with constraints.	The proposed method is able to find feasible paths that satisfy different constraints while considering multiple objectives, leading to better performance compared to other optimization methods.	The proposed method is computationally intensive and may not be suitable for real-time applications with limited computational resources. Additionally, the effectiveness of the method is dependent on the accuracy of the input data used to represent the terrain.
[33]	The paper proposes an Adaptive Genetic Algorithm (AGA) and an Improved Artificial Bee Colony (IABC) method for multi-UAV mission assignment and path planning for disaster rescue.	The proposed method is able to simultaneously optimize the mission assignment and path planning for multiple UAVs, improving the efficiency and effectiveness of disaster rescue operations. It also incorporates adaptive strategies to enhance the performance of the optimization algorithms.	The proposed method has not been tested in real-world disaster rescue scenarios, and its performance may depend on the accuracy of the underlying models and assumptions. Additionally, the computational complexity of the method may limit its scalability to larger numbers of UAVs or more complex scenarios.
[34]	A Hybrid Differential Symbiotic Organisms Search (HDSOS) algorithm for unmanned aerial vehicle (UAV) path planning.	HDSOS combines the advantages of differential evolution and symbiotic organisms search algorithms, leading to improved global search capabilities and faster convergence in finding optimal paths for UAVs.	The effectiveness of HDSOS for UAV path planning may be limited by the complexity of the environment and the number of waypoints to be visited.

Table 2. *Cont.*

Optimization Techniques-Based UAV Path Planning			
Ref	**Proposed Algorithm**	**Advantages**	**Limitations**
[35]	An improved Equilibrium Optimizer (EO) for unmanned aerial vehicle (UAV) path planning, which includes a modified search strategy and a new balance operator.	The improved EO algorithm demonstrated improved performance in terms of finding optimal paths for UAVs compared to other optimization algorithms.	The effectiveness of the improved EO algorithm may be limited by the complexity of the environment and the number of waypoints to be visited. Additionally, the algorithm may require tuning its parameters for optimal performance.
[36]	Distributed 3D path planning for multiple UAVs based on the Particle Swarm Optimization (PSO) algorithm for full-area surveillance.	Provides a distributed approach to path planning for multiple UAVs with high accuracy and efficiency, as well as full-area surveillance with reduced time and cost.	The proposed method may require a large number of UAVs to achieve full-area surveillance, and it may not consider dynamic obstacles in real-time.
[37]	Parallel Cooperative Coevolutionary Grey Wolf Optimizer (CCGWO) for path planning of Unmanned Aerial Vehicles (UAVs) in complex environments.	Improved convergence speed and solution quality compared to other optimization algorithms, and the ability to handle complex environments with dynamic obstacles.	The proposed CCGWO method may require significant computational resources to run, and it may not perform well in cases where the search space is not well defined.
[38]	Path planning for multiple Unmanned Aerial Vehicles (UAVs) with time windows using the Grey Wolf Algorithm (GWA).	Improved efficiency and accuracy in path planning with time constraints, as well as the ability to optimize multiple UAVs simultaneously.	The proposed GWA-based method may struggle to find optimal solutions in complex environments with many obstacles, and it may not be suitable for real-time applications due to the computational resources required.
[39]	A Chaos-Enhanced Particle Swarm Optimization (CPSO) algorithm for safe path planning of Unmanned Aerial Vehicles (UAVs).	The proposed CPSO algorithm is efficient at finding safe paths for UAVs while avoiding obstacles and can handle complex environments. The chaos enhancement improves the algorithm's convergence speed and accuracy.	The CPSO algorithm's performance may be affected by the selection of hyperparameters, and it may struggle with time-constrained applications as the search process can take longer than expected.
[40]	An algorithm for path planning for Unmanned Aerial Vehicles (UAVs) based on an improved Harris Hawks Optimization (IHHO) algorithm.	The IHHO algorithm provides high accuracy in finding optimal paths for UAVs while avoiding obstacles, and it can handle complex environments with multiple objectives.	The proposed algorithm may require significant computational resources to run, and it may not be suitable for real-time applications where fast planning is required.
[41]	Adaptive Cylinder Vector Particle Swarm Optimization with Differential Evolution (ACVPSO-DE) for unmanned aerial vehicle (UAV) path planning.	The proposed ACVPSO-DE algorithm provides high-quality solutions for finding safe paths for UAVs while avoiding obstacles in real-time applications.	The performance of the algorithm may be impacted by the selection of hyperparameters, and it may struggle with complex environments with multiple objectives.
[42]	A hybrid algorithm based on the Grey Wolf optimizer and Differential Evolution (GWO-DE) for unmanned aerial vehicle (UAV) path planning.	The proposed GWO-DE algorithm provides an efficient and effective solution for finding safe paths for UAVs while avoiding obstacles in dynamic environments.	The proposed algorithm may struggle with complex environments with multiple objectives, and the selection of hyperparameters may impact the performance.

Table 2. *Cont.*

Optimization Techniques-Based UAV Path Planning			
Ref	**Proposed Algorithm**	**Advantages**	**Limitations**
[43]	The proposed method is RGSO-UAV, which is a path planning algorithm inspired by Reverse Glowworm Swarm Optimization (RGSO) for Unmanned Aerial Vehicles (UAVs) in a 3D dynamic environment.	The advantages of RGSO-UAV are that it is effective in finding optimal or near-optimal paths for UAVs in dynamic environments while also considering energy consumption, obstacle avoidance, and smoothness of the path.	The limitations of RGSO-UAV are not explicitly mentioned in the abstract, but further investigation of the paper may reveal any potential shortcomings, such as computational complexity or limited applicability to certain types of environments.

3. Problem Formulation

Path planning is a crucial component of a UCAV mission control system, and emergencies must be taken into account. The objective of this numerical problem is to find an optimal or near-optimal flight path for the UCAV that avoids obstacles and reaches its destination safely. The 2D mathematical model for UCAV path planning used in this paper was introduced in [44]. As shown in Figure 1, there are several threats on the battlefield, such as radars, missiles, and natural barriers. The impact of each threat is represented by a circle with a particular radius, which signifies the risk area that must be avoided during the flight. If any part of the path falls within a circle, the risk of the guided UCAV being damaged increases. Hence, considering all existing threats on the battlefield and fuel consumption, the main goal of the flight mission is to find an optimal flight path that connects the starting point to the endpoint.

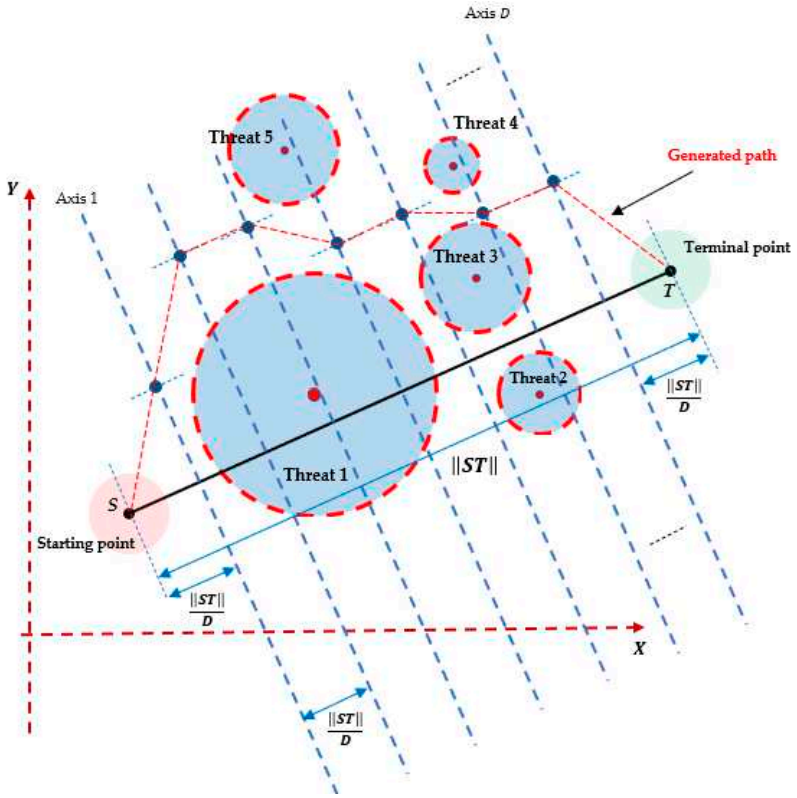

Figure 1. The schematic diagram for modeling the combat field. The *X* and *Y* axes represent the length and width of the horizontal battlefield, respectively. S and T indicate the starting and terminal points, respectively. All blue circles represent the threats on the battlefield. The black line connecting *S* and *T* directly represents the segment *ST*. The blue points indicate all generated nodes, which must be linked together to construct the optimal path indicated by the red dashed line.

3.1. UCAV Path Planning Model

According to [44], a mathematical modeling of this problem is carried out as follows:

1. A segment ST that directly connects the starting and terminal point is drawn,
2. The segment is divided into $(D+1)$ equal parts by D perpendicular lines,
3. All D lines are taken as new axes, and a series of points are assigned and connected on them to form a complete path.

The whole path is divided into $(D+1)$ steps, represented by a vector $S_i = Y_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$, where x_{ij} represents the jth dimension of the ith solution vector and indicates the discrete position of each step in the vertical axis. In reality, the components of the S_i vector represent the ordinates of the D path points relative to the ST segment, and their abscissae are $X_l = (x_1, x_2, \ldots, x_D) = ((\|ST\|/D), (\|ST\|/D) \times 2, \ldots, \|ST\|)$, $l = 1, \ldots D$, where $\|ST\|$ represents the length of the segment ST. For example, if $(0,0)$ is the starting point and $(\|ST\|, 0)$ is the endpoint, then the distance between them is $\|ST\|$. The vector S_i can be used to create a path with D nodes, where the coordinates of each node are $P_1((\|ST\|/D), x_{i1})$, $P_2((\|ST\|/D) \times 2, x_{i2})$, and so on. Linking all the nodes results in the complete path. Restrictions can be imposed on each solution path to keep it within the battle area, according to the size of the battlefield (refer to Figure 2). It is important to note that the coordinates (x^*, y^*) of the generated nodes $P_1, \ldots P_D$ in the plane SX^*Y^* are related to their coordinates (x, y) in the plane OXY through a simple transformation. This transformation can be expressed as $\begin{bmatrix} x^x \\ y^* \end{bmatrix} = \begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix}$.

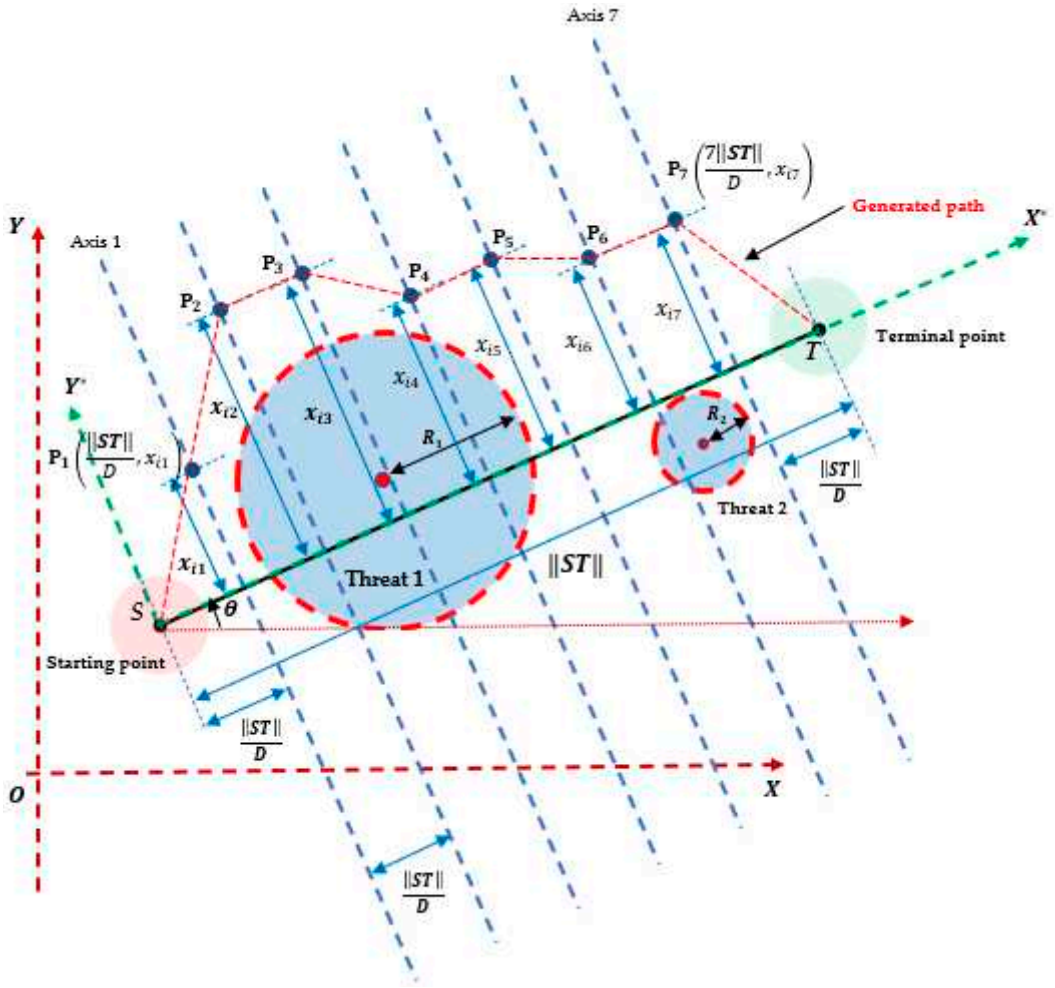

Figure 2. Thorough graphical depiction of the path planning modeling procedure (e.g., here $D = 7$ and $n_{thr} = 2$).

3.2. Objective Function

A probability density model, expressed in Equation (1), is used to evaluate the pathways. Unlike a conventional cost–function model, there is no clear cutoff where damage risk is negligible. As the distance grows, the likelihood of encountering a threat decreases, but it never reaches zero. The symbol $\|d_{kj}\|$ represents the distance between the jth step and the kth threat center, $\delta = R_i/log(20)$ is a parameter that determines the form of the density function, R_i is the radius of the ith threat element, and n_{thr} is the number of threats in the battlefield.

$$Cost_{threats} = \sum_{k=1}^{n_{thr}} exp\left(-\frac{\sum_{J=1}^{D} \|d_{kj}\|}{\delta}\right) \tag{1}$$

The fuel consumption of a UCAV with constant flight speed is solely tied to the flight path length, and its cost can be expressed as follows:

$$Cost_{fuel} = \frac{\sum_{J=1}^{D} d_{ij}}{\|ST\|} \tag{2}$$

where D denotes the number of steps, $\|ST\|$ is the length of the segment ST. Additionally, $\sum_{J=1}^{D} d_{ij}$ represents the length of the whole generated flight path, and d_{ij} is the length of the jth step of the ith solution.

Therefore, the objective function of the model taking into account threat cost and fuel cost can be written as:

$$f_{obj} = \gamma \sum_{k=1}^{n_{thr}} exp\left(-\frac{\sum_{J=1}^{D} \|D_{kj}\|}{\delta}\right) + (1-\gamma)\frac{\sum_{J=1}^{D} d_{ij}}{\|ST\|} \tag{3}$$

where γ is the weight coefficient ranging in [0, 1]. In this paper, γ is set to 0.5, meaning that both the threats cost and the fuel cost are of the same importance [29].

4. The Proposed New SMO Variant

4.1. The SMO Basic Algorithm

The Spider Monkey Optimization (SMO) algorithm is a population-based metaheuristic inspired by the Fission-Fusion System (FFS) of spider monkeys [5]. When stagnation occurs, the population is divided into subgroups, and a local leader is assigned to each subgroup while a global leader is assigned to the entire population. The algorithm operates in two main stages to update the population during the evolution process. In the first stage, called the Local Leader Stage or Local Leader Phase (LLP), the population is updated using the experience of the local leaders. In the second stage, called the Global Leader Stage or Global Leader Phase (GLP), the population is updated using the experience of the global leader. If no better individual is found through iterations, the global leader will divide the entire population into subgroups. If the number of subgroups reaches the Maximum Group limit (MG), the subgroups will reunite into a single population. If a local leader cannot be updated, the subgroup will be randomly generated and a new direction will be explored. In addition to MG, the SMO algorithm has three main control parameters: the *Global Leader Limit* (GLL), the *Local Leader Limit* (LLL), and the perturbation rate (pr). The first two parameters are used to track the number of times the global and local leaders cannot be updated through iterations, while the last parameter controls the amount of perturbation in the current position. Based on the FFS, the SMO algorithm performs better than other Swarm Intelligence algorithms, particularly for the UCAV path planning problem [7]. The basic version of the SMO algorithm is described in detail in the following sub-sections.

4.1.1. Initialization Step

Initially, N Spider Monkey positions ($X_{SM}^{i,j}$) with D dimensions are generated using a random distribution in the search interval $[Lb, Ub]$, as shown in the following equation:

$$X_{SM}^{i,j} = Lb_j + rand(0,1) \times \left(Ub_j - Lb_j\right),\ j = 1, \ldots\ldots D \tag{4}$$

where $X_{SM}^{i,j}$ is the ith spider monkey position in the jth dimension, Lb_j and Ub_j are the lower and the upper bounds respectively for the jth component in the ith spider monkey position, and $rand(0,1)$ is a randomly generated number in the interval $[0,\ 1]$.

4.1.2. Local Leader Phase (LLP)

In this step, each spider monkey element adjusts its current position based on the experience of the Local Leader and all members of the same kth local spider monkey group, as described by the following equation:

$$X_{SM_{new}}^{i,j} = X_{SM}^{i,j} + rand(0,1) \times \left(X_{LL\ SM}^{k,j} - X_{SM}^{i,j}\right) + rand(-1,1) \times \left(X_{SM}^{r,j} - X_{SM}^{i,j}\right) \tag{5}$$

where $X_{SM}^{i,j}$ is the jth component of the ith spider monkey position that belongs to the kth local spider monkey group, $X_{LL\ SM}^{k,j}$ represents the jth component of the kth local group leader position, $X_{SM}^{r,j}$ is the jth element of the rth spider monkey position in the k^{th} local spider monkey group, with the condition that $r \neq i$. $rand(0,\ 1)$ and $rand(-1,\ 1)$ are randomly generated numbers in the intervals $[0,\ 1]$ and $[-1,\ 1]$ respectively. It is important to note that the new solution $X_{SM_{new}}^{i,j}$obtained is evaluated and compared to the old one $X_{SM}^{i,j}$ based on a fitness value calculated using the objective function $f_{obj}()$. If $X_{SM}^{i,j}$ is worse than $X_{SM_{new}}^{i,j}$, an update of $X_{SM}^{i,j}$ to $X_{SM_{new}}^{i,j}$ is performed. Algorithm 1 provides a detailed description of the process for generating and updating solutions in the SMO algorithm LLP.

Algorithm 1: Local Leader Phase (LLP)

Input: pr
Output: $X_{SM_{new}}^{i,j}$
function SMO-LLP
. for each member spider monkey$_i \in k$th group do
  if rand$(0,1) <$ pr then
    Compute a novel spider monkey solution $X_{SM_{new}}^{i,j}$ using (5).
    if $f_{obj}\left(X_{SM_{new}}^{i,j}\right) < f_{obj}\left(X_{SM}^{i,j}\right)$ then
    $X_{SM}^{i,j} = X_{SM_{new}}^{i,j}$
    end if
  else
    $X_{SM_{new}}^{i,j} = X_{SM}^{i,j}$
  end if
 end for
end function

4.1.3. Global Leader Phase (GLP)

In this stage of the SMO algorithm, which starts immediately after the LLP, all members of the spider monkey group change their positions using information received from both the Global Leader and local group members. This SMO step generates a new global spider monkey solution as follows:

$$X_{SM_{new}}^{i,j} = X_{SM}^{i,j} + rand(0,1) \times \left(X_{GL\ SM}^{j} - X_{SM}^{i,j}\right) + rand(-1,1) \times \left(X_{SM}^{r,j} - X_{SM}^{i,j}\right) \tag{6}$$

where $X^{j}_{GL\ SM}$ represents the jth component of the global leader position, which is chosen randomly from the set $j \in \{1, 2, \ldots\ldots \mathrm{D}\}$. The generation of $X^{i,j}_{SM_{new}}$ follows a random selection mechanism based on a random value called $prob_i$, which is calculated using the following formula [31]:

$$prob_i = \frac{fitness_i}{\sum_{i=1}^{N} fitness_i} \tag{7}$$

It is evident from Equation (7) that the spider monkey with a better fitness value has a greater chance of improving itself, where $fitness_i$ represents the fitness value of the ith spider monkey. Furthermore, just as in the LLP, the existing solution $X^{i,j}_{SM}$ is replaced by $X^{i,j}_{SM_{new}}$ if the new one is superior. Algorithm 2 provides a detailed explanation of the process of generating a new solution and updating the existing one in the GLP SMO algorithm.

Algorithm 2: Global Leader Phase (GLP)

Input: $prob_i$
Output: $X^{i,j}_{SM_{new}}$
function SMO-GLP
. for each member spider monkey$_i \in$ whole SM group do
if rand$(0,1) < prob_i$ then
Select a j index randomly from the set $\{1, 2, \ldots\ldots D\}$
Compute a novel spider monkey solution $X^{i,j}_{SM_{new}}$ using (6).
if $f_{obj}\left(X^{i,j}_{SM_{new}}\right) < f_{obj}\left(X^{i,j}_{SM}\right)$ **then**
$X^{i,j}_{SM} = X^{i,j}_{SM_{new}}$
end if
else
$X^{i,j}_{SM_{new}} = X^{i,j}_{SM}$
end if
end for
end function

4.1.4. Global Leader Learning (GLL) Phase

In this phase of the SMO algorithm, the solution carried by the Global Leader element is updated using greedy selection across the entire population. This means that the spider monkey element with the best fitness simply becomes the global leader. However, there is a possibility that the update may not be performed, and in such cases, the number of non-updating times is tracked using the Global Limit Count (GLL). This counter is incremented by one each time the update is not performed, until it reaches the limit value specified by the Global Leader Limit.

4.1.5. Local Leader Learning (LLL) Phase

This stage is similar to the GLL phase in terms of its steps. The only difference is that the update at this stage concerns the position of each Local Leader in its subgroup. If the position of a Local Leader is not updated, a Local Limit Count counter specific to that subgroup is incremented by one until it reaches the global limit value specified by the Local Leader Limit.

4.1.6. Local Leader Decision (LLD) phase

The SMO algorithm enters this stage after checking if the kth Local Limit Count has reached or exceeded the Local Leader Limit. If this condition is met, all members of the

*k*th spider monkey group update their position either through random initialization (as indicated by Equation (4)) or using the following equation:

$$X_{SM_{new}}^{l,j} = X_{SM}^{l,j} + rand(0,1) \times \left(X_{GL\ SM}^{j} - X_{SM}^{l,j}\right) + rand(0,1) \times \left(X_{SM}^{l,j} - X_{LL\ SM}^{k,j}\right) \quad (8)$$

where $X_{SM_{new}}^{l,j}$ is the newly generated solution for the *k*th group. l is an index that takes the following cyclic values: $\left[l_1^k, l_1^k + 1, \ldots\ldots, l_1^k + \frac{N}{MG}\right]$, where l_1^k is the first index of the *k*th group, and $X_{SM}^{l,j}$ is the existing solution for the *k*th group. The position of each Local Leader is updated by taking the position of the best spider monkey element within each group. Algorithm 3 provides a detailed description of the LLD phase.

Algorithm 3: Local Leader Decision (LLD) phase

Input: pr, Local Leader Limit (LLL), MG
Output: $X_{SM_{new}}^{l,j}$ **for the *k*th group,** $X_{LL\ SM_{new}}^{k,j}$**, the *k*th *Local Limit Count* = 0**
function SMO-LLD phase
. if the *k*th Local Limit Count $\leq$ Local Leader Limit
The *k*th Local Limit Count = 0
for each $l \in \left\{l_1^k, l_1^k + 1, \ldots\ldots, l_1^k + \frac{N}{MG}\right\}$ do
for each $j \in \{1, 2, \ldots\ldots D\}$ do
if rand$(0,1) <$ pr then
Compute a novel spider monkey solution $X_{SM_{new}}^{l,j}$ using (4).
else
Compute a novel spider monkey solution $X_{SM_{new}}^{l,j}$ using (8).
end if
end for
end for
$X_{LL\ SM_{new}}^{k,j} = the\ best\ X_{SM_{new}}^{l,j}$
The *k*th Local Limit Count = 0
end if
end function

4.1.7. Global Leader Decision (GLD) Phase

This step in the SMO algorithm is based on the status of the GLL phase counter. If the latter is greater than or equal to the Global Leader Limit, the population is divided. This division leads to the creation of new local leaders for each part of the population and a global leader for the entire population. These actions, including fragmentation, creation of new local leaders, and global leader creation, are monitored by the status of the GLL phase counter until the maximum number of divisions, equal to the Maximum Group Limit, is reached. At this stage, all groups created are combined into a single global group. It is important to note that the GLL phase counter is reset to zero after each division and aggregation. Algorithm 4 provides a detailed explanation of the GLD phase.

4.2. The Beta-Hill Climbing Optimizer (BHC)

The Beta-Hill Climbing optimizer, created by Mohammed Azmi Al-Betar [45], is a novel local search method that explores solutions. The BHC algorithm is iterative and starts with the generation and evaluation of a random solution, $x_j = (x_1, x_2, \ldots\ldots x_D)$, where D is the dimension of the problem. The algorithm employs two main operators, the $\mathcal{N}$-operator and the $\mathcal{B}$-operator, which are used for exploitation and exploration, respectively. The $\mathcal{N}$-operator serves as a neighborhood search, while the $\mathcal{B}$-operator operates similarly to a mutation operator. In each iteration of the BHC algorithm, the $\mathcal{N}$-operator is executed before the $\mathcal{B}$-operator. After the initial random solution x is evaluated using the objective

function $f_{obj}()$, the $\mathcal{N}$-operator updates it by producing a new solution x' in the vicinity of the original solution x:

$$x_j^{'} = x_j \pm rand(0,1) \times b_w \tag{9}$$

where j is randomly selected from the set $j \in \{1, 2, \ldots, D\}$, b_w denotes the distance between the new solution and the current solution, and $rand(0,1)$ represents a random number in the range $[0,\ 1]$. It is important to note that in each iteration, the $\mathcal{N}$-operator only modifies one component randomly in the solution vector $x_j = (x_1, x_2, \ldots\ldots x_D)$.

Using the B-operator, a new solution vector component, x'' is designed. Its index j is selected beforehand using a random selection, as shown in the following formula:

$$x_j^{''} = \begin{cases} l_j + (u_j - l_j) \times rand_1(0,1) & rand_2(0,1) < \mathcal{B} \\ x_j^{'} & otherwise \end{cases} \tag{10}$$

If the condition $rand_2(0,1) < \mathcal{B}$ is verified, the index j is randomly selected from the set $\{1, 2, \ldots, D\}$. $rand_1(0,1)$ and $rand_2(0,1)$ are random numbers in $[0,\ 1]$. Finally, we obtain a new solution vector x_j^{new}, for example, equal to:

$$x_j^{new} = \left(x_1, x', \ldots x'', \ldots x'', \ldots . x_D\right) \tag{11}$$

It should be noted that the x_j^{new} has only one component obtained using the $\mathcal{N}$-operator and at least one component found using the $\mathcal{B}$-operator. Additionally, the element x' can be replaced by an another x'' produced during the $\mathcal{B}$-operator stage. Finally, if $f_{obj}(x^{new}) < f_{obj}(x)$, then x is replaced by x^{new}.

Algorithm 4: Global Leader Decision (GLD) phase

Input: Global Leader Limit (GLL), MG

Output: $X_{GL\ SM_{new}}^{j}$, $X_{LL\ SM_{new}}^{k,j}$ **for each** *k***th group,** ***Global*** **Limit Count** $= 0$

function SMO-GLD phase
. if Global Limit Count $\leq$ Global Leader Limit
if Numbers of groups $<$ MG
Number of groups = Number of groups + 1
Generate a $X_{LL\ SM_{new}}^{k,j}$ equal to the existing groups, where $l \in \left\{l_1^k, l_1^k + 1, \ldots\ldots, l_1^k + \frac{N}{MG}\right\}$, k is the index of the kth group
Generate a new $X_{GL\ SM_{new}}^{j}$
Global Limit Count $= 0$
else
Numbers of groups =1
Global Limit Count $= 0$
end if
end if
end function

4.3. Proposed Method

The SMO algorithm has been improved by combining it with the BHC optimizer to create a new and more efficient variant called SMOBHC. This hybrid approach leverages the standard form of the BHC optimizer to optimize solutions generated in the SMO Local Leader Phase (LLP), improve solutions produced in the SMO Global Leader Phase (GLP), and finally update the positions of Local Leader members under specific conditions in the SMO Local Leader Decision (LLD) phase. The algorithmic changes made in each phase are outlined in Algorithms 5–7 and visually depicted in Figures 3–5. A comprehensive flowchart of the SMOBHC algorithm can be found in Figure 6.

It's noteworthy that integrating $\mathcal{N}$ and $\mathcal{B}$ operators into these three crucial phases of the SMO algorithm can greatly enhance its global exploration and exploitation abilities

(action of the $\mathcal{N}$-operator) and convergence behavior (action of the $\mathcal{B}$-operator). These improvements will be highlighted in the experimental results.

Algorithm 5: Local Leader Phase (LLP) based on BHC

Input: pr

Output: $X_{SM_{new}}^{i,j}$

function SMO-LLP
. for each member $SM_i \in k$th group do
  if rand(0,1) < pr then
    Compute a novel spider monkey solution $X_{SM_{new}}^{i,j}$ using (5).
    if $f_{obj}(X_{SM_{new}}^{i,j}) < f_{obj}(X_{SM}^{i,j})$ **then**
      $X_{SM}^{i,j} = X_{SM_{new}}^{i,j}$
      $X_{BHC}^{j} = X_{SM}^{i,j}$ **BHC**
      $X_{BHC_{\mathcal{B}}-\text{operator}}^{j} = \mathcal{B} - Operator(X_{BHC}^{j})$
      $X_{BHC_{\mathcal{N}}-\text{operator}}^{j} = \mathcal{N} - Operator(X_{BHC_{\mathcal{B}}-\text{operator}}^{j})$
      $X_{BHC}^{j} = X_{BHC_{\mathcal{N}}-\text{operator}}^{j}$
      if $f_{obj}(X_{BHC}^{j}) < f_{obj}(X_{SM}^{i,j})$ **then**
        $X_{SM}^{i,j} = X_{BHC}^{j}$
        $X_{SM_{new}}^{i,j} = X_{SM}^{i,j}$
      end if
    end if
  else
    $X_{SM_{new}}^{i,j} = X_{SM}^{i,j}$
  end if
. end for
end function

Algorithm 6: Global Leader Phase (GLP) based on BHC

Input: $prob_i$

Output: $X_{SM_{new}}^{i,j}$

function SMO-GLP
. for each member $SM_i \in$ whole SM group do
  if rand(0,1) < $prob_i$ then
    Select a j index randomly from the set $\{1,2,\ldots..D\}$
    Compute a novel spider monkey solution $X_{SM_{new}}^{i,j}$ using (6).
    if $f_{obj}(X_{SM_{new}}^{i,j}) < f_{obj}(X_{SM}^{i,j})$ **then**
      $X_{SM}^{i,j} = X_{SM_{new}}^{i,j}$
      $X_{BHC}^{j} = X_{SM}^{i,j}$ **BHC**
      $X_{BHC_{\mathcal{B}}-\text{operator}}^{j} = \mathcal{B} - Operator(X_{BHC}^{j})$
      $X_{BHC_{\mathcal{N}}-\text{operator}}^{j} = \mathcal{N} - Operator(X_{BHC_{\mathcal{B}}-\text{operator}}^{j})$
      $X_{BHC}^{j} = X_{BHC_{\mathcal{N}}-\text{operator}}^{j}$

12: if $f_{obj}(X_{BHC}^{j}) < f_{obj}(X_{SM}^{i,j})$ **then**
13: $X_{SM}^{i,j} = X_{BHC}^{j}$
14: $X_{SM_{new}}^{i,j} = X_{SM}^{i,j}$
15: end if
16: end if
17: else
18: $X_{SM_{new}}^{i,j} = X_{SM}^{i,j}$
19: end if
20: end for
21: **end function**

Algorithm 7: Local Leader Decision (LLD) phase based on BHC

Input: pr, **Local Leader Limit**, **MG**

Output: $\boldsymbol{X_{SM_{new}}^{l,j}}$ for the ***k*th** group, $\boldsymbol{X_{LL\,SM_{new}}^{k,j}}$, the ***k*th *Local*** *Limit Count* = 0

1: **function** SMO-LLD phase
2: . if the *k*th *Local Limit Count* $\leq$ *Local Leader Limit*
3: The *k*th *Local Limit Count* = 0
4: for each $l \in \left\{l_1^k, l_1^k + 1, \ldots\ldots, l_1^k + \frac{N}{MG}\right\}$ do
5: for each $j \in \{1,2, \ldots\ldots D\}$ do
6: if rand(0,1) < pr then
7: Compute a novel spider monkey solution $X_{SM_{new}}^{l,j}$ using (4).
8: else
9: Compute a novel spider monkey solution $X_{SM_{new}}^{l,j}$ using (8).
10: end if
11: end for
12: end for
13: for each $l \in \left\{l_1^k, l_1^k + 1, \ldots\ldots, l_1^k + \frac{N}{MG}\right\}$ do
14: if $f_{obj}(X_{SM_{new}}^{l,j}) < f_{obj}(X_{LL\,\boldsymbol{SM}}^{k,j})$ **then**
15: if rand(0,1) < 0.5 then
16: $X_{LL\,\boldsymbol{SM}}^{k,j} = X_{SM_{new}}^{l,j}$
17: The k^{th} *Local Limit Count* = 0
18: else
19: $X_{BHC}^{j} = X_{SM_{new}}^{l,j}$ **BHC**
20: $X_{BHC_{\mathcal{B}}-\text{operator}}^{j} = \mathcal{B} - Operator(X_{BHC}^{j})$
21: $X_{BHC_{\mathcal{N}}-\text{operator}}^{j} = \mathcal{N} - Operator(X_{BHC_{\mathcal{B}}-\text{operator}}^{j})$
22: $X_{BHC}^{j} = X_{BHC_{\mathcal{N}}-\text{operator}}^{j}$
23: if $f_{obj}(X_{BHC}^{j}) < f_{obj}(X_{SM_{new}}^{l,j})$ **then**
24: $X_{LL\,SM}^{k,j} = X_{BHC}^{j}$
25: The k^{th} *Local Limit Count* = 0
26: else
27: $X_{LL\,SM}^{k,j} = X_{SM_{new}}^{l,j}$
28: The k^{th} *Local Limit Count* = 0

29: end if

30: end if

31: end if

32: end for

33: $X^{k,j}_{LL\,SM_{new}} = X^{k,j}_{LL\,SM}$

34: end if

35: **end function**

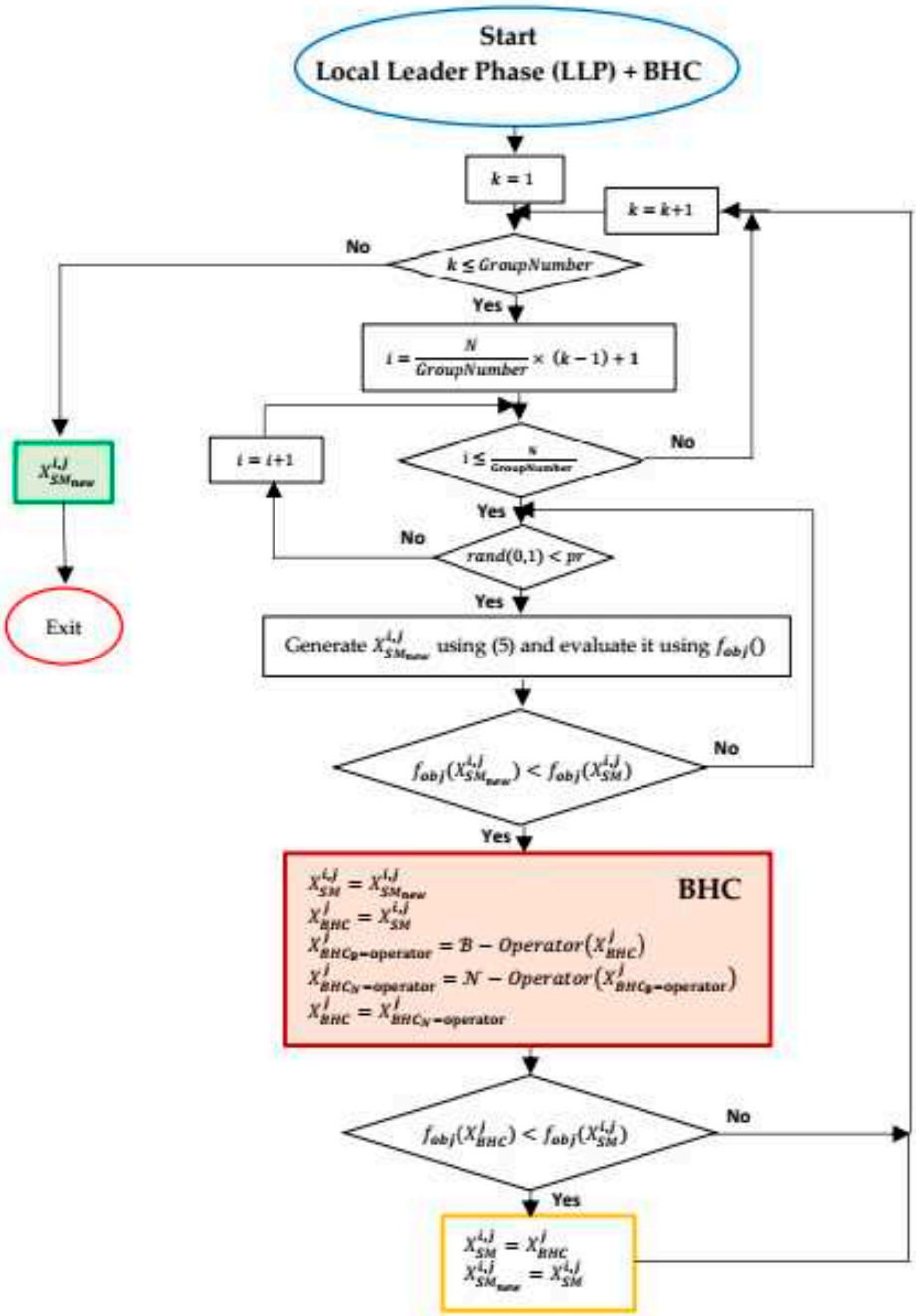

Figure 3. Flowchart of the Local Leader Phase (LLP) based on BHC.

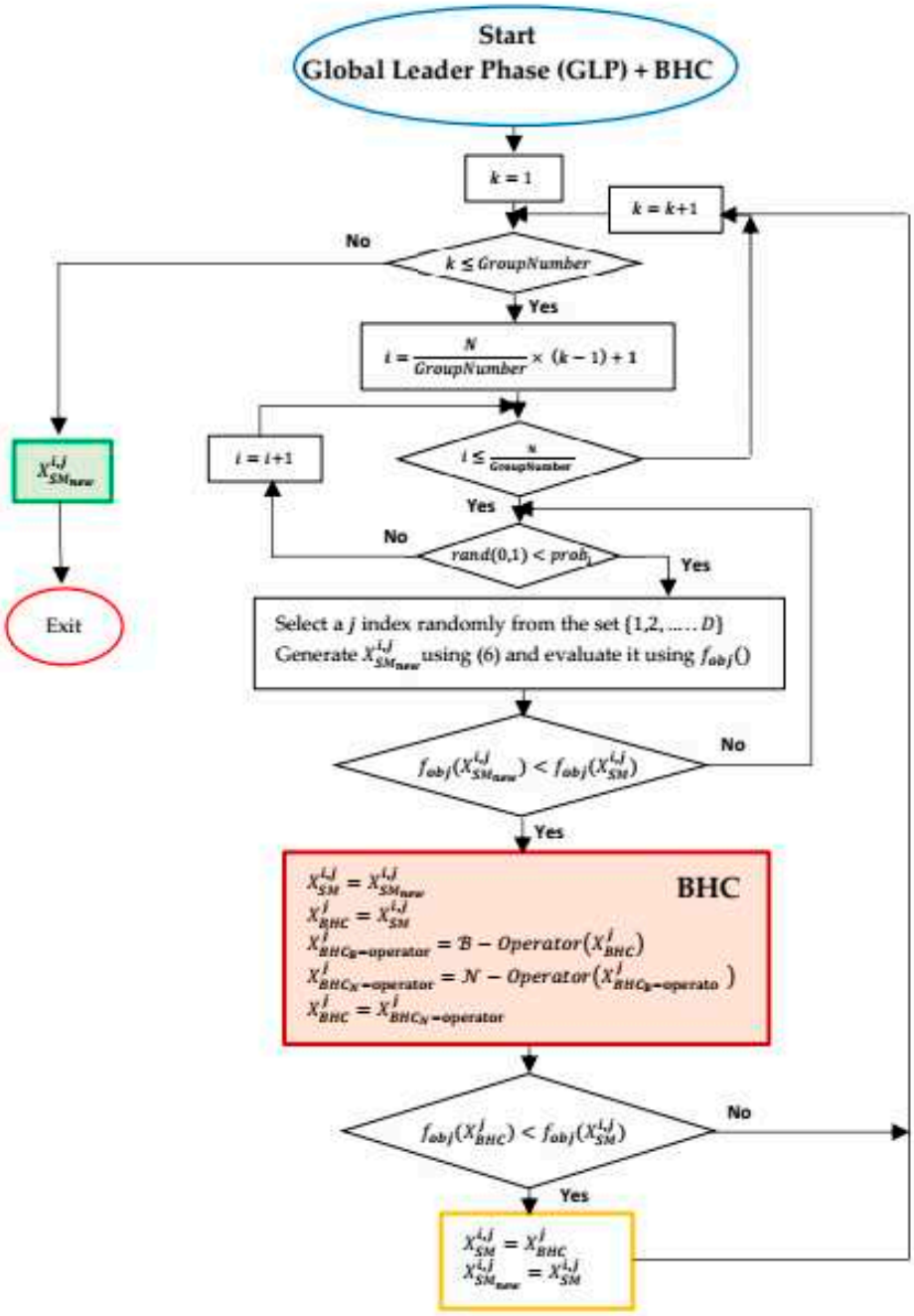

Figure 4. Flowchart of the Global Leader Phase (GLP) based on BHC.

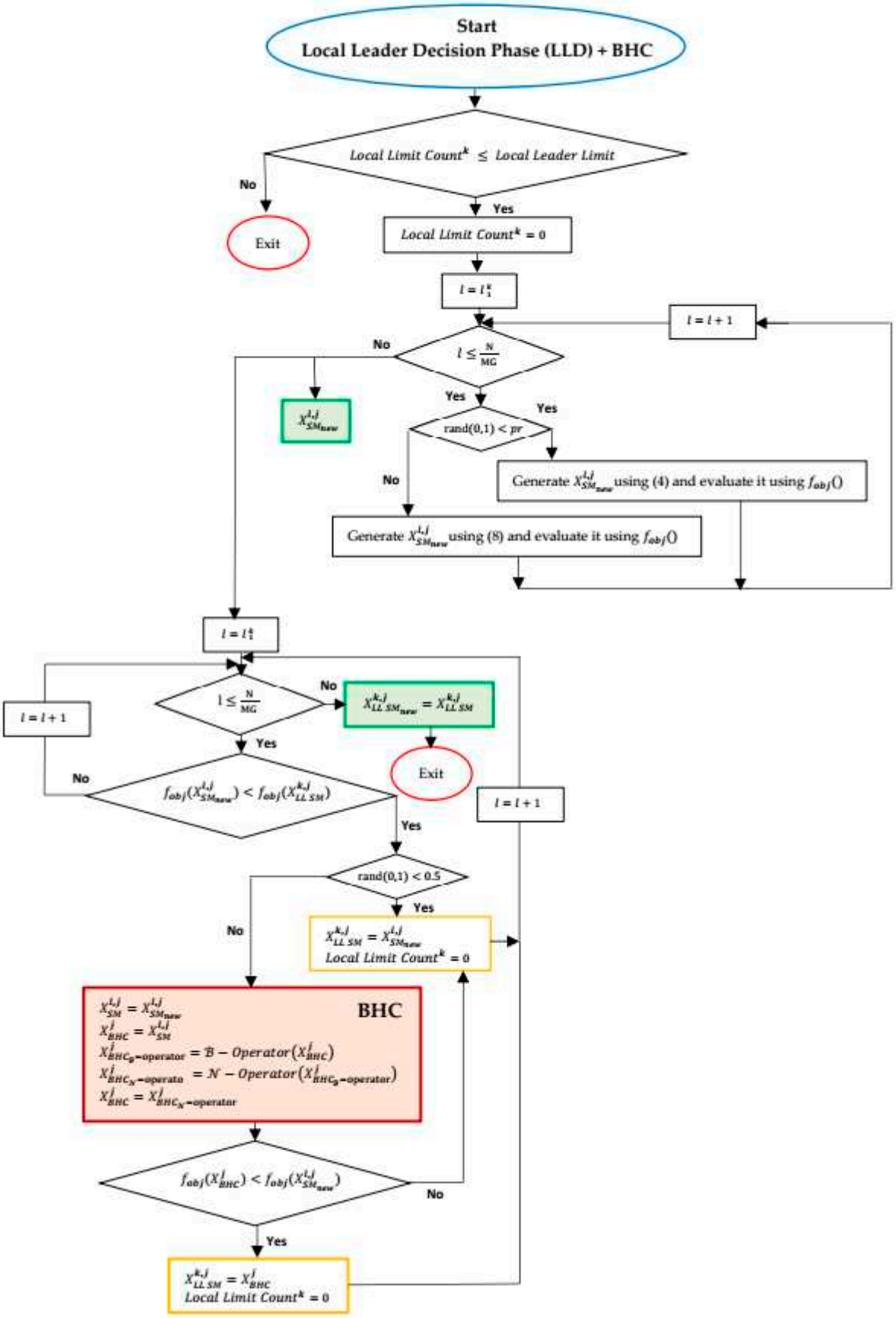

Figure 5. Flowchart of the Local Leader Decision (LLD) phase based on BHC.

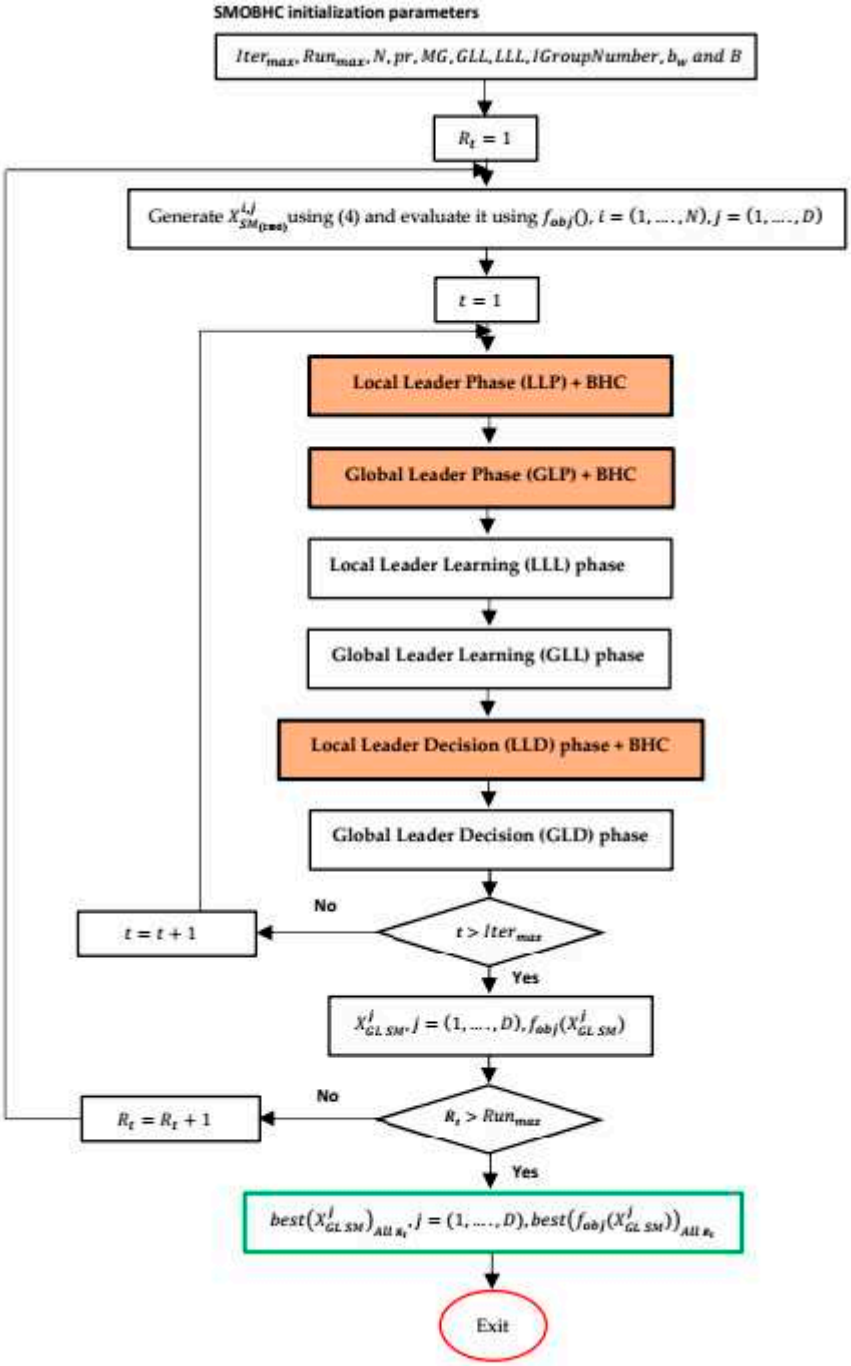

Figure 6. Flowchart of the SMOBHC algorithm.

Additionally, Figure 7 depicts the graphical representation of the UCAV path planning based on the SMOBHC algorithm. The flight device receives the coordinates of the optimal path nodes offline.

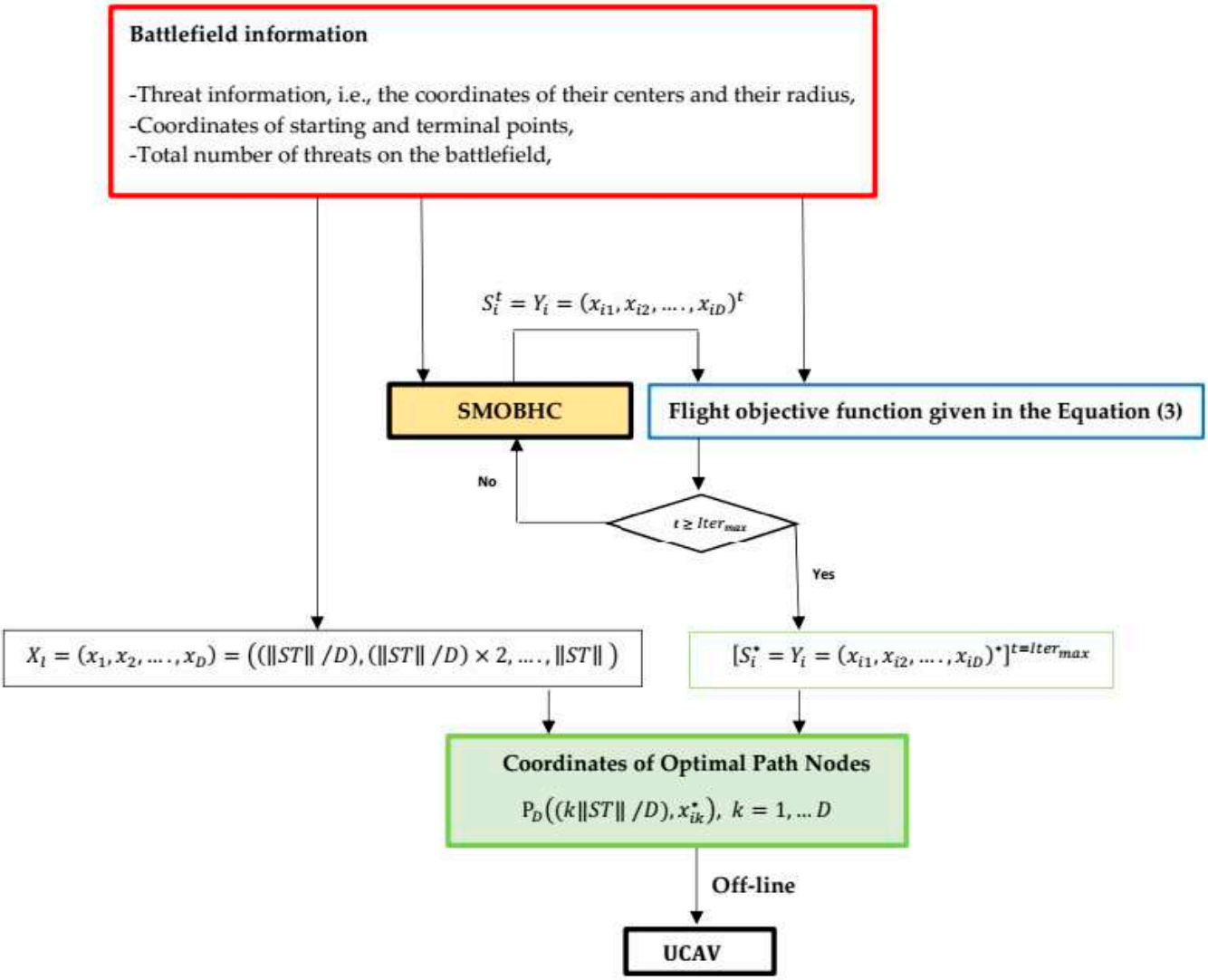

Figure 7. Graphical representation of UCAV Path planning based on SMOBHC.

5. Experimental Results and Discussions

To demonstrate the effectiveness and efficiency of our proposed method, SMOBHC, we conducted tests on three battlefields, each containing ten, thirty, and twenty threats distributed randomly under specific conditions. The algorithms used in the tests were run on a PC with an Intel(R) Core (TM) i7-4510U CPU at 2.60 GHz, 6 GB RAM, and 64-bit Windows 10 operating system. The algorithm codes were compiled using MATLAB 2022b. The centers and radii of all threats in the three battlefields were defined as follows:

Battlefield 1:

$$X^1_{\text{threat } center} = [350,\ 105,\ 305,\ 105, 175,\ 245,\ 415,\ 480,\ 40,\ 470],$$

$$Y^1_{\text{threat } center} = [200, -25, -150,\ 95,\ 55,\ 60,\ 0,\ 110,\ 100, -50\ ],$$

$$\text{Radius}^1_{\text{threat } center} = [140,\ 70,\ 150,\ 30,\ 25,\ 25,\ 25,\ 90,\ 40,\ 30\ ].$$

Battlefield 2:

$$X^2_{\text{threat } center_j} = \left[X^1_{\text{threat } center}, X^1_{\text{threat } center} + 500, X^1_{\text{threat } center} + 1000\right],\ \text{where } j = 1, \ldots 30,$$

$$Y^2_{\text{threat } center_j} = \left[Y^1_{\text{threat } center}, Y^1_{\text{threat } center}, Y^1_{\text{threat } center}\right],\ \text{where } j = 1, \ldots 30,$$

$$\text{Radius}^2_{\text{threat } center_j} = .\left[\text{Radius}^1_{\text{threat } center}, \text{Radius}^1_{\text{threat } center}, \text{Radius}^1_{\text{threat } center}\right], \text{ where } j = 1, ...30,$$

Battlefield 3:

$$X^3_{\text{threat } center_j} = \left[X^1_{\text{threat } center}, X^1_{\text{threat } center} + 500\right], \text{ where } j = 1, ...20,$$
$$Y^3_{\text{threat } center_j} = \left[Y^1_{\text{threat } center}, Y^1_{\text{threat } center}\right], \text{ where } j = 1, ...20,$$
$$\text{Radius}^3_{\text{threat } center_j} = .\left[\text{Radius}^1_{\text{threat } center}, \text{Radius}^1_{\text{threat } center}\right], \text{ where } j = 1, ...20,$$

The starting and terminal nodes for the three battlefields are positioned at points S and T, which have coordinates $S^1(0,0)$, $T^1(500,0)$, $S^2(0,0)$, $T^2(1000,0)$ and $S^3(10,10)$, $T^1(950,0)$, respectively. Additionally, the length of the segment ST for each battlefield is calculated using the following equation:

$$\|ST\| = \sqrt{\left(x_{Tj} - x_{Sj}\right)^2 - \left(y_{Tj} - y_{Sj}\right)^2} \tag{12}$$

In this study, the SMOBHC is benchmarked using three battlefields under three different problem dimensions of 30, 60, and 90. The control parameters for SMOBHC are set as follows: the size of the population N is 50, the maximum group limit (MG) is 2, the initial group number (GroupNumber) is 1, $LLL = N\ x\ D$, $GLL = 60$, $pr = 0.8$, $\mathcal{N}$-operator bandwidth (b_w) is 0.001, and $\mathcal{B}$-operator probability ($\mathcal{B}$) is 0.1. The SMOBHC is evaluated with a maximum of 1000 iterations ($Iter_{max}$) and run 20 times (Run_{max}). The best, worst, mean, and standard deviation (std.) of all SMOBHC results are calculated.

The SMOBHC algorithm is compared to SMO and 13 other algorithms for the first two battlefields under the same conditions. These algorithms are PSO, GQPSO [46], BA [47], CS [48], DE, FA [49], GCMBO [50], GWO [51], HS [52], WOA [53], ALO [54], AOA [55], and DA [56]. For the third battlefield, it is compared to SMO and other algorithms, such as AHA [57], AOS [58], ARO [59], SO [60], BWO [61], SCO [62], GBO [63], DMOA [64], DO [65], EO [66], FHO [67], WSO [68], and POA [69]. The results of the first two tests (battlefields 1 and 2) obtained using all 15 algorithms are shown in Tables 3–6, and those of the third test (battlefield 3) are shown in Tables 7 and 8.

Table 4, Table 6, and Table 8 present the statistical results obtained from the Wilcoxon's rank-sum nonparametric statistical test, which was performed with a significance level of 0.05. The best results and rankings are shown in bold. The comparison of the convergence behavior of SMOBHC with other algorithms, the boxplot illustrations, and the path planning results optimized by SMOBHC and seven other competitive algorithms (selected as examples) are presented for Battlefield 1, Battlefield 2, and Battlefield 3.

The boxplot illustrations consist of 20 run results in each box, where the central line (drawn in red) represents the median, the edges of the box represent the 25th and 75th percentiles, and the whiskers extend to the most extreme data points that are not considered outliers. Outliers are plotted individually.

Figures 8–12 show the convergence behavior of SMOBHC compared to other algorithms, the boxplot illustrations, and the path planning results optimized by SMOBHC and seven other competitive algorithms for Battlefield 1. Figures 13–17 display the same information for Battlefield 2, while Figures 18–22 show the results for Battlefield 3 in the same order of illustration as in Battlefield 1.

Battlefield 1:

For the first experimental test related to the first battlefield, we can clearly see from Table 1 that the SMOBHC outperforms all its competitors in the best, worst, and mean values, except for the ALO algorithm in the worst value for the third-dimension case. Furthermore, the ALO algorithm had the best standard deviation values among competitors due to its adaptive boundary shrinking and elitism mechanisms that control its exploitation, as well as its use of a random walk and roulette wheel selection in its research process.

The SMOBHC also provides very competitive results and outperforms the SMO in most comparison indices (best, worst, mean, and standard deviation values). This result confirms that the integration of the BHC optimizer into the SMO algorithmic structure is useful. Specifically, the addition of the BHC optimizer to the SMO LLP, GLP, and LLD phases can significantly improve its global exploration and exploitation characteristics through the use of both $\mathcal{B}$-operator and $\mathcal{N}$-operator.

Figures 8–10 shows that the SMOBHC has an acceptable convergence speed (specifically for the $D = 30$ and $D = 60$ cases) compared to its competitors. The superiority of SMOBHC can also be seen from the boxplot graphs (Figure 11), which illustrate that SMOBHC has the smallest average value among other algorithms. The non-parametric statistical results, based on Wilcoxon's rank-sum test, indicated in Table 3 prove that SMOBHC performs better than all other algorithms in the three problem dimensions, except for the SMO in the third-dimension case. Figure 12 shows the UCAV flight paths obtained by SMOBHC, SMO, and six other algorithms (selected as examples) for the first-dimension problem. The optimized paths generated by SMOBHC are smooth and can significantly avoid threat areas with the lowest threat cost. It is clear from Figure 12, especially the paths relative to BA, GCMBO, and HS, that their calculated paths have poor quality in terms of stability and avoiding local optima. Other algorithms, such as DE, AOA, and DA, were able to find an acceptable optimal flight path, but with a lower quality compared to SMOBHC and SMO.

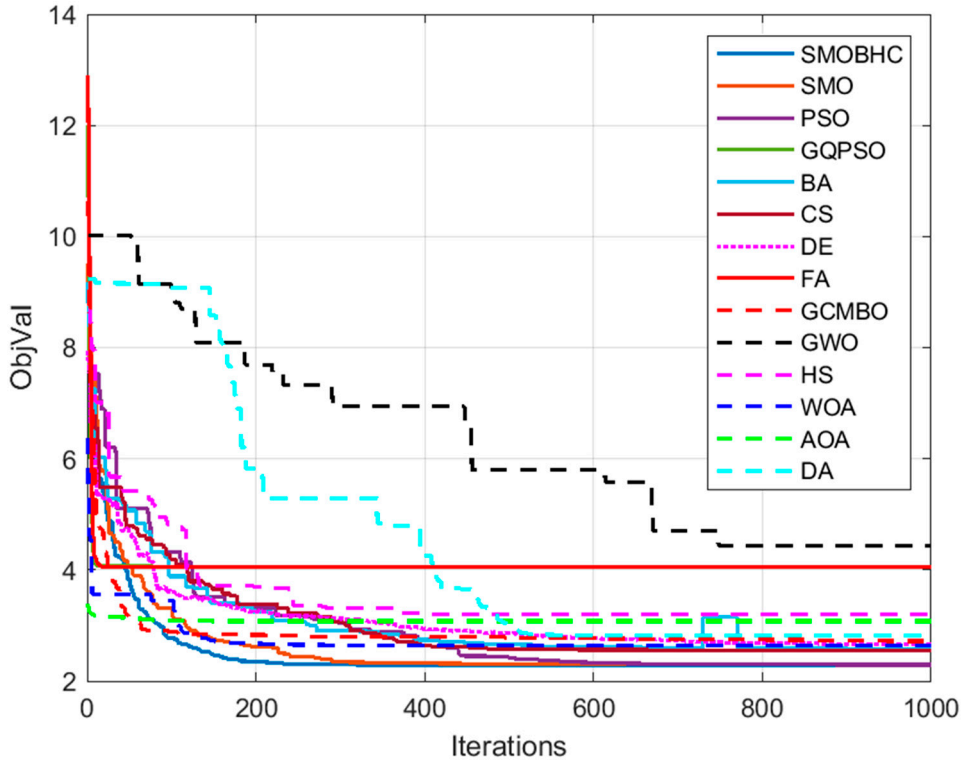

Figure 8. Comparative convergence curves of algorithms for battlefield 1 with $D = 30$.

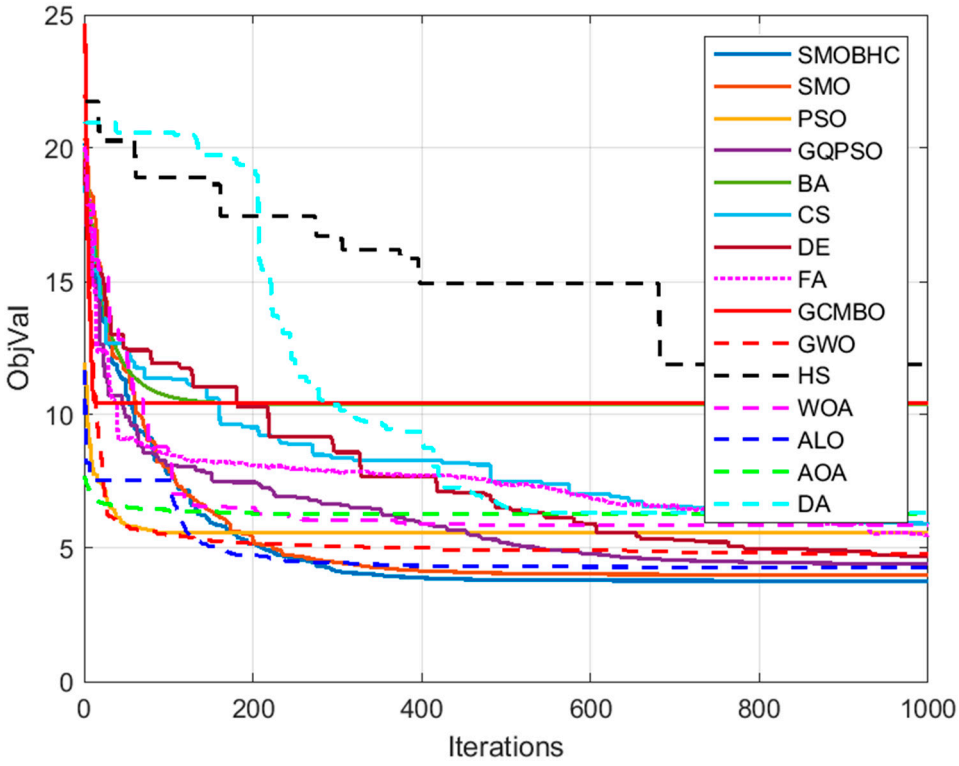

Figure 9. Comparative convergence curves of algorithms for battlefield 1 with $D = 60$.

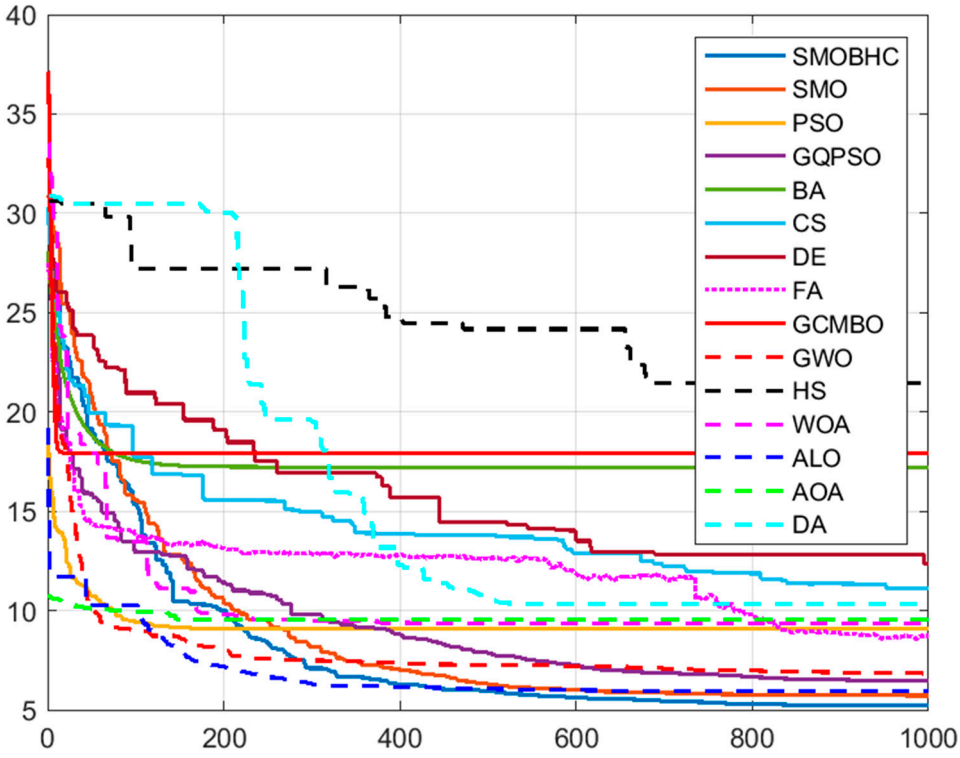

Figure 10. Comparative convergence curves of algorithms for battlefield 1 with $D = 90$.

Table 3. Experimental results obtained for the fifteen experimented algorithms relative to three different problem dimensions equal respectively to 30, 60, and 90 for the first battlefield composed of ten threats.

D	Results	SMOBHC	SMO	PSO	GQPSO	BA	CS	DE	FA	GCMBO	GWO	HS	WOA	ALO	AOA	DA
30	Best	**2.28947e+00**	2.29310e+00	2.75424e+00	2.29066e+00	3.92196e+00	2.57876e+00	2.55853e+00	2.65663e+00	4.05979e+00	2.71719e+00	4.44264e+00	3.20219e+00	2.63780e+00	3.07582e+00	2.82583e+00
	Worst	**2.61597e+00**	2.71327e+00	3.22385e+00	4.61145e+00	5.29548e+00	2.88037e+00	2.99855e+00	3.72796e+00	6.17494e+00	3.23827e+00	5.66178e+00	3.99601e+00	2.86127e+00	3.44006e+00	4.34395e+00
	Mean	**2.43943e+00**	2.53647e+00	2.93352e+00	2.94125e+00	4.69916e+00	2.65740e+00	2.66122e+00	3.06191e+00	4.98465e+00	2.87950e+00	5.21530e+00	3.62562e+00	2.67352e+00	3.21786e+00	3.59273e+00
	Std.	1.36303e-01	1.17987e-01	1.13051e-01	6.03611e-01	4.82167e-01	8.13444e-02	1.23028e-01	2.90825e-01	4.23119e-01	1.43531e-01	2.94617e-01	2.39591e-01	**7.52754e-02**	9.50957e-02	4.16094e-01
60	Best	**3.77426e+00**	3.99230e+00	5.61468e+00	4.41786e+00	1.04104e+01	5.87736e+00	4.71361e+00	5.49910e+00	5.49910e+00	4.77953e+00	1.19106e+01	5.86641e+00	4.29269e+00	6.27408e+00	6.32505e+00
	Worst	**4.27692e+00**	4.53576e+00	6.45057e+00	8.40310e+00	1.40119e+01	7.81696e+00	8.08800e+00	7.43200e+00	7.43200e+00	5.69980e+00	1.42811e+01	7.74777e+00	4.50942e+00	6.59945e+00	9.00176e+00
	Mean	**4.06937e+00**	4.21641e+00	5.97044e+00	7.45684e+00	1.18586e+01	6.65323e+00	6.05573e+00	6.52296e+00	6.52296e+00	5.15119e+00	1.32566e+01	6.69630e+00	4.31833e+00	6.44717e+00	7.32161e+00
	Std.	1.29764e-01	1.49738e-01	2.28953e-01	1.54426e+00	9.28414e-01	5.57735e-01	8.38343e-01	6.33541e-01	6.33541e-01	2.62726e-01	6.23371e-01	4.79265e-01	**6.27870e-02**	9.40193e-02	7.25221e-01
90	Best	**5.20185e+00**	5.72008e+00	9.06882e+00	6.46893e+00	1.72251e+01	1.10881e+01	1.23797e+01	8.72690e+00	8.72690e+00	6.83429e+00	2.14295e+01	9.34339e+00	5.94606e+00	9.53456e+00	1.03423e+01
	Worst	6.77599e+00	6.70230e+00	1.04167e+01	1.19491e+01	2.25479e+01	1.51550e+01	1.58565e+01	1.21446e+01	1.21446e+01	8.07238e+00	2.40816e+01	1.11104e+01	**6.22909e+00**	1.03621e+01	1.34498e+01
	Mean	**5.97088e+00**	6.10449e+00	9.71420e+00	1.05624e+01	2.00473e+01	1.31477e+01	1.44052e+01	9.69276e+00	9.69276e+00	7.39528e+00	2.24904e+01	1.01782e+01	6.01703e+00	9.86784e+00	1.17318e+01
	Std.	4.33765e-01	3.02698e-01	4.57557e-01	2.22147e+00	1.53747e+00	9.22082e-01	8.91705e-01	8.24435e-01	8.24435e-01	3.63992e-01	8.21393e-01	5.09086e-01	**8.62215e-02**	2.41530e-01	9.63863e-01

Table 4. Summarised Wilcoxon rank-sum comparisons between the SMOBHC algorithm as a reference and *fourteen* experimented algorithms for the first battlefield composed of ten threats.

D		SMOBHC vs. SMO	SMOBHC vs. PSO	SMOBHC vs. GQPSO	SMOBHC vs. BA	SMOBHC vs. CS	SMOBHC vs. DE	SMOBHC vs. FA	SMOBHC vs. GCMBO	SMOBHC vs. GWO	SMOBHC vs. HS	SMOBHC vs. WOA	SMOBHC vs. ALO	SMOBHC vs. AOA	SMOBHC vs. DA
30	p	3.3362e-03	6.7956e-08	8.2924e-05	6.7956e-08	1.6571e-07	2.7451e-04	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08
	h	1	1	1	1	1	1	1	1	1	1	1	1	1	1
60	p	4.3202e-03	6.7956e-08	6.7574e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08
	h	1	1	1	1	1	1	1	1	1	1	1	1	1	1
90	p	3.2348e-01	6.7956e-08	1.1772e-07	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08	7.3527e-01	6.7956e-08	6.7956e-08
	h	0	1	1	1	1	1	1	1	1	1	1	1	1	1
Diff (h = 1)		2	3	3	3	3	3	3	3	3	3	3	2	3	3
No diff (h = 0)		1	0	0	0	0	0	0	0	0	0	0	0	0	0
Best		**2**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**

Table 5. Experimental results obtained for the fifteen experimented algorithms relative to three different problem dimensions equal respectively to 30, 60, and 90 for the second battlefield composed of thirty threats.

D	Results	SMOBHC	SMO	PSO	GQPSO	BA	CS	DE	FA	GCMBO	GWO	HS	WOA	ALO	AOA	DA
30	Best	**2.51028e+00**	**2.51028e+00**	2.84819e+00	2.52220e+00	2.95020e+00	2.51845e+00	**2.51028e+00**	2.84445e+00	3.06962e+00	2.61220e+00	3.22308e+00	3.00608e+00	2.60448e+00	3.00800e+00	2.78753e+00
	Worst	**2.51477e+00**	2.51580e+00	3.04426e+00	2.71888e+00	3.78299e+00	2.65383e+00	2.72929e+00	3.16937e+00	3.77617e+00	2.85639e+00	3.61163e+00	3.90492e+00	2.96006e+00	3.26194e+00	3.74059e+00
	Mean	**2.51177e+00**	2.51184e+00	2.91253e+00	2.60964e+00	3.33122e+00	2.57734e+00	2.59191e+00	2.97449e+00	3.45475e+00	2.75032e+00	3.39670e+00	3.45755e+00	2.75710e+00	3.12795e+00	3.23671e+00
	Std.	**1.80077e-03**	1.80160e-03	5.15307e-02	3.21338e-02	2.33012e-01	4.46921e-02	5.79417e-02	8.75677e-02	1.66528e-01	6.27334e-02	1.01415e-01	1.86706e-01	1.15131e-01	5.81913e-02	2.04256e-01
60	Best	**3.83595e+00**	3.83733e+00	5.14681e+00	4.00704e+00	6.25087e+00	4.53306e+00	4.05029e+00	5.53866e+00	6.31199e+00	4.45724e+00	6.93861e+00	5.92446e+00	4.61219e+00	5.82414e+00	5.60451e+00
	Worst	**4.09095e+00**	4.12617e+00	5.74427e+00	6.84385e+00	7.89760e+00	5.55078e+00	5.31220e+00	6.68037e+00	7.79732e+00	5.11988e+00	7.98475e+00	7.10324e+00	5.04743e+00	6.52705e+00	7.15782e+00
	Mean	**3.93090e+00**	3.96579e+00	5.36391e+00	4.46120e+00	7.09673e+00	5.08951e+00	4.57179e+00	5.99368e+00	6.99630e+00	4.77400e+00	7.50167e+00	6.48509e+00	4.75926e+00	6.16272e+00	6.43501e+00
	Std.	**7.73957e-02**	9.64438e-02	1.32390e-01	7.97121e-01	4.34957e-01	2.80242e-01	2.95791e-01	2.49162e-01	4.45219e-01	1.83088e-01	2.77829e-01	3.82256e-01	1.30515e-01	2.31844e-01	3.63704e-01
90	Best	**5.11541e+00**	5.15118e+00	7.55355e+00	5.56467e+00	1.06711e+01	7.36641e+00	8.14985e+00	7.79696e+00	9.64809e+00	6.34746e+00	1.15298e+01	8.46152e+00	6.39245e+00	8.95448e+00	9.04897e+00
	Worst	**5.41980e+00**	5.48288e+00	8.72640e+00	1.18545e+01	1.27817e+01	9.79593e+00	1.00464e+01	1.09265e+01	1.27122e+01	7.53783e+00	1.31188e+01	1.08261e+01	6.91228e+00	9.95324e+00	1.11885e+01
	Mean	**5.26593e+00**	5.34675e+00	8.13438e+00	9.39583e+00	1.14995e+01	8.77740e+00	8.95983e+00	9.00075e+00	1.10847e+01	6.88814e+00	1.23627e+01	9.39447e+00	6.58353e+00	9.57711e+00	1.02167e+01
	Std.	1.23101e-01	**7.47452e-02**	2.67716e-01	1.80024e+00	6.84311e-01	5.45987e-01	5.44965e-01	7.38526e-01	7.58873e-01	2.79571e-01	4.12738e-01	5.80251e-01	1.32454e-01	3.06967e-01	5.35252e-01

Table 6. Summarised Wilcoxon rank-sum comparisons between the SMOBHC algorithm as a reference and *fourteen* experimented algorithms for the second battlefield composed of thirty threats.

D		SMOBHC vs. SMO	SMOBHC vs. PSO	SMOBHC vs. GQPSO	SMOBHC vs. BA	SMOBHC vs. CS	SMOBHC vs. DE	SMOBHC vs. FA	SMOBHC vs. GCMBO	SMOBHC vs. GWO	SMOBHC vs. HS	SMOBHC vs. WOA	SMOBHC vs. ALO	SMOBHC vs. AOA	SMOBHC vs. DA
30	p	2.9768e-01	6.7956e-08	6.7956e-08	6.7956e-08	2.3557e-06	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08
	h	0	1	1	1	1	1	1	1	1	1	1	1	1	1
60	p	1.9883e-01	6.7956e-08	1.5997e-05	6.7956e-08	6.7956e-08	1.0646e-07	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08
	h	1	1	1	1	1	1	1	1	1	1	1	1	1	1
90	p	9.6196e-02	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08
	h	0	1	1	1	1	1	1	1	1	1	1	1	1	1
Diff (h = 1)		0	3	3	3	3	3	3	3	3	3	3	3	3	3
No diff (h = 0)		3	0	0	0	0	0	0	0	0	0	0	0	0	0
Best		0	3	3	3	3	3	3	3	3	3	3	3	3	3

Table 7. Experimental results obtained for the fifteen experimented algorithms relative to three different problem dimensions equal respectively to 30, 60, and 90 for the third battlefield composed of twenty threats.

D	Results	SMOBHC	SMO	AHA	AOS	ARO	SO	BWO	SCO	GBO	DMOA	DO	EO	FHO	WSO	PDO
30	Best	**2.42035e+00**	**2.42035e+00**	2.74338e+00	2.74393e+00	2.74339e+00	2.42839e+00	2.90601e+00	4.78941e+00	2.73036e+00	2.48093e+00	2.44246e+00	2.63792e+00	2.86500e+00	2.77533e+00	3.18897e+00
	Worst	**2.57116e+00**	2.57117e+00	2.77586e+00	3.17199e+00	2.93080e+00	2.83734e+00	2.99695e+00	4.11803e+00	2.87057e+00	3.22760e+00	2.78509e+00	2.92475e+00	3.02361e+00	3.10438e+00	4.23746e+00
	Mean	**2.42963e+00**	2.44267e+00	2.75207e+00	2.90602e+00	2.76218e+00	2.62672e+00	2.96914e+00	4.62706e+00	2.76473e+00	2.73652e+00	2.67790e+00	2.74856e+00	2.96439e+00	2.95392e+00	3.68263e+00
	Std.	3.35353e-02	4.85828e-02	**8.11277e-03**	1.41608e-01	4.09533e-02	1.19591e-01	1.89275e-02	1.93943e-01	3.34833e-02	1.96120e-01	8.77372e-02	6.73683e-02	4.49008e-02	1.04436e-01	2.76668e-01
60	Best	**3.70782e+00**	3.71227e+00	4.51038e+00	4.63060e+00	4.51131e+00	3.86218e+00	5.11575e+00	7.95593e+00	4.50639e+00	8.76551e+00	4.25750e+00	4.49766e+00	5.12686e+00	4.91835e+00	6.23948e+00
	Worst	**4.15695e+00**	4.25617e+00	4.55403e+00	5.22236e+00	4.91454e+00	4.84537e+00	5.16204e+00	8.86259e+00	5.02363e+00	1.10561e+01	4.86091e+00	4.89462e+00	5.21059e+00	5.84186e+00	8.64117e+00
	Mean	**3.87064e+00**	3.95329e+00	4.53216e+00	4.90928e+00	4.58364e+00	4.40985e+00	5.14784e+00	8.74873e+00	4.62033e+00	1.02349e+01	4.58760e+00	4.63640e+00	5.18204e+00	5.40936e+00	7.42693e+00
	Std.	1.45410e-01	1.67976e-01	1.29237e-02	2.00486e-01	9.84922e-02	2.45296e-01	**1.26257e-02**	2.06429e-01	1.31735e-01	5.48412e-01	1.90264e-01	1.26067e-01	2.27147e-02	2.73430e-01	7.61039e-01
90	Best	**5.03620e+00**	5.17218e+00	6.29085e+00	6.63783e+00	6.26632e+00	6.45604e+00	7.24355e+00	1.19037e+01	6.25195e+00	1.86112e+01	6.01061e+00	6.24270e+00	7.27208e+00	7.56239e+000	1.03499e+01
	Worst	**5.61129e+00**	6.10704e+00	6.56629e+00	7.17214e+00	6.79300e+00	7.34307e+00	7.29806e+00	1.26385e+01	6.70000e+00	2.05484e+01	6.98320e+00	7.02330e+00	7.33953e+00	8.62073e+00	1.22534e+01
	Mean	**5.36272e+00**	5.47085e+00	6.43876e+00	6.86873e+00	6.44597e+00	6.83742e+00	7.28132e+00	1.24897e+01	6.37801e+00	1.97401e+01	6.45996e+00	6.54829e+00	7.32062e+00	8.21006e+00	1.10281e+01
	Std.	1.30557e-01	2.02255e-01	8.43281e-02	1.65042e-01	1.22716e-01	2.47851e-01	**1.25090e-02**	2.19932e-01	1.22234e-01	6.45794e-01	2.74162e-01	1.86834e-01	1.83397e-02	3.13058e-01	5.36196e-01

Table 8. Summarised Wilcoxon rank-sum comparisons between the SMOBHC algorithm as a reference and *fourteen* experimented algorithms for the third battlefield composed of twenty threats.

D		SMOBHC vs. SMO	SMOBHC vs. AHA	SMOBHC vs. AOS	SMOBHC vs. ARO	SMOBHC vs. SO	SMOBHC vs. BWO	SMOBHC vs. SCO	SMOBHC vs. GBO	SMOBHC vs. DMOA	SMOBHC vs. DO	SMOBHC vs. EO	SMOBHC vs. FHO	SMOBHC vs. WSO	SMOBHC vs. PDO
30	p	3.9417e-01	6.7956e-08	6.7860e-08	6.7956e-08	2.2178e-07	6.7956e-08	6.7956e-08	6.7956e-08	1.0646e-07	9.1728e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08
	h	0	1	1	1	1	1	1	1	1	1	1	1	1	1
60	p	1.3328e-01	6.7956e-08	6.7956e-08	6.7956e-08	2.9598e-07	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08
	h	0	1	1	1	1	1	1	1	1	1	1	1	1	1
90	p	6.7868e-02	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08	6.7956e-08
	h	0	1	1	1	1	1	1	1	1	1	1	1	1	1
Diff (**h** =1)		0	3	3	3	3	3	3	3	3	3	3	3	3	3
No diff (**h** =0)		3	0	0	0	0	0	0	0	0	0	0	0	0	0
Best		0	3	3	3	3	3	3	3	3	3	3	3	3	3

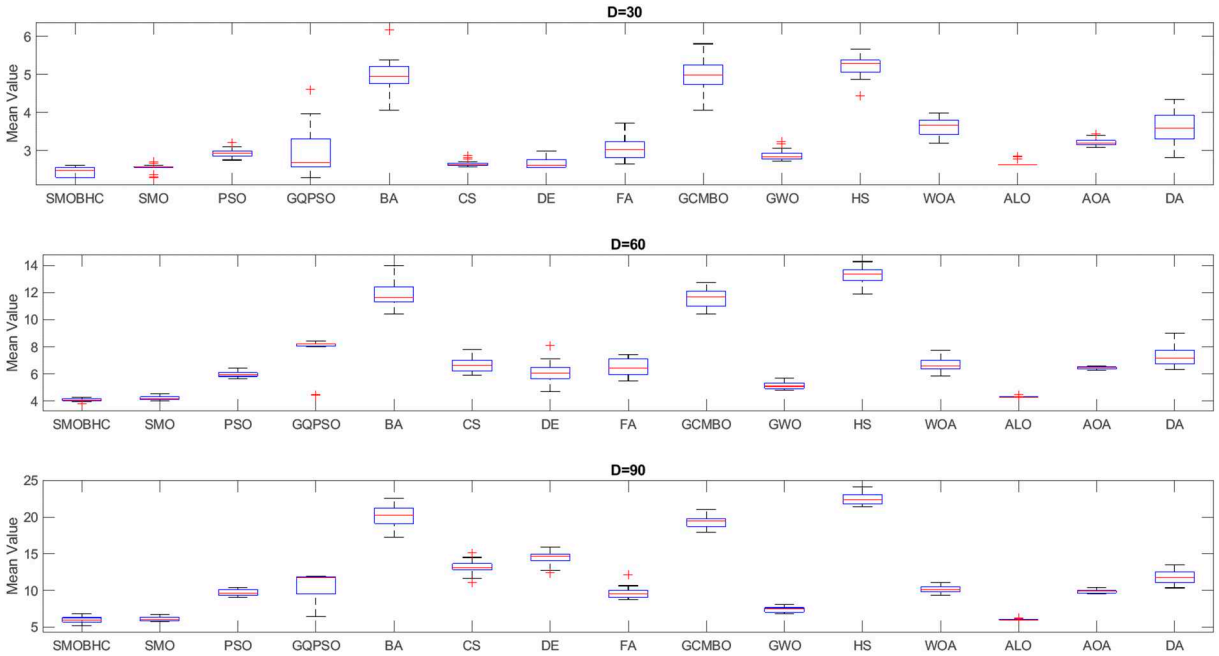

Figure 11. Boxplot graphs of mean values relative to all algorithms for battlefield 1 with $D = 30$, 60, and 90.

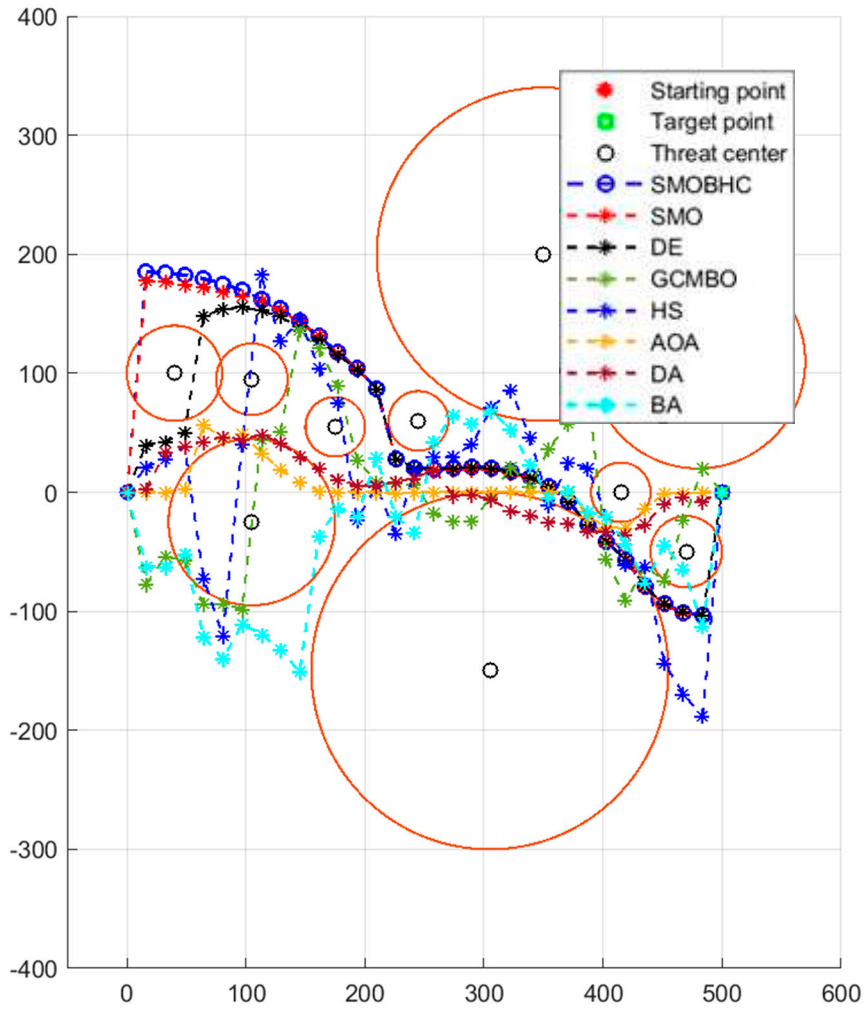

Figure 12. Comparison between SMOBHC and other seven algorithms for battlefield 1 with $D = 30$.

Battlefield 2:

For this experimental test, the battlefield subject of the examination has an increased number of threats, totaling thirty elements distributed over a domain with a length of 1000 units. As seen in Table 5, SMOBHC clearly outperforms all its competitors' algorithms in terms of the best, worst, mean, and std. values, except for the std. value obtained by SMO for the third-dimension problem. Statistically, SMOBHC outperforms all examined methods except SMO, where there is no significant difference between them. Figures 13–15 demonstrates that SMOBHC converges faster to the optimal flight path than all other algorithms for a 30-dimensional problem, but as the dimension increases to 60 and 90, the convergence speed of all algorithms decreases, with SMOBHC having the lowest. Additionally, the final result is better than all competitors. The boxplot illustrations in Figure 16 show the dominance of SMOBHC over its competitor algorithms. Figure 17 illustrates the flight paths of SMOBHC, SMO, and six other algorithms (selected as examples) for the third-dimension problem on the battlefield. The first observation is that the SMOBHC-optimized flight path is superior to all tested methods in terms of flight path stability and avoidance of local optima. On the other hand, SMO is trapped in a local optimum, similar to PSO and GWO, while BA, CS, and HS algorithms fail to produce an acceptable quality flight path.

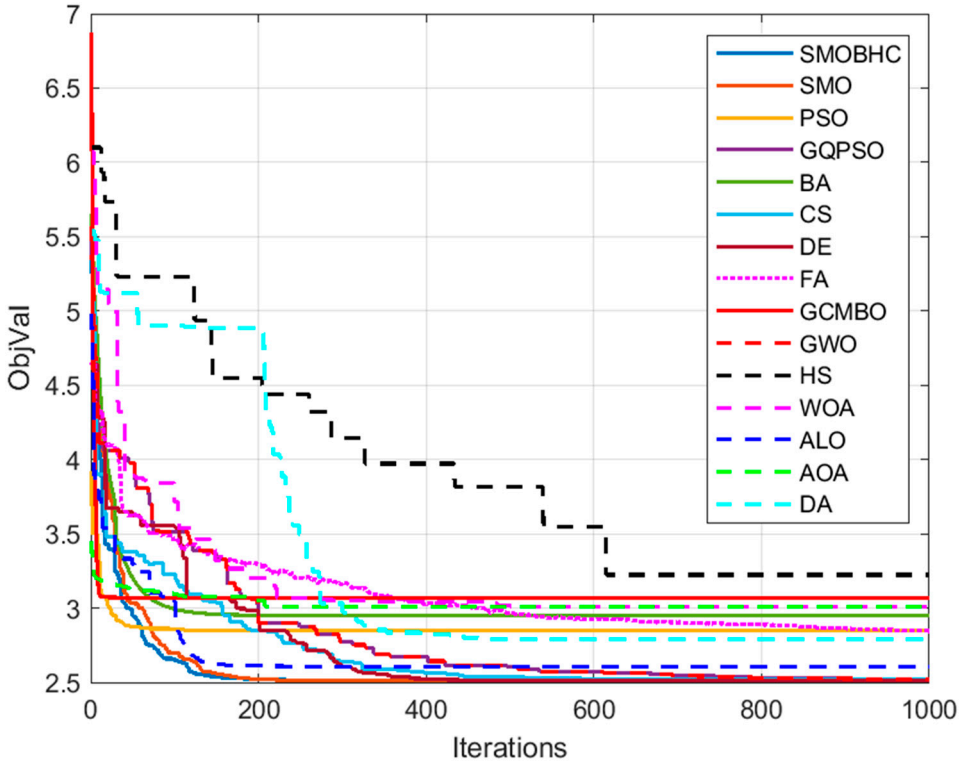

Figure 13. Comparative convergence curves of algorithms for battlefield 2 with $D = 30$.

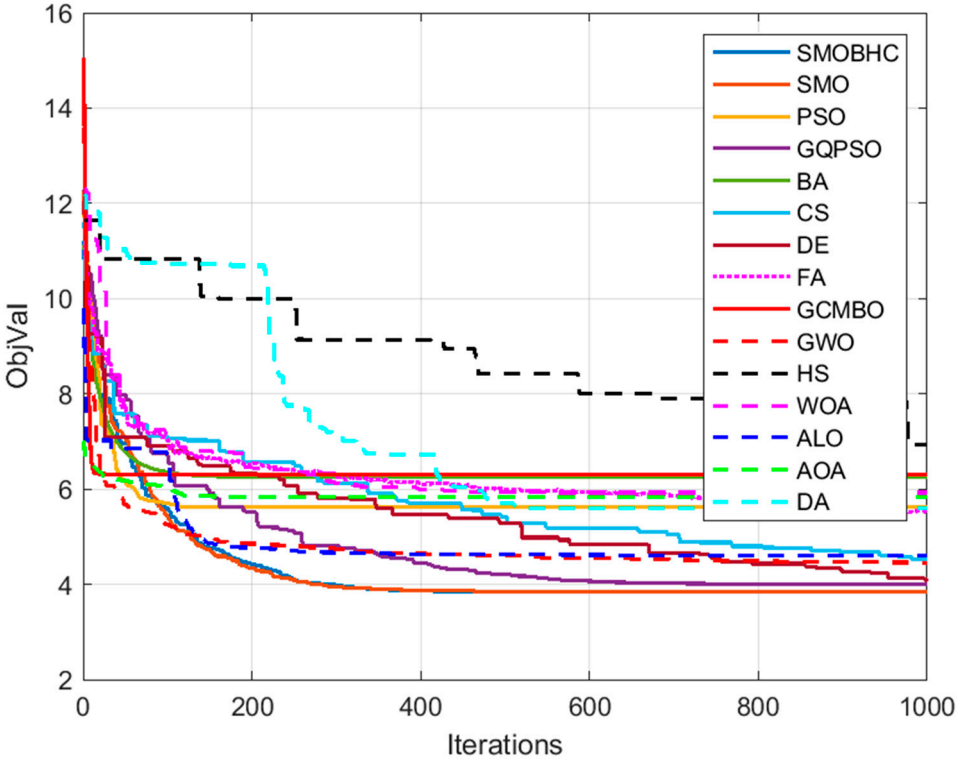

Figure 14. Comparative convergence curves of algorithms for battlefield 2 with $D = 60$.

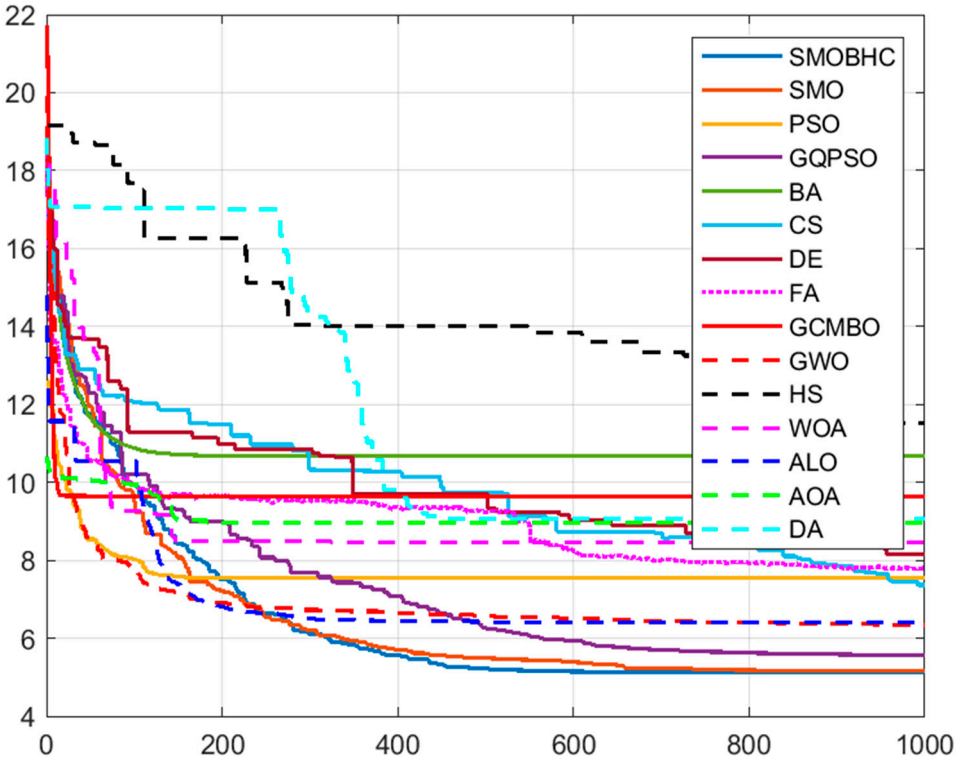

Figure 15. Comparative convergence curves of algorithms for battlefield 2 with $D = 90$.

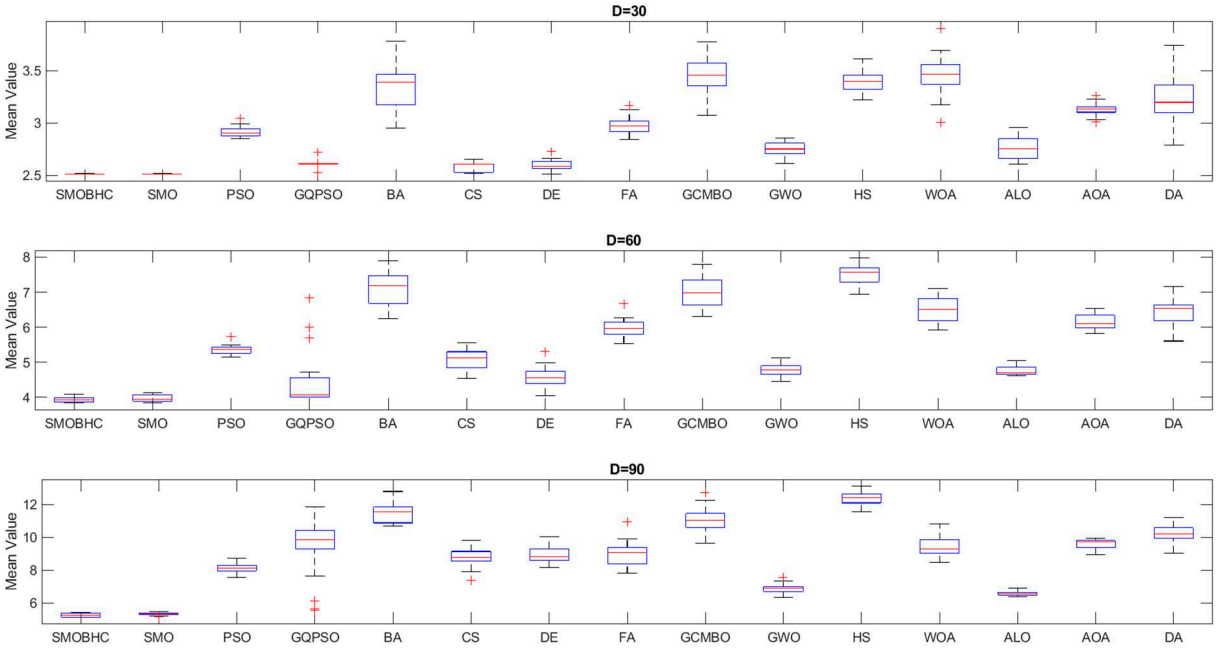

Figure 16. Boxplot graphs of mean values relative to all algorithms for battlefield 2 with $D = 30$, 60, and 90.

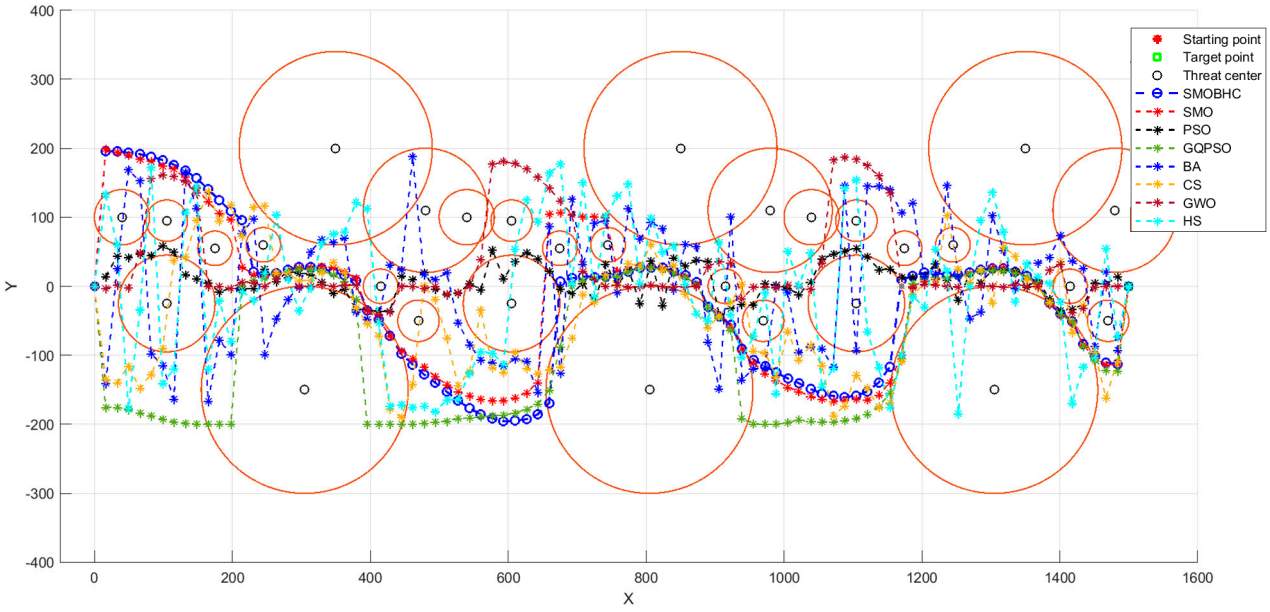

Figure 17. Comparison between SMOBHC and other seven algorithms for battlefield 2 with $D = 90$.

Battlefield 3:

For the next experimental test related to the third battlefield consisting of twenty threats, the proposed algorithm outperforms all its competitors in the best, worst, and mean values, as shown in Table 7. In regards to the obtained standard deviation values, the AHA algorithm performs better for the first-dimension case and the BWO algorithm for both the second- and third-dimension cases. The SMOBHC has a relatively slow convergence speed, but its curve continuously converges to a better fitness value compared to all other algorithms. As shown in Table 8, the SMOBHC displays superior performance to all other algorithms except the SMO, where there is no significant difference between them. The boxplot illustrations in Figure 21 indicate the superiority of SMOBHC over its

competitors, while Figure 22 shows the flight paths produced by SMOBHC, SMO, and six other algorithms for the second-dimension problem with a small modification to the starting and ending points. Despite the change in flight conditions, SMOBHC and SMO still produced optimal flight paths of good quality.

In particular, the optimized flight paths of SMOBHC and SMO have minimal collisions with the battlefield threat areas, unlike some cases like DMOA and PDO, which have a clear insufficiency in problem treatment as seen in their optimized flight paths. Meanwhile, the paths related to the ASO, SO, DO, and EO algorithms are all trapped in local optima.

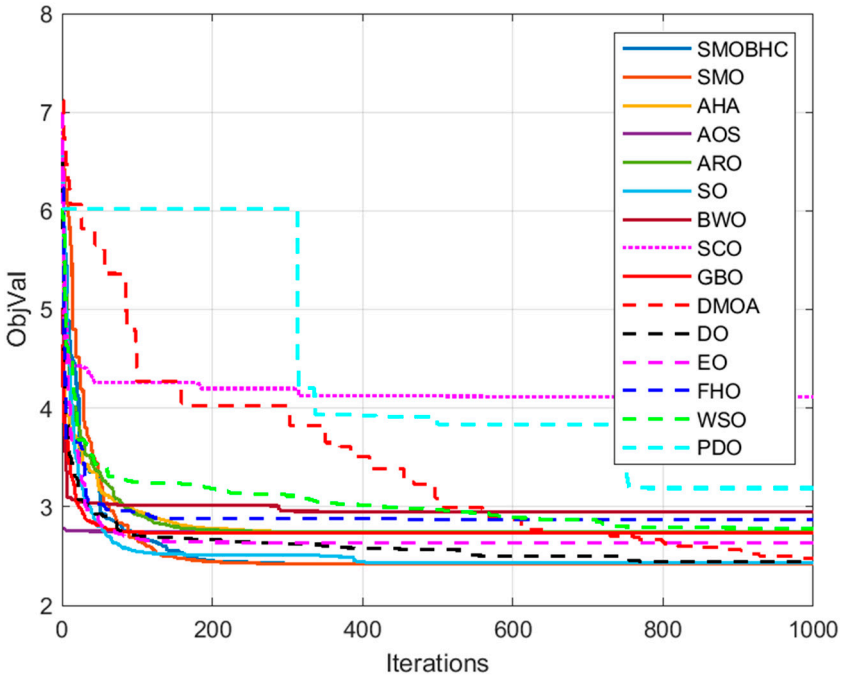

Figure 18. Comparative convergence curves of algorithms for battlefield 3 with $D = 30$.

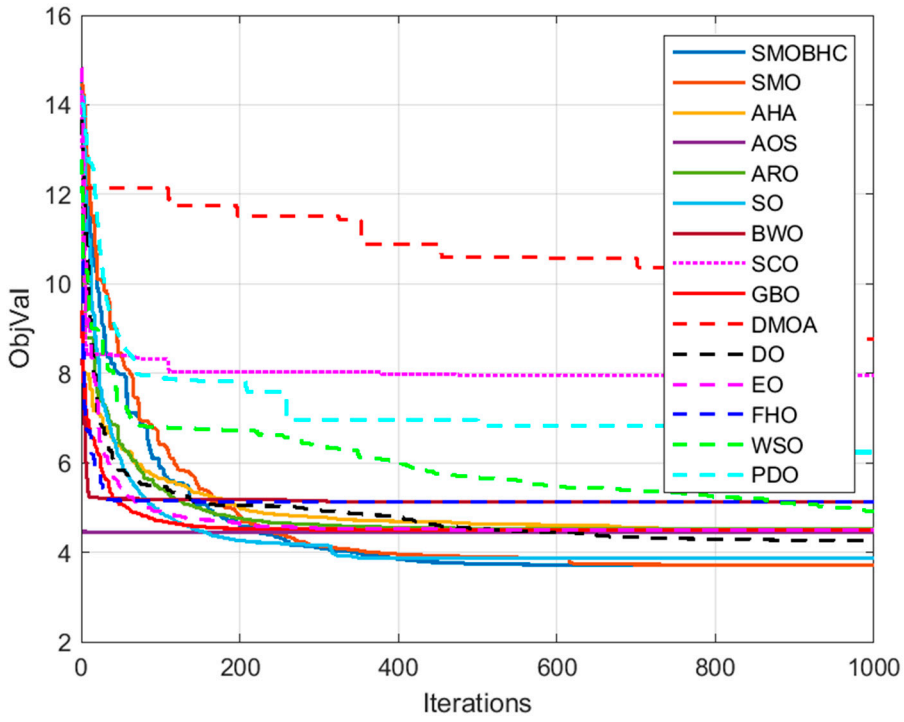

Figure 19. Comparative convergence curves of algorithms for battlefield 3 with $D = 60$.

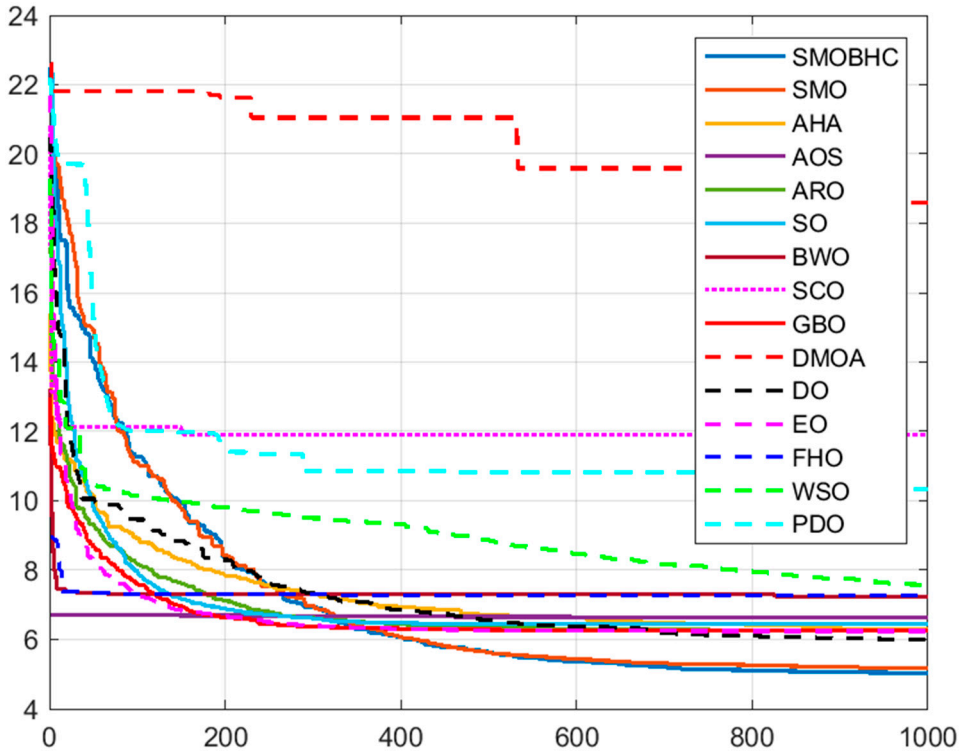

Figure 20. Comparative convergence curves of algorithms for battlefield 3 with $D = 90$.

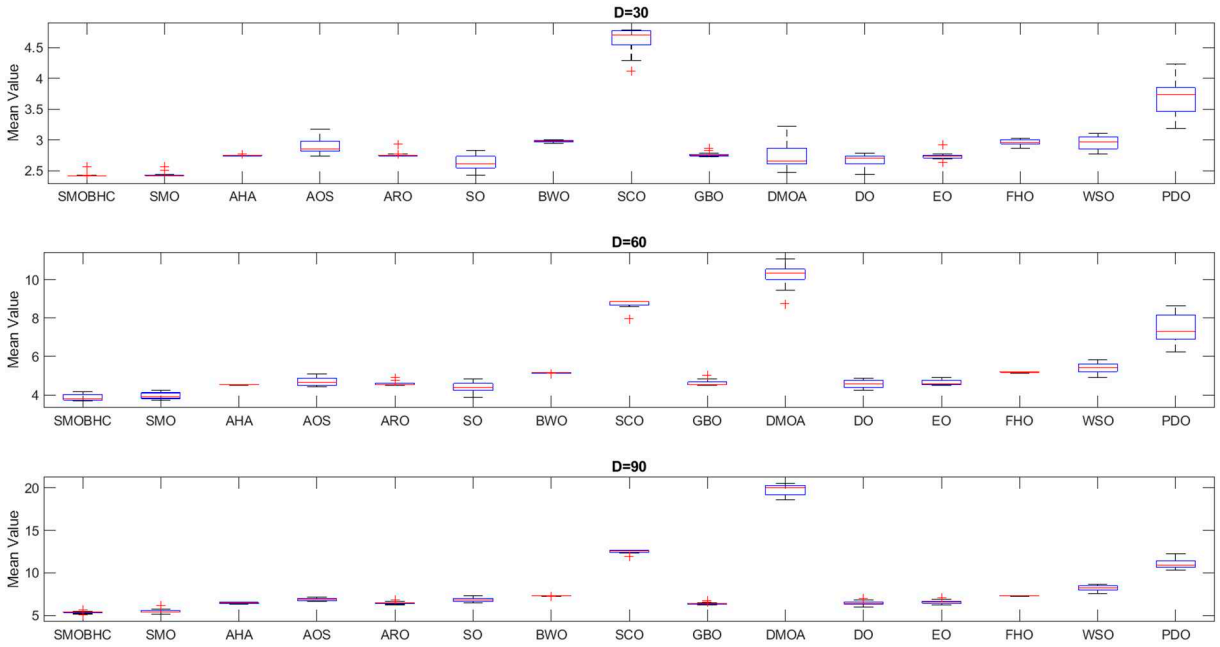

Figure 21. Boxplot graphs of mean values relative to all algorithms for battlefield 3 with $D = 30$, 60, and 90.

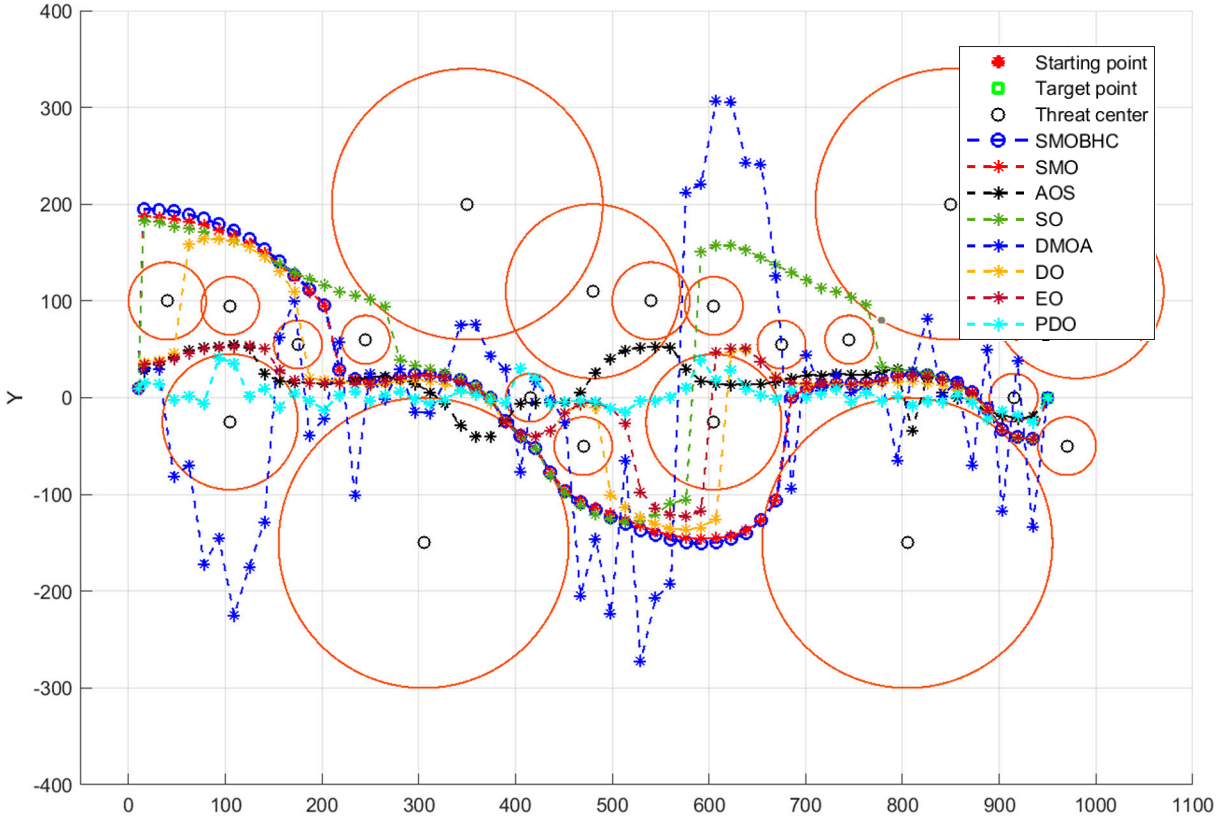

Figure 22. Comparison between SMOBHC and other seven algorithms for battlefield 3 with $D = 60$.

The integration of the basic SMO algorithm with the BHC optimizer has resulted in the creation of a new SMO variant, referred to as SMOBHC. This hybrid algorithm boasts a simple yet effective structure and has been shown to outperform a recent class of meta-heuristic algorithms in solving the path planning problem for UCAVs of varying scales.

The combination of these two algorithms results in a powerful tool for addressing the complex and challenging task of UCAV path planning. The significance of the SMOBHC algorithm lies in its ability to provide optimal solutions for the path planning problem with a high degree of accuracy and efficiency, outperforming other contemporary methods. These findings make SMOBHC a valuable contribution to the field of UCAV path planning and optimization.

6. Conclusions

In this paper, a novel variant of the Spider Monkey Optimization (SMO) algorithm is proposed and investigated for UCAV path planning. Our proposed algorithm combines the Beta-Hill Climbing Optimizer (BHC) with the basic version of the SMO algorithm to improve its exploration and exploitation capabilities and convergence behavior. The hybridization mechanism uses the BHC optimizer in its standard form to first improve each new Spider Monkey solution (position) generated in the SMO Local Leader Phase (LLP), then enhance each new solution produced in the SMO Global Leader Phase (GLP), and finally update the positions of all Local Leader members of each local group under a specific condition in the SMO Local Leader Decision (LLD) phase.

Experimental results show that our proposed SMO variant, called SMOBHC, is more competitive than twenty state-of-the-art evolutionary algorithms, including the standard SMO, in terms of the quality and stability of the final generated paths for UCAV path planning. SMOBHC outperforms almost all competing algorithms in terms of accuracy (best, worst, mean, and std. values) for the three battlefields under different problem dimensions (30, 60, and 90 respectively). Statistically, SMOBHC performs better than all

its rival algorithms for the three battlefields except SMO, where there is no significant difference between them. In terms of convergence behavior, SMOBHC has an acceptable convergence speed compared to its competitors, especially for the first battlefield with problem dimensions equal to 30 and 60.

Despite the promising results achieved in our study, it must be acknowledged that it also has several limitations. Firstly, our focus on UCAV path planning within a 2D space can be considered a constraint. Secondly, the method's limited applicability in a static environment is another hindrance. Lastly, the computational demands of the method can prove to be challenging. However, in light of these limitations, we are motivated to explore the potential for further improvement. Our plan is to develop a more comprehensive 2D mathematical model for UCAV path planning, which we believe will help us better understand the advantages of the SMO algorithm and its variants. With this new model, we aim to create an even more advanced algorithm and rigorously test it using a newly developed 3D UCAV battlefield model that takes into account both static and dynamic threats. Our ultimate goal is to provide a more comprehensive solution for UCAV path planning in complex and challenging environments.

Author Contributions: Conceptualization, F.A., A.A. and X.-Z.G.; methodology, F.A. and S.B.; software, F.A., S.B. and X.-Z.G.; validation, F.A., X.-Z.G. and I.B.; formal analysis, F.A., A.A. and S.B.; investigation, F.A., S.B. and I.B.; writing—original draft preparation, F.A., N.K. and F.L.; writing—review and editing, F.A., N.K. and F.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

Appendix A

Table A1. Variables, abbreviations, and adjustable control variables signification.

Parameter	Signification
D	Number of steps
ST	The points
δ	Parameter that determines the form of the density function
R_i	The radius of the ith threat element
γ	A weight coefficient ranging in [0, 1]
$\|\|ST\|\|$	The length of the segment ST
n_{thr}	Total number of threats in the battlefield
FFS	Fission–Fusion System of spider monkeys
LLP	Local Leader Phase
GLP	Global Leader Phase
LLD	Local Leader Decision phase
MG	Maximum Group limit
GLL	Global Leader Limit
LLL	Local Leader Limit

Table A1. *Cont.*

Parameter	Signification
pr	Perturbation rate
N	Spider Monkey positions number
$GroupNumber$	Group number counter (number of groups)
b_w	$\mathcal{N}$-operator bandwidth
$\mathcal{B}$	$\mathcal{B}$-operator probability
$Iter_{max}$	Maximum number of iterations
Run_{max}	Maximum number of runs

References

1. Elmokadem, T.; Savkin, A.V. Towards fully autonomous UAVs: A survey. *Sensors* **2021**, *21*, 6223. [CrossRef]
2. Altan, A.; Hacıoglu, R. Model predictive control of three-axis gimbal system mounted on UAV for real-time target tracking underexternal disturbances. *Mech. Syst. Signal Process.* **2020**, *138*, 106548. [CrossRef]
3. Saadi, A.A.; Soukane, A.; Meraihi, Y.; Gabis, A.B.; Mirjalili, S.; Ramdane-Cherif, A. UAV Path Planning Using Optimization Approaches: A Survey. *Arch. Comput. Methods Eng.* **2022**, *29*, 4233–4284. [CrossRef]
4. Belge, E.; Altan, A.; Hacıoglu, R. Metaheuristic Optimization-Based Path Planning and Tracking of Quadcopter for Payload Hold-Release Mission. *Electronics* **2022**, *11*, 1208. [CrossRef]
5. Bansal, J.C.; Sharma, H.; Jadon, S.S.; Clerc, M. Spider monkey optimization algorithm for numerical optimization. *Memet. Comput.* **2014**, *6*, 31–47. [CrossRef]
6. Zhu, H.; Wang, Y.; Li, X. UCAV path planning for avoiding obstacles using cooperative co-evolution Spider Monkey Optimization. *Knowl. Based Syst.* **2022**, *246*, 108713. [CrossRef]
7. Zhu, H.; Wang, Y.; Ma, Z.; Li, X. A comparative study of swarm intelligence algorithms for UCAV path-planning problems. *Mathematics* **2021**, *9*, 171. [CrossRef]
8. Guo, T.; Jiang, N.; Li, B.Y. UAV navigation in high dynamic environments: A deep reinforcement learning approach. *Chin. J. Aeronaut.* **2021**, *34*, 479–489. [CrossRef]
9. Huang, H.; Savkin, A.V. Autonomous Navigation of a Solar-Powered UAV for Secure Communication in Urban Environments with Eavesdropping Avoidance. *Future Internet* **2020**, *12*, 170. [CrossRef]
10. Huang, H.; Savkin, A.V.; Ni, W. Energy-Efficient 3D Navigation of a Solar-Powered UAV for Secure Communication in the Presence of Eavesdroppers and No-Fly Zones. *Energies* **2020**, *13*, 1445. [CrossRef]
11. Isik, O.K.; Hong, J.; Petrunin, I.; Tsourdos, A. Integrity analysis for GPS-based navigation of UAVs in urban environment. *Robotics* **2020**, *9*, 66. [CrossRef]
12. Zhang, S.; Li, Y.; Dong, Q. Autonomous navigation of UAV in multi-obstacle environments based on a Deep Reinforcement Learning approach. *Appl. Soft Comput.* **2022**, *115*, 108194. [CrossRef]
13. Delamer, J.-A.; Watanabe, Y.; Chanel, C.P.C. Safe path planning for UAV urban operation under GNSS signal occlusion risk. *Rob. Auton. Syst.* **2021**, *142*, 103800. [CrossRef]
14. He, L.; Aouf, N.; Song, B. Explainable Deep Reinforcement Learning for UAV autonomous path planning. *Aerosp. Sci. Technol.* **2021**, *118*, 107052. [CrossRef]
15. Al-Kabi, H.; Mazinani, S.M. DNCS: New UAV navigation with considering the no-fly zone and efficient selection of the charging station. *Ain Shams Eng. J.* **2021**, *12*, 3669–3676. [CrossRef]
16. Wang, S.; Zhan, X.; Zhai, Y.; Chi, C.; Shen, J. Highly reliable relative navigation for multi-UAV formation flight in urban environments. *Chin. J. Aeronaut.* **2020**, in press. [CrossRef]
17. Zhu, X.; Wang, L.; Li, Y.; Song, S.; Ma, S.; Yang, F.; Zhai, L. Path planning of multi-UAVs based on deep Q-network for energy-efficient data collection in UAVs-assisted IoT. *Veh. Commun.* **2022**, *36*, 100491. [CrossRef]
18. Li, J.; Yang, G.; Cai, Q.; Niu, H.; Li, J. Cooperative navigation for UAVs in GNSS-denied area based on optimized belief propagation. *Meas. J. Int. Meas. Confed.* **2022**, *192*, 110797. [CrossRef]
19. Hoon, L.M.; Moon, J. Deep reinforcement learning-based model-free path planning and collision avoidance for UAVs: A soft actor–critic with hindsight experience replay approach. *ICT Express* **2022**, in press.
20. Habibi, H.; Safaei, A.; Voos, H.; Darouach, M.; Sanchez-Lopez, J.L. Safe navigation of a quadrotor UAV with uncertain dynamics and guaranteed collision avoidance using barrier Lyapunov function. *Aerosp. Sci. Technol.* **2023**, *132*, 108064. [CrossRef]
21. Chen, H.; Lu, P. Real-time identification and avoidance of simultaneous static and dynamic obstacles on point cloud for UAVs navigation. *Robot. Auton. Syst.* **2022**, *154*, 104124. [CrossRef]
22. El-Basioni, B.M.M.; El-Kader, S.M.A. Mission-based PTR triangle for multi-UAV systems flight planning. *Ad Hoc Netw.* **2023**, *142*, 103115. [CrossRef]

23. Wang, Y.; Liu, W.; Liu, J.; Sun, C. Cooperative USV–UAV marine search and rescue with visual navigation and reinforcement learning-based control. *ISA Trans.* **2023**, in press. [CrossRef] [PubMed]
24. Seo, D.; Kang, J. Collision-avoided Tracking Control of UAV Using Velocity-adaptive 3D Local Path Planning. *Int. J. Control Autom. Syst.* **2023**, *21*, 231–243. [CrossRef]
25. Li, T.; Zhang, X.; Zhang, S.; Zhang, X.; Chen, X. STUNS-Planner: A Spatiotemporal Motion Planner with Unbending and Consistency Awareness for Quadrotors in Unknown Environments. *J. Intell. Robot Syst.* **2023**, *107*, 7. [CrossRef]
26. Zhang, Y.; Wang, P.; Yang, L.; Liu, Y.; Lu, Y.; Zhu, X. Novel Swarm Intelligence Algorithm for Global Optimization and Multi-UAVs Cooperative Path Planning: Anas Platyrhynchos Optimizer. *Appl. Sci.* **2020**, *10*, 4821. [CrossRef]
27. Shi, K.; Zhang, X.; Xia, S. Multiple Swarm Fruit Fly Optimization Algorithm Based Path Planning Method for Multi-UAVs. *Appl. Sci.* **2020**, *10*, 2822. [CrossRef]
28. Nayeem, G.M.; Fan, M.; Li, S.; Ahammad, K. A Modified Particle Swarm Optimization for Autonomous UAV Path Planning in 3D Environment. In *Cyber Security and Computer Science*; Bhuiyan, T., Rahman, M.M., Ali, M.A., Eds.; ICONCS 2020. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering; Springer: Cham, Switzerland, 2020; Volume 325.
29. Ge, F.; Li, K.; Han, Y.; Xu, W.; Wang, Y. Path planning of UAV for oilfield inspections in a three-dimensional dynamic environment with moving obstacles based on an improved pigeon-inspired optimization algorithm. *Appl. Intell.* **2020**, *50*, 2800–2817. [CrossRef]
30. Chai, X.; Xiao, J.; Zheng, Z.; Zhang, L.; Qu, B.; Yan, L.; Sun, H. UAV 3D Path Planning Based on Multi-Population Ensemble Differential Evolution. In *Bio-Inspired Computing: Theories and Applications*; Pan, L., Liang, J., Qu, B., Eds.; BIC-TA 2019. Communications in Computer and Information Science; Springer: Singapore, 2020; p. 1159.
31. Mesquita, R.; Gaspar, P.D. A Novel Path Planning Optimization Algorithm Based on Particle Swarm Optimization for UAVs for Bird Monitoring and Repelling. *Processes* **2022**, *10*, 62. [CrossRef]
32. Xia, S.; Zhang, X. Constrained path planning for unmanned aerial vehicle in 3D terrain using modified multi-objective particle swarm optimization. *Actuators* **2021**, *10*, 255. [CrossRef]
33. Liu, H.; Ge, J.; Wang, Y.; Li, J.; Ding, K.; Zhang, Z.; Guo, Z.; Li, W.; Lan, J. Multi-UAV optimal mission assignment and path planning for disaster rescue using adaptive genetic algorithm and improved artificial bee colony method. *Actuators* **2022**, *11*, 4. [CrossRef]
34. Huo, L.S.; Zhu, J.H.; Li, Z.M.; Ma, M.H. A Hybrid Differential Symbiotic Organisms Search Algorithm for UAV Path Planning. *Sensors* **2021**, *21*, 3037. [CrossRef] [PubMed]
35. Tang, A.D.; Han, T.; Zhou, H.; Xie, L. An improved equilibrium optimizer with application in unmanned aerial vehicle path planning. *Sensors* **2021**, *21*, 1814. [CrossRef]
36. Ahmed, N.; Pawase, C.J.; Chang, K. Distributed 3-D Path Planning for Multi-UAVs with Full Area Surveillance Based on Particle Swarm Optimization. *Appl. Sci.* **2021**, *11*, 3417. [CrossRef]
37. Jarray, R.; Al-Dhaifallah, M.; Rezk, H.; Bouallègue, S. Parallel Cooperative Coevolutionary Grey Wolf Optimizer for Path Planning Problem of Unmanned Aerial Vehicles. *Sensors* **2022**, *22*, 1826. [CrossRef]
38. Zhang, C.; Liu, Y.; Hu, C. Path Planning with Time Windows for Multiple UAVs Based on Gray Wolf Algorithm. *Biomimetics* **2022**, *7*, 225. [CrossRef]
39. Chu, H.; Yi, J.; Yang, F. Chaos Particle Swarm Optimization Enhancement Algorithm for UAV Safe Path Planning. *Appl. Sci.* **2022**, *12*, 8977. [CrossRef]
40. Zhang, R.; Li, S.; Ding, Y.; Qin, X.; Xia, Q. UAV Path Planning Algorithm Based on Improved Harris Hawks Optimization. *Sensors* **2022**, *22*, 5232. [CrossRef] [PubMed]
41. Chen, H.; Zhou, X.; Ran, X.; Wang, J.; Chen, H.; Deng, W. Adaptive cylinder vector particle swarm optimization with differential evolution for UAV path planning. *Eng. Appl. Artif. Intell.* **2023**, *121*, 105942.
42. Yu, X.; Jiang, N.; Wang, X.; Li, M. A hybrid algorithm based on grey wolf optimizer and differential evolution for UAV path planning. *Expert Syst. Appl.* **2022**, *215*, 119327. [CrossRef]
43. Chowdhury, A.; De, D. RGSO-UAV: Reverse Glowworm Swarm Optimization inspired UAV path-planning in a 3D dynamic environment. *Ad Hoc Netw.* **2023**, *140*, 103068. [CrossRef]
44. Li, B.; Gong, L.; Zhao, C. Unmanned combat aerial vehicles path planning using a novel probability density model based on artificial bee colony algorithm. In Proceedings of the 2013 Fourth International Conference on Intelligent Control and Information Processing (ICICIP), Beijing, China, 9–11 June 2013; pp. 620–625.
45. Al-Betar, M.A. B-hill climbing: An exploratory local search. *Neural Comput. Appl.* **2017**, *28*, 153–168. [CrossRef]
46. Coelho, L.D.S. Gaussian quantum-behaved particle swarm optimization approaches for constrained engineering design problems. *Expert Syst. Appl.* **2010**, *37*, 1676–1683. [CrossRef]
47. Yang, X.S. A New Metaheuristic Bat-Inspired Algorithm. In *Nature Inspired Cooperative Strategies for Optimization (NISCO 2010)*; Cruz, C., Gonzalez, J.R., Pelta, D.A., Terrazas, G., Eds.; Studies in Computational Intelligence; Springer: Berlin/Heidelberg, Germany, 2010; Volume 284, pp. 65–74.
48. Yang, X.S.; Deb, S. Cuckoo Search via L´evy Flights. In Proceedings of the World Congress on Nature & Biologically Inspired Computing (NaBIC 2009), Coimbatore, India, 9–11 December 2009; IEEE Publications: Piscataway, NJ, USA; pp. 210–214.
49. Yang, X.-S. Firefly algorithm, Levy flights and global optimization. In *Research and Development in Intelligent Systems*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 209–218.

50. Wang, G.G.; Zhao, X.; Deb, S. A novel monarch butterfly optimization with greedy strategy and self-adaptive. In Proceedings of the 2015 Second International Conference on Soft Computing and Machine Intelligence (ISCMI), Hong Kong, China, 23–24 November 2015; pp. 45–50.
51. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey wolf optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
52. Lee, K.; Geem, Z. A new meta-heuristic algorithm for continuous engineering optimization: Harmony search theory and practice. *Comput. Methods Appl. Mech. Eng.* **2005**, *194*, 3902–3933. [CrossRef]
53. Mirjalili, S.; Lewis, A. The whale optimization algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
54. Mirjalili, S. The ant lion optimizer. *Adv. Eng. Softw.* **2015**, *83*, 80–98. [CrossRef]
55. Abualigah, L.; Diabat, A.; Mirjalili, S.; Elaziz, M.A.; Gandomi, A.H. The arithmetic optimization algorithm. *Comput. Methods Appl. Mech. Eng.* **2021**, *376*, 113609. [CrossRef]
56. Mirjalili, S. Dragonfly algorithm: A new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. *Neural Comput. Appl.* **2016**, *27*, 1053–1073. [CrossRef]
57. Weiguo, Z.; Wang, L.; Mirjalili, S. Artificial hummingbird algorithm: A new bio-inspired optimizer with its engineering applications. *Comput. Methods Appl. Mech. Eng.* **2022**, *388*, 114194.
58. Azizi, M. Atomic orbital search: A novel metaheuristic algorithm. *Appl. Math. Model* **2021**, *93*, 657–683. [CrossRef]
59. Wang, L.; Cao, Q.; Zhang, Z.; Mirjalili, S.; Zhao, W. Artificial rabbits optimization: A new bio-inspired meta-heuristic algorithm for solving engineering optimization problems. *Eng. Appl. Artif. Intell.* **2022**, *114*, 105082. [CrossRef]
60. Hashim, F.A.; Hussien, A.G. Snake Optimizer: A novel meta-heuristic optimization Algorithm. *Knowl. Based Syst.* **2022**, *242*, 108320. [CrossRef]
61. Zhong, C.; Li, G.; Meng, Z. Beluga whale optimization: A novel nature-inspired metaheuristic algorithm. *Knowl. Based Syst.* **2022**, *251*, 109215. [CrossRef]
62. Shami, T.M.; Grace, D.; Burr, A.; Mitchell, P.D. Single candidate optimizer: A novel optimization algorithm. *Evol. Intel.* **2022**. [CrossRef]
63. Ahmadianfar, O. Bozorg-Haddad, X. Chu, Gradient-based optimizer: A new metaheuristic optimization algorithm. *Inform. Sci.* **2020**, *540*, 131–159. [CrossRef]
64. Agushaka, J.O.; Ezugwu, A.E.; Abualigah, L. Dwarf Mongoose Optimization Algorithm. *Comput. Methods Appl. Mech. Eng.* **2022**, *391*, 114570. [CrossRef]
65. Zhao, S.; Zhang, T.; Ma, S.; Chen, M. Dandelion Optimizer: A nature-inspired metaheuristic algorithm for engineering applications. *Eng. Appl. Artif. Intell.* **2022**, *114*, 105075. [CrossRef]
66. Faramarzi, A.; Heidarinejad, M.; Stephens, B.; Mirjalili, S. Equilibrium optimizer: A novel optimization algorithm. *Knowl. Based Syst.* **2020**, *191*, 105190. [CrossRef]
67. Azizi, M.; Talatahari, S.; Gandomi, A.H. Fire Hawk Optimizer: A novel metaheuristic algorithm. *Artif. Intell. Rev.* **2022**, *56*, 287–363. [CrossRef]
68. Braik, M.; Hammouri, A.; Atwan, J.; Al-Betar, M.A.; Awadallah, M.A. White Shark Optimizer: A novel bio-inspired meta-heuristic algorithm for global optimization problems. *Knowl. Based Syst.* **2022**, *243*, 108457. [CrossRef]
69. Trojovský, P.; Dehghani, M. Pelican Optimization Algorithm: A Novel Nature-Inspired Algorithm for Engineering Applications. *Sensors* **2022**, *22*, 855. [CrossRef] [PubMed]