vehicles

MDPI

Article

Hybrid Cuckoo Search–Tabu Search Metaheuristic with Fuzzy Multi-Objective Optimization for UAV Path Planning in Urban Environments

Ghadah Alshammari *, Abeer Hakeem, Afraa Attiah and Linda Mohaisen

Department of Information Technology, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 21589, Saudi Arabia; ahakim@kau.edu.sa (A.H.); aattiah@kau.edu.sa (A.A.); lmohaisen@kau.edu.sa (L.M.)

* Correspondence: galshammari0005@stu.kau.edu.sa; Tel.: +966-595070733

Abstract

Most UAV missions currently require visiting multiple checkpoints to perform field tasks in environments with varying levels of obstacle complexity. These missions become more challenging because UAVs have limited onboard resources, particularly in terms of energy, making it necessary to determine a safe and efficient path that enables all required visits to be completed while minimizing both travel distance and energy consumption. To address these challenges, this study proposes a hybrid fuzzy metaheuristic approach that integrates Cuckoo Search and Tabu Search for multi-objective UAV path planning. The proposed approach generates collision-free paths in environments with static obstacles and employs fuzzy logic to construct a unified evaluation function, in which distance and energy values are mapped to membership functions and combined into a single fitness score to guide the optimization process. Cuckoo Search drives global exploration of the solution space, while Tabu Search refines solutions locally. Together, they improve path quality and avoid premature convergence. Experimental results across two scenarios with varying obstacle densities and checkpoint counts demonstrate the efficacy of the proposed hybrid approach. Compared with two baseline algorithms, the hybrid approach achieves reductions in path length ranging from 0.01% to 42.11% and in energy consumption ranging from 0.08% to 27.91%, depending on scenario complexity. Moreover, it maintains a high success rate of 96–100% as both checkpoint counts and obstacle density increase, whereas the baseline algorithms drop to 3–13% in more complex environments. These results highlight the effectiveness and scalability of the approach for multi-checkpoint UAV path planning in obstacle-rich environments.

Keywords: UAV path planning; metaheuristics; multi-objective optimization; urban environment; obstacle avoidance.

Academic Editor: Yongmin Zhong

Received: 15 April 2026
Revised: 1 June 2026
Accepted: 4 June 2026
Published: 11 June 2026

1. Introduction

Path planning remains a critical challenge in unmanned aerial vehicle (UAV) operations, particularly in three-dimensional (3D) urban environments characterized by dense obstacles, where onboard energy resources are inherently limited. UAVs are increasingly deployed in real-world applications such as urban delivery, infrastructure inspection and search-and-rescue operations. In all these applications, UAVs are required to depart from a base station, visit multiple spatially distributed checkpoints to perform assigned tasks, and return safely to their origin.

This multi-checkpoint structure introduces a combinatorial ordering problem analogous to the Traveling Salesman Problem (TSP), where determining the optimal visitation sequence is itself NP-hard. Failure caused by energy depletion or collision may lead to significant operational and safety consequences, underscoring the need for effective and efficient path planning strategies.

The complexity of this process increases in obstacle-rich environments, where buildings and other static structures constrain feasible flight paths in three dimensions and may force detours [1]. Under such conditions, UAV missions require not only collision-free navigation but also high operational efficiency, as these vehicles are inherently constrained by finite battery capacity [2]. Consequently, determining a 3D trajectory that minimizes both travel distance and energy consumption while satisfying obstacle-avoidance constraints constitutes a complex multi-objective optimization problem [3].

This challenge becomes more pronounced because energy consumption is not determined solely by travel distance. Hovering, maneuvering, and trajectory deviations caused by obstacle avoidance can introduce additional energy demands, making route evaluation more difficult. Therefore, multi-checkpoint UAV path planning requires optimization strategies capable of balancing distance efficiency, energy conservation, and navigation safety while ensuring safe and efficient mission execution.

Existing approaches to UAV path optimization generally fall into three categories: classical optimization techniques [4], reinforcement learning (RL)-based methods [5], and metaheuristic algorithms [6]. Classical methods can provide precise solutions when the problem has a clear mathematical structure, but they often struggle with large-scale and high-dimensional problems. RL-based approaches offer a more adaptive framework by learning from environmental feedback; however, they commonly suffer from high computational cost, long training time, and large data requirements.

In contrast, metaheuristic algorithms have gained widespread adoption due to their ability to efficiently explore large solution spaces without strong assumptions about problem structure. They employ iterative search mechanisms, often population-based or memory-based, to sample, evaluate, and improve candidate solutions. This makes metaheuristics particularly suitable for UAV path planning, where the search space is often complex, constrained by obstacles, and characterized by multiple conflicting objectives.

Despite their strengths, many existing metaheuristic-based methods for UAV path planning still face several important limitations. First, some approaches emphasize global exploration but do not provide adequate local refinement, which can lead to premature convergence or reduced solution quality. Second, many studies address path length and energy consumption separately or combine them through fixed weighting schemes that may not properly account for differences in scale and relative importance [7]. Third, only a limited number of methods effectively combine exploration–exploitation balance with a structured multi-objective framework specifically designed for obstacle-constrained missions involving multiple checkpoints.

To address these limitations, hybrid metaheuristic approaches offer a suitable solution. By integrating complementary search mechanisms, such approaches can improve global exploration while reinforcing local exploitation. Among metaheuristics, Cuckoo Search (CS) is a widely used swarm intelligence algorithm inspired by the breeding behavior of cuckoos. It uses Lévy flights to explore the search space and replaces weaker solutions with better ones according to their fitness [8]. Although CS has strong global search capability, its performance can be further improved by strengthening its local search behavior. For this purpose, Tabu Search (TS) provides an effective complementary mechanism. TS is a memory-based metaheuristic that employs a tabu list to avoid revisiting previously explored solutions, while allowing selective revisits through an aspiration criterion when

better solutions are found [9]. Therefore, combining CS and TS provides a balanced optimization approach that can enhance both convergence behavior and solution quality.

Since UAV path optimization is inherently multi-objective, an effective objective aggregation mechanism is required [10]. Fuzzy logic offers a systematic approach for handling incommensurable objectives by mapping distance and energy consumption into membership functions and combining them into a unified fitness measure. This helps address conflicts between optimization criteria and uncertainties in the data [11].

Motivated by these considerations, this study proposes Hybrid Cuckoo–Tabu Search (HCTS), a fuzzy metaheuristic approach for multi-objective UAV path planning.

The proposed approach aims to determine an effective visiting order for a set of checkpoints while minimizing path length and energy consumption under obstacle-avoidance constraints.

The main contribution of this study is the integration of Cuckoo Search and Tabu Search to improve the exploration–exploitation balance for the considered UAV path planning problem. To the best of our knowledge, this integration is novel for the problem setting considered in this study.

The main contributions of this study are summarized as follows:

1. The UAV path planning problem is formulated as a constrained fuzzy multi-objective optimization problem, with path length and energy consumption as the objective functions, and obstacle avoidance as the constraint.
2. A hybrid metaheuristic algorithm integrating Cuckoo and Tabu Search is developed to enhance the exploration–exploitation balance and improve convergence behavior in complex search spaces.
3. The proposed HCTS algorithm is evaluated across two representative scenarios with varying checkpoint counts and obstacle densities to assess its performance under different levels of mission complexity.
4. A comprehensive comparative analysis is conducted against the conventional Cuckoo Search and Genetic Algorithms, using multiple evaluation metrics including statistical validation, solution effectiveness, success rate, and computational efficiency.

The rest of the paper is organized as follows. Section 2 covers the review of relevant literature. The problem definition is presented in Section 3. This is followed by a discussion on the proposed HCTS algorithm in Section 4. Results and analysis are presented in Section 5. Finally, the paper concludes in Section 6.

2. Literature Review

The path planning problem, in general, has received notable attention in the literature through various applications. Bahwini et al. [12] studied the path planning in the presence of soft tissue deformation based on bio-heat transfer while using finite element modeling. Zhong et al. [13] proposed a methodology based on neural dynamics for optimal robot path planning by drawing an analogy between cellular neural network (CNN) and path planning of mobile robots. Zhou et al. [14] designed and implemented a crossover recombination-based global-best brain storm optimization algorithm to solve a multi-constraint 3D path planning problem while considering the continuous curvature of path. Hills and Zhong [15] proposed a heat conduction method for optimal robot path planning by drawing an analogy between heat conduction and path planning of mobile robots.

With regard to the path planning for UAVs, the problem has been extensively studied using various algorithmic approaches. This section reviews the relevant literature, categorizing existing methods into classical, heuristic, and learning-based algorithms, and metaheuristic approaches. The review identifies key contributions and limitations that motivate the present study.

2.1. Classical, Heuristic, and Learning-Based Algorithms

Classical algorithms, including A*, Dijkstra, and Fast Marching methods, have been widely applied to UAV path planning due to their completeness and optimality guarantees in discrete search spaces. More recently, deep reinforcement learning methods have emerged as data-driven alternatives that learn planning policies through environment interaction.

Primatesta et al. [16] proposed a risk-aware path planning strategy for UAVs in urban environments, combining an offline A*-based algorithm with an online repair mechanism to minimize risk to the population on the ground. While their work effectively addresses safety considerations in populated areas, it focuses solely on risk minimization as the planning objective and does not consider energy consumption or multi-checkpoint mission requirements. Muñoz et al. [17] addressed the multi-UAV coverage path planning problem by proposing the Fast Marching Square (FM^2) algorithm, developed from the Fast Marching Method. Their approach optimizes path distance as the single decision objective, without considering energy consumption or the sequential visitation of multiple checkpoints required in point-to-point missions.

Sadallah et al. [18] proposed a hybrid FM2 and A approach for real-time safety and travel time optimization. Although achieving sub-3 ms computation, their weighted sum method requires manual tuning and often fails to yield true Pareto-optimal solutions. Tang et al. [19] proposed a risk-based A* algorithm for urban UAV path planning, integrating obstacle risk, death risk, and property loss risk into a unified cost function using weighted sum aggregation with AHP-determined weights. However, the weighted sum approach yields a single compromise solution rather than exploring the full trade-off space.

Rienecker et al. [20] proposed an energy-optimal 3D flight path planning approach for UAVs in urban environments, exploiting local wind phenomena generated by building airflow. Their method integrates Large Eddy Simulation (LES) wind field prediction with a tailored A* algorithm using an energy-based cost function that accounts for headwind, tailwind, and upwind components. Song and Zhou [21] implemented an improved A* algorithm that incorporates search space reduction and node selection enhancements to optimize path length in complex environments. While the improved algorithm demonstrates faster convergence than the basic A* variant, it remains focused on distance minimization without addressing energy consumption or multi-objective optimization requirements.

Tian et al. [22] proposed a fast UAV path planning algorithm called Three-Step Experience Buffer Deep Deterministic Policy Gradient (TSEB-DDPG). A key component of their approach is optimization through Hierarchical Learning Particle Swarm Optimization (HL-PSO) for generating short, collision-free flight paths.

Zhu et al. [23] proposed an improved deep reinforcement learning (DRL) path planning algorithm based on Double Deep Q-Network (DDQN). Their approach optimizes energy consumption as the primary objective while ensuring obstacle avoidance through the reward function design. However, the single-objective formulation does not address the trade-offs between energy and distance, and multi-checkpoint sequential planning is not considered. Sarhan et al. [24] proposed the Deep Deterministic Policy Gradient (DDPG) algorithm for UAV path planning with noise-aware optimization. Two objectives were optimized: path length and noise level, addressing environmental impact considerations.

2.2. Metaheuristic Optimization Approaches

Metaheuristic algorithms, including evolutionary computation and swarm intelligence methods, have gained significant attention for UAV path planning due to their ability to handle complex, high-dimensional search spaces and avoid local optima through stochastic search mechanisms.

Wu et al. [25] proposed a path planning framework for solar-powered UAVs in urban environments, combining a modified Whale Optimization Algorithm (WOA) with a fluid dynamics-based obstacle-avoidance method. The framework optimized path length and energy consumption but was limited to single-destination missions and did not address multi-checkpoint planning. Ren et al. [7] developed CDNSGA-II for multi-objective UAV path planning, optimizing distance and safety using Pareto dominance. However, their method is also restricted to single-destination planning and ignores energy consumption, and the Pareto-based approach requires additional decision-making to select the final path.

Hu et al. [26] proposed a risk-based approach to UAV path planning in urban environments, where operational risk cost is optimized rather than path length. Their framework quantifies three risk categories ground impact on people, ground vehicles, and collision with manned aircraft and compares Dijkstra, A*, and ant colony optimization (ACO) algorithms for generating cost-effective paths.

Majeed and Hwang [27] addressed multi-objective coverage path planning for UAVs in 3D urban environments. Their algorithm optimizes four objectives, number of turning maneuvers, perfect coverage, path overlapping, and path length, using ant colony optimization (ACO) to determine the traversal order of areas of interest formulated as a Traveling Salesman Problem. Wei and Xu [28] proposed a distributed path planning algorithm for multi-UAV communication chains based on dual decomposition, improving the basic ant colony optimization (ACO) algorithm through enhanced path selection strategies, pheromone update rules, and rollback mechanisms. Their approach optimizes path length as the sole objective. Chen et al. [29] proposed a reinforcement learning-based multi-strategy Cuckoo Search (RL-MSCS) algorithm for 3D UAV path planning, employing Q-learning to dynamically select search strategies and improve convergence speed. While the approach addresses the static parameter limitation of conventional CS, it focuses solely on path length optimization without considering multi-checkpoint missions or energy consumption.

Cao et al. [30] proposed an improved ACO algorithm that integrates traditional ACO with the Orthogonal Jump Point Search (OJPS) algorithm for enhanced search efficiency. Although designed as a single-objective optimization algorithm, performance was evaluated across three metrics individually: average risk value of the path, flight time, and number of turns. The lack of unified multi-objective formulation limits the algorithm's ability to simultaneously balance competing objectives.

Ma et al. [31] optimized path length by employing an improved parrot optimization (IPO) algorithm. The improved algorithm incorporates Spatial Pyramid Matching (SPM) chaotic mapping, dynamic adaptive foraging mechanisms, adaptive switching factors, and hybrid Cauchy–Gaussian mutation strategies to enhance global search capability and convergence speed. Lin et al. [32] proposed an enhanced particle swarm optimization algorithm, called Zaslavskii Chaotic Multi-Objective Particle Swarm Optimization (ZAMOPSO). Their approach considers path length, flight altitude variation, and safety margin as optimization objectives. While addressing multiple objectives, the algorithm does not explicitly model energy consumption. Cheng et al. [33] proposed an improved PSO algorithm that modeled UAV path planning as an optimization problem. A cost function was developed that incorporates safety requirements and operational constraints for UAVs.

The analysis reveals that most existing approaches optimize single objectives (distance, energy, or risk) in isolation, with limited support for multi-checkpoint missions common in inspection and delivery applications. Multi-objective methods typically rely on weighted sum aggregation requiring manual parameter tuning, while energy consumption critical for UAV operational feasibility is frequently overlooked. To address these gaps, this study proposes a HCTS algorithm with fuzzy logic-based fitness unification, enabling simulta-

neous distance and energy optimization for multi-checkpoint missions while ensuring collision-free navigation through static obstacles in urban environments.

3. Problem Definition

The path planning problem for a UAV in an urban environment is focused on static obstacle avoidance and path planning, searching a path for a UAV to minimize the total sum of distances during their assigned mission as well as UAV energy consumption, keeping into account safety constraints throughout the mission. The main purpose of path planning is to facilitate the flight of a UAV from location i to location j [6]. Mathematically, it can be defined as follows [6]:

$$i \xrightarrow{r} j$$

where $\xrightarrow{r}$ is a flight path connecting i and j. In practice, there are various constraints depending on mission requirements. Hence, the constrained path planning can be written as follows:

$$i \xrightarrow{][} j$$

where $\xrightarrow{][}$ represents the constraints. Consequently, the UAV path planning problem in an urban environment has several elements, including terrain modeling, calculations of path lengths and energy consumption, and definition of safety constraints. These aspects are discussed below.

3.1. Terrain Modeling

The urban type terrain model is adopted from Pehlivanoglu et al. [6]. The model is generated by using rectangular points with different locations and heights. These points simulate city buildings, and their locations along with heights are randomly generated as follows:

$$\begin{bmatrix} x_{1,i} & x_{2,i} \end{bmatrix}^T = \begin{bmatrix} \text{rand}[x_1^L, x_1^U] \\ \text{rand}[x_2^L, x_2^U] \end{bmatrix}$$

$$B_i^c \mid^{i=1,2,\ldots,n} = (x_{1,i}, x_{2,i})$$

$$B_i^h = \text{rand}[0, h]$$

In the above equations, rand is a random number operator; x_{L1} and x_{U1} represent the boundary values of the plane area, B_{ci} is the center of the ith point, n is the total number of check points on the plane area, B_{hi} is the height of the ith point, and h is the height limitation for points. n is taken as 5, 10, 15, and 20 in this study. Further details can be found in Pehlivanoglu et al. [6].

3.2. Calculation of Path Length

The distance (path length) of the flight should be minimized and should also not exceed the maximum distance due to a restriction on energy usage. Mathematically,

$$\text{Minimize } L_i \quad (\text{such that } L_i \leq L_{\max})$$

where L_i is the path length of the mission and $L_{\max}$ is the maximum possible distance from the mission.

For path length, we calculate the straight-line distance between each pair of points along the route. The goal is to sum these distances to determine the total length of the path.

The distance between two checkpoints is calculated in terms of the Euclidean distance using the Pythagorean theorem as follows [34]:

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{1}$$

The following equation calculates the total distance (the sum of all linear distances) between all checkpoints, including the distance covered due to deviations caused by obstacles. In the case of obstacle avoidance, the path is divided into smaller straight segments, resulting in the total distance traveled by the UAV.

$$L_{ij} = \sqrt{\sum_{k=1}^{n} \left(x_{ik} - x_{jk}\right)^2 + \left(y_{ik} - y_{jk}\right)^2} \tag{2}$$

where n is total number of checkpoints.

3.3. Calculation of Energy Consumption

The energy used throughout the flight should not exceed the total available energy $E_{\max}$, because the UAV's battery capacity is limited. The total energy consumed during the mission must remain below the battery capacity [35].

$$\text{Minimize } E_i \quad (\text{such that } E_i \leq E_{\max})$$

where E_i is the energy used during the mission and $E_{\max}$ is the total available energy. The difficulty lies in ensuring that the UAV can locate targets, navigate checkpoints, and return to base, while maintaining reliable performance.

The energy consumption of a UAV is mainly associated with two tasks: overcoming the gravity for staying airborne, and countering drag forces due to wind and forward motion [36]. Following the model proposed by Stolaroff et al. [37], the minimum power with forward motion for a UAV is calculated as

$$p_{\min} = (\hat{v} + v \sin \beta) T \tag{3}$$

where $\hat{v}$ is the induced velocity required for a given thrust T, v is the average ground speed of the UAV, and β denotes the pitch angle.

For a UAV with mass m, including the mass of the UAV body and battery, the required thrust T is the following [37]:

$$T = mg + f_d \tag{4}$$

where g is the gravitational constant and f_d represents the drag force. This drag force is proportional to the speed of the air, the density of air ρ and the drag coefficient.

The induced velocity is then calculated as [37]

$$\hat{v} = \frac{2T}{qr^2 \pi \rho \sqrt{(v \cos \beta)^2 + (v \sin \beta + \hat{v})^2}} \tag{5}$$

In Equation (5), r represents the diameter of the UAV rotors, while q indicates the number of UAV rotors.

The total flying energy that an arbitrary UAV k consumes to traverse distance d is

$$E_f(k) = \frac{p_f^{\min} d}{v \eta k} \tag{6}$$

where η_k is the power efficiency of the UAV k and is found empirically.

Furthermore, the actual power consumption for hovering of UAV k is [37]

$$p_h(k) = \frac{p_h^{\min}}{\eta_k} = \frac{T\sqrt{T}}{\eta_k\sqrt{0.5\pi q r^2 \rho}} \tag{7}$$

where $p_h^{\min}$ is the theoretical minimum power for hovering. Using this information, the hovering energy per node is calculated as

$$E_h(k) = p_h(k) \times t_h(k) \tag{8}$$

where $E_h(k)$ is the hovering energy per stop and $t_h(k)$ is the hovering time per stop. The total hovering energy, $E_{h\_total}(k)$, is therefore calculated as follows:

$$E_{h\_total}(k) = E_h(k) \times (n+1) \tag{9}$$

In the equation above, the term $(n+1)$ accounts for hovering events occurring at all mission stops, including the initial deployment position and the final return location, in addition to the n intermediate checkpoints.

The total energy consumption would be the sum of energy required for flying and hovering, as given below:

$$E(k) = E_f(k) + E_{h\_total}(k) \tag{10}$$

3.4. Safety Constraints

A UAV is focused on searching a path to minimize the total sum of distances and energy consumption, while satisfying UAV safety constraints. As suggested by Majeed and Hwang [27], the UAV must maintain a safe distance d_{safe} from all obstacles O_i.

These obstacles include both structured urban buildings and obstacles that are semi-randomly generated within predefined boundaries. However, the distribution is not entirely random, as a fixed random seed is used to ensure reproducibility and maintain consistent testing scenarios across all experiments.

The environment is represented as a 3D grid, where obstacles are defined by both their horizontal footprint (width and depth along the x- and y-axes) and their height (along the z-axis). This representation allows precise verification of the distance between the UAV and obstacles in all dimensions, ensuring that the UAV does not collide with any obstacle during flight.

$$d(p_d, O_i) \geq d_{\text{safe}} \quad s.t. \quad d_{\text{safe}} > d_{UAV}$$

where d_{UAV} is the maximum diameter of the UAV. Furthermore, p_d represents the UAV trajectory segment between two consecutive checkpoints, and the minimum distance between the segment and each obstacle is evaluated to ensure collision-free motion.

3.5. Fuzzy Logic-Based Multi-Objective Optimization Approach

In a multi-objective optimization problem, conflicting objectives are optimized simultaneously to find the best tradeoff between them. A key condition for defining a problem as multi-objective is addressing the incommensurability between objectives, as they often have different units and magnitudes. One approach to resolving this is by bringing the objectives onto the same scale, often through aggregation techniques [38], with fuzzy logic being a widely used method for this purpose [39].

For the UAV path planning problem, two objectives—path length and energy consumption—must be simultaneously optimized. These objectives can conflict in certain scenarios: a UAV may cover a shorter distance but consume more energy due to extensive hovering, caused by terrain structure or obstacle distribution. Furthermore, path length is

measured in meters (or another unit), while energy consumption is measured in Watts. To address the incommensurability of these objectives, fuzzy logic is employed for aggregation, as explained below.

The first step is form a decision function which is a mathematical equation. This equation represents an optimization rule (also known as decision rule). For the UAV path planning problem, the following decision rule may be constructed.

Rule: IF Path Length is decreased AND Energy Consumption is decreased THEN the Path is efficient

In the above rule, *Path Length*, and *Energy Consumption* are the linguistic variables. To implement the above rule using fuzzy logic, membership functions for the linguistic variables have to be defined. A membership function returns value in the interval [0, 1]. A value near 1 signifies higher level of satisfaction whereas a value near 0 indicates lower level of satisfaction with regard to the decision criterion under consideration [40]. Using the formula by Tawfeek et al. [41], the rule translates to the following equation:

$$\mu_R(x) = w_1\mu_L(x) + w_2\mu_E(x) \tag{11}$$

In the above equation, $\mu_R(x)$ defines the membership function for the route. Similarly, $\mu_L(x)$ and $\mu_E(x)$ represent the membership functions for path length and energy consumption, respectively. Furthermore, w_1 and w_2 are the weights associated with $\mu_L(x)$ and $\mu_E(x)$, respectively.

Note that the two weights should be between 0 and 1, and that $w_1 + w_2 = 1$. In the present study, $w_1 = w_2 = 0.5$ to give balanced importance to both objectives and avoid introducing a prior preference toward either path length or energy consumption. However, different weight combinations can be adopted in mission-specific applications when one objective is prioritized over the other. In order to calculate $\mu_R(x)$, both $\mu_L(x)$ and $\mu_E(x)$ must first be calculated, as explained below.

3.5.1. Membership Function for Path Length

The membership function for path length, $\mu_L(x)$, can be defined as follows. The terms L_{max} and L_{min} define the maximum (ideal) and minimum values, respectively, for the path length. The membership function for path length is mathematically represented as follows.

$$\mu_L(x) = \begin{cases} 1 & \text{if } L(x) \leq L_{min} \\ \frac{L_{max} - L(x)}{L_{max} - L_{min}} & \text{if } L_{min} < L(x) \leq L_{max} \\ 0 & \text{if } L(x) > L_{max} \end{cases} \tag{12}$$

where the term $L(x)$ represents the path length of the current solution x.

3.5.2. Membership Function for Energy Consumption

The membership function for energy consumption is termed as $\mu_E(x)$. To define $\mu_E(x)$, the maximum and minimum (ideal) values 'E_{max}' and 'E_{min}' need to defined respectively. The membership value for energy consumption time is found as follows.

$$\mu_E(x) = \begin{cases} 1 & \text{if } E(x) \leq E_{min} \\ \frac{E_{max} - E(x)}{E_{max} - E_{min}} & \text{if } E_{min} < E(x) \leq E_{max} \\ 0 & \text{if } E(x) > E_{max} \end{cases} \tag{13}$$

where the term $E(x)$ represents the energy consumption of the current solution x.

3.6. Calculations of Upper and Lower Limits of Objectives

The calculation of the membership values from the two membership functions discussed above requires that the upper and lower limits of the objectives are determined first. That is, the values of L_{max}, L_{min},E_{max}, and E_{min} need to be found beforehand. These values are calculated as follows:

The minimum distance, L_{min}, is taken as 0. The maximum distance, L_{max} is modeled according to a worst-case scenario that represents the longest feasible trajectory inside the operational environment. It is defined as follows:

$$L_{max} = (n+1)D_{env} \tag{14}$$

where D_{env} is the maximum Euclidean distance across the environment, and is mathematically represented as

$$D_{env} = \sqrt{(x_{range}^2 + y_{range}^2 + h_{max}^2)} \tag{15}$$

In Equation (15), x_{range}, y_{range}, and h_{max} represent the spatial limits of the mission area.

Similarly, the minimum energy consumption, E_{min}, is taken as 0. The maximum energy consumption, E_{max}, corresponds to the worst-case scenario and incorporates both forward-flight and hovering energy components. It is expressed as

$$E_{\max} = P_f^{\text{nom}} \frac{D_{\max}}{v_{\text{nom}}} + (n+1)E_{\text{hover}} \tag{16}$$

where P_f^{nom} represents the nominal forward-flight power, v_{nom} is the nominal UAV velocity, and E_{hover} denotes the hovering energy consumed at each checkpoint.

4. Proposed Path Planning Algorithm

This section first presents the basic Cuckoo Search (CS) algorithm for the sake of completeness. Then, the proposed Hybrid Cuckoo–Tabu Search (HCTS) algorithm is presented, whereby each step of the HCTS algorithm is explained.

4.1. Basic Cuckoo Search Algorithm

The CS algorithm is a swarm intelligence-based metaheuristic inspired by the brooding parasitism of cuckoo species. Proposed by Yang and Deb [8], the algorithm is used for optimization problems both in continuous and discrete domains. In brood parasitic behavior, some cuckoo species lay their eggs in the nests of other host birds (of other species). Some host birds can engage in direct conflict with the intruding cuckoos. Inspired by this, the original CS algorithm evolves from the following three behavioral patterns of real cuckoos [8]:

- Each cuckoo lays one egg at a time. The egg is dumped in a nest randomly chosen by the cuckoo.
- The best nests with high quality of eggs (solutions) will carry over to the next generations.
- The number of available host nests is fixed, and a host can discover an alien egg with probability $p_a \in (0,1)$. In this situation, the host bird can either throw the egg out of its nest or abandon the nest in order to build a completely new nest in a new location.

A nest represents a potential solution in the search space. Each solution is updated via the concept of *Lévy* flight. A *Lévy* flight is a combination of short and long steps (represented by α), with sudden turns. These steps contribute mainly towards exploitation

phase during the search process, and determine the next position of the cuckoo (i.e., the updated solution). The position is calculated using the following equation:

$$x_i(t+1) = x_i(t) + \alpha * Lévy(\lambda) \tag{17}$$

where $\alpha > 0$ represents a step size. This step size should be closely related to the scale of the test function that the algorithm is applied on. In most cases, α can be set to the value of 1 [8]. Furthermore, λ controls the scale of *Lévy* flight and $\lambda \in (0,3]$.

In the context of the present study, the position update expression given in (17) is adapted to a discrete Traveling Salesman Problem (TSP) formulation. Since route solutions are represented as permutations rather than continuous vectors, the additive update is replaced by Lévy-driven swap operations.

In the proposed implementation, the *Lévy* distribution parameter is set to λ = 1.5. Furthermore, α is implicitly determined by the problem size through the permutation length N, is proportional to the $|\,Lévy\,(\lambda)\,| \times N$ number of swap operations.

4.2. Hybrid Cuckoo–Tabu Search (HCTS) Algorithm

A major challenge in metaheuristic optimization is achieving an effective balance between exploration and exploitation. One approach to achieve this is through *hybridization* of metaheuristics [42]. As such, the integration of CS and TS allows the broad exploration capability of CS to be complemented by the focused exploitation capability of TS, resulting in a hybrid search strategy for UAV path planning in complex urban environments.

In the proposed checkpoint-ordering problem, CS enhances exploration by generating diverse checkpoint-visiting permutations using *Lévy*-flight-based updates, which helps the algorithm search different regions of the solution space and reduces the risk of premature convergence.

In contrast, TS enhances exploitation by refining the current best visiting sequence through neighborhood-based moves, while the tabu list prevents cycling back to recently visited sequences. Thus, HCTS maintains global diversification through CS while improving local intensification through TS. The complete procedure of the HCTS algorithm is summarized in Algorithm 1.

Given a 3D urban environment defined by an obstacle set O and a checkpoint set C, the algorithm initializes a population of nests, each nest i represents a permutation of checkpoints. For each nest, the total path length L_i, energy consumption E_i, and fuzzy fitness F_i are evaluated, with collision detection employed to reject any infeasible path that intersects an obstacle. During each iteration $t \leq MaxIter_{CS}$, a new candidate solution $\hat{S}_i$ is generated via Lévy-flight-based permutation. The corresponding path length $\hat{L}_i$, energy consumption $\hat{E}_i$, and fuzzy fitness $\hat{F}_i$ are then computed.

A randomly selected nest j is then considered for comparison. If its fitness exceeds that of the newly generated candidate solution, nest i is replaced accordingly. To maintain population diversity, a fraction p_a of the worst-performing nests is abandoned and regenerated stochastically. Once the global best solution S is updated, a Tabu Search procedure is applied in the neighborhood of S for local intensification. If an improved solution is identified, S is updated accordingly.

Algorithm 1 HCTS-based urban environment UAV path planning.

Input: Urban environment parameters, obstacle set O, checkpoint set C, UAV specifications, energy constants, $MaxIter_{CS}$, HCTS parameters
Output: Total path length L, total energy consumption E, and runtime
Initialize the 3D urban environment, and define the checkpoint set C and obstacle set O
Initialize Cuckoo Search population (nests i as checkpoint permutations).
for each nest i **do**
    Compute the total path length, total energy consumption, and fuzzy fitness of nest i as (L_i, E_i, F_i)
    Check for collision: If path intersects any obstacle, reject nest i
end for
Determine the initial best solution S with fitness F
for $t = 1$ to $MaxIter_{CS}$ **do**
    for each nest i **do**
        Generate a new candidate solution $\hat{S}_i$ using Lévy-flight-based permutation
        Compute the total path length, total energy consumption, and fuzzy fitness of $\hat{S}_i$ as $(\hat{L}_i, \hat{E}_i, \hat{F}_i)$
        Check for collision: If $\hat{S}_i$ intersects any obstacle, reject this candidate solution
        Randomly select a nest j
        if $\hat{F}_i > \hat{F}_j$ **then**
            Replace nest j with i
        end if
    end for
    Abandon a fraction p_a of the worst nests and regenerate them randomly
    for each regenerated nest k **do**
        Compute the total path length, total energy consumption, and fuzzy fitness of nest k as (L_k, E_k, F_k)
        Check for collision: If k intersects any obstacle, reject this regenerated nest
    end for
    Update the current global best solution S
    Apply Tabu Search around S for local intensification
    Update S if an improved local solution is found
end for

5. Experimental Results and Analysis

This section discusses the experimental setup, performance measures, and empirical analysis of the proposed Hybrid Cuckoo–Tabu Search (HCTS) approach.

5.1. Experimental Setup

The simulation environment is developed using MATLAB, where a 100 × 100 m map is designed in both length and width, with random heights for the obstacles, as described in Section 3.1. The checkpoints and obstacles are defined within this environment to simulate UAV paths in an urban setting. Figure 1 illustrates an example of a simulation scenario, representing an urban environment containing five checkpoints and five obstacles.

Table 1 presents the various parametric values used in the experiment, including technical information related to the UAV, drag force, average ground speed, and the algorithmic parameter values.

To evaluate the performance of the HCTS algorithm, two experimental scenarios are considered. In the first scenario, both the number of checkpoints and the number of obstacles are gradually increased to examine how the algorithm adapts to progressively higher levels of complexity. In the second scenario, the number of checkpoints remains constant while the number of obstacles is incrementally increased, allowing for a focused assessment of how obstacle count alone affects algorithmic performance. In both scenarios,

HCTS is compared against two baseline algorithms, Cuckoo Search (CS) and Genetic Algorithm (GA).

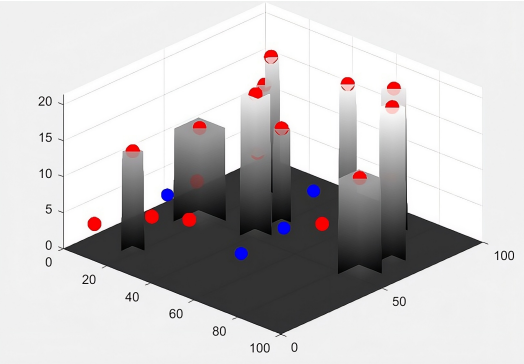

Figure 1. Urban environment with checkpoints (blue) and obstacles (red).

Table 1. Parametric values used in experimentation

Symbol	Parameter	Value
MU	mass of UAV	1.07 Kg [37]
MB	mass of battery	1 Kg [37]
ρ	density of air	1.225 Kg/m^3 [37]
f_d	drag force	9.6998 N [37]
q	number of rotors	4 [37]
r	rotor diameter	0.254 m [37]
η	power efficiency	70% [37]
v	average ground speed	1.49 m/s [37]
TLS	tabu list size	3, 5, and 7
λ	Lévy distribution	1.5 and 2
p_a	probability of abundance	0.25 and 0.5
pop	population size	10, 15, 20, and 25
CR	crossover probability	0.8 [43]
MR	mutation rate	0.1 [43]

GA is selected as a benchmark because it is a well-established, nature-inspired algorithm with proven effectiveness in solving a wide range of optimization problems. GA operates by evolving a population of candidate solutions over successive generations through the application of genetic operators, namely selection, crossover, and mutation. In each generation, individuals are selected based on their fitness, combined to produce offspring via crossover, and subjected to random mutations to maintain diversity and explore the search space. Following the parameter settings reported in [43–45], the GA parameters were set as follows: population size = 15, crossover probability = 0.8, and mutation rate = 0.1, with tournament selection employed as the selection strategy. The CS parameters were set as follows: population size = 15, p_a = 0.5, and λ = 1.5.

5.2. Performance Metrics

The performance of the HCTS algorithm was assessed using three measures, namely, statistical validation, effectiveness, and success rate. These metrics were selected to provide a comprehensive evaluation of the algorithm's efficiency. The following sub-sections describe each metric in detail and explain how they are computed and interpreted in the context of UAV path planning.

5.2.1. Statistical Validation

The measure was adopted following the established guidelines proposed by Derrac et al. [46]. Accordingly, all results were subjected to statistical validation using the Wilcoxon rank-sum test, which is a non-parametric statistical test. A confidence level of 95% was used for statistical testing, corresponding to a significance level of $\alpha = 0.05$. Furthermore, 30 independent runs were performed for each experimental setting, and the results were reported as the mean of 30 runs along with the standard deviation. All simulations were run for 100 iterations. The Wilcoxon rank-sum test was applied to compare the performance distributions obtained from the 30 independent runs. Specifically, the proposed HCTS algorithm was compared with the baseline algorithms, CS and GA, in terms of path length and energy consumption under each experimental scenario. A p-value lower than 0.05 indicates that the difference between the compared algorithms is statistically significant.

5.2.2. Effectiveness

This measure (also known as average percentage error) was adopted from Campuzano et al. [47]. The effectiveness measure provides information on how far the average solution (average of 30 runs) is from the optimum solution, and is calculated as follows:

$$Effectiveness = \frac{p_{average} - p_{opt}}{p_{opt}} \times 100 \tag{18}$$

where $p_{average}$ is the average fitness of solutions, and p_{opt} is the optimum fitness. As could be interpreted from the above equation, the closer the average fitness is to the optimum fitness, the smaller is the value of the effectiveness measure. As such, a small numerical value of the measure is desirable.

5.2.3. Success Rate

The measure was adopted from [48], where the quality index represents the rate of successful or "good" solutions obtained according to the requirements of the optimization problem. A good solution may be defined as a solution whose cost does not exceed a certain threshold over the minimum cost [48].

Similarly, Liang et al. [49] suggest a success condition with $f(x) - f(x_{opt}) \leq 0.0001$, where $f(x)$ is the objective value of the obtained solution, and $f(x_{opt})$ is the objective value of the optimum or reference best solution. However, the threshold value can be determined by the problem solver based on the nature of the optimization problem. Mathematically, the success rate is defined as follows.

$$\eta_{quality} = \frac{N_{successful}}{N_{sim}} \tag{19}$$

where $N_{successful}$ is the number of "good" solutions depending on the optimization goal, and N_{sim} is the total number of simulations performed (30 in the present study). In this work, a run is considered successful if the obtained path length or energy value is within 1% of the best value achieved by the same algorithm under the same scenario.

5.3. Empirical Analysis

The performance of the HCTS algorithm is evaluated through two distinct sets of experiments. The first set focuses on a parameter sensitivity analysis, in which the algorithm's behavior is examined under different configurations of tabu list size (TLS), the Lévy distribution parameter (λ), and the probability of abundance (p_a). Additionally, the influence of population size on solution quality is investigated.

The second set of experiments involves a comparative analysis with the standard Cuckoo Search (CS) and the Genetic Algorithm (GA). This comparative analysis is conducted under two scenarios: in the first scenario, both the number of checkpoints and obstacles increase simultaneously, whereas in the second scenario, the number of checkpoints remains fixed while the number of obstacles increases.

5.3.1. Parameter Sensitivity Analysis

A comprehensive parameter sensitivity analysis was conducted to determine the optimal configuration for the HCTS algorithm. As detailed in Table 2, twelve unique combinations of tabu list size (TLS), probability of abundance (p_a), and Lévy distribution (λ) were evaluated for 10 checkpoints and 10 obstacles. This scenario provides a balanced environment where the number of checkpoints and obstacles is both sufficiently large to challenge the algorithm, yet not excessively large to introduce overly complicated conditions. The results indicate that Case 11 (TLS = 7, $p_a = 0.5, \lambda = 1.5$) achieved the most favorable performance, with a minimum path length of 302.10 units and energy consumption of 113,261.78 units.

Notably, configurations using $\lambda = 2$ exhibited a standard deviation of zero, indicating strong algorithmic stability and consistent convergence. Furthermore, the impact of population size was analyzed using four values: 10, 15, 20, and 25. The population size refers to the number of candidate solutions considered by the algorithm in each iteration.

As shown in Figure 2, a population size of 15 achieved the lowest path length and energy consumption among the tested values. Increasing the population size beyond this value did not provide further improvement and resulted in higher path length and energy consumption. Therefore, a population size of 15 was selected as the final setting for the subsequent experiments.

To ensure the reliability of the parameter sensitivity analysis, the Wilcoxon rank-sum test was applied across all parameter and population size experiments. In this analysis, Case 11 was selected as the reference configuration because it achieved the best overall performance among the tested parameter settings. The remaining configurations were then statistically compared against this reference configuration.

The results showed statistically significant differences for most cases in both path length and energy consumption at a significance level of 0.05, except for Case 9, where no significant difference was observed. Overall, the statistical validation confirmed that Case 11 and a population size of 15 provide robust parameter settings for the subsequent comparative experiments.

Table 2. Sensitivity analysis of HCTS parameters. The best results are marked in boldface.

Case	TLS	p_a	λ	Path length	Energy	Average Runtime (s)
Case 1	3	0.25	1.5	324.03 ± 3.45	118,347.48 ± 848.70	0.61
Case 2	3	0.25	2	309.22 ± 0.00	115,061.00 ± 0.00	0.56
Case 3	3	0.5	1.5	332.11 ± 0.68	118,090.81 ± 197.69	0.65
Case 4	3	0.5	2	302.42 ± 0.00	113,436.26 ± 0.00	0.59
Case 5	5	0.25	1.5	303.02 ± 0.63	113,631.30 ± 225.42	0.60
Case 6	5	0.25	2	309.22 ± 0.00	115,061.00 ± 0.00	0.56
Case 7	5	0.5	1.5	320.78 ± 0.65	113,551.62 ± 214.39	0.64
Case 8	5	0.5	2	302.42 ± 0.00	113,436.26 ± 0.00	0.58
Case 9	7	0.25	1.5	302.82 ± 2.58	113,458.85 ± 608.56	0.58
Case 10	7	0.25	2	309.22 ± 0.00	115,061.00 ± 0.00	0.55
Case 11	7	0.5	1.5	**302.10 ± 0.32**	**113,261.78 ± 111.85**	0.64
Case 12	7	0.5	2	302.42 ± 0.00	113,436.26 ± 0.00	0.57

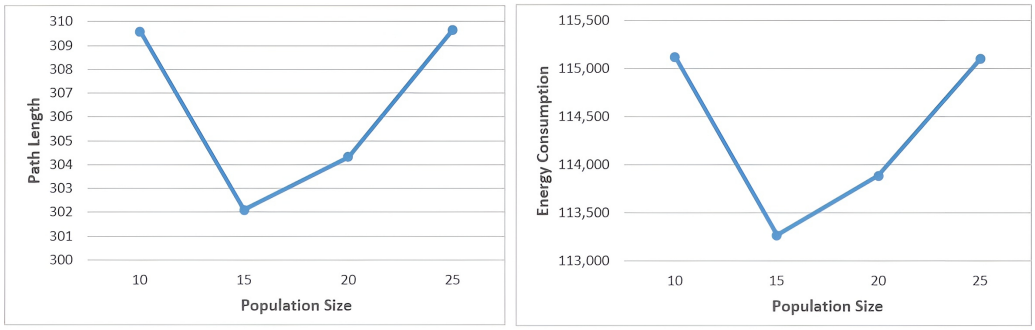

Figure 2. Effect of population size on HCTS performance: (**a**) path length and (**b**) energy consumption.

5.3.2. Comparative Analysis of HCTS, CS, and GA

This section presents a comparative analysis of the proposed HCTS algorithm against two baseline algorithms, Cuckoo Search (CS) and Genetic Algorithm (GA), across two experimental scenarios. The comparison is conducted in terms of path length, energy consumption, and runtime, with additional assessment of statistical significance, effectiveness, and success rate to provide a comprehensive evaluation of algorithmic performance.

5.3.3. Comparative Analysis of Scenario I: Increasing Obstacles and Checkpoints

This section provides a comprehensive evaluation of the HCTS algorithm's performance, across varying levels of environmental complexity compared to the CS and GA baselines. Table 3 presents a summary of the results for HCTS, CS, and GA. A comparison of the three algorithms across four levels of complexity, where both the number of obstacles and checkpoints increase progressively. Specifically, the levels are defined as follows: Level 1 with 5 checkpoints and 5 obstacles, Level 2 with 10 checkpoints and 10 obstacles, Level 3 with 15 checkpoints and 15 obstacles, and Level 4 with 20 checkpoints and 20 obstacles. The comparison considers two optimization objectives: path length and energy consumption.

As seen from Table 3, HCTS performed better than CS and GA in terms of path length across all levels. For the first levels (5 checkpoints and 5 obstacles), HCTS and CS almost produced the same average path length (279.46 units and 279.49 units, respectively). However, in the other three test levels, HCTS showed better performance than CS. Furthermore, GA demonstrated the worst performance among the three algorithms for all test levels in terms of path length. Regarding the energy consumption objective, once again, the HCTS algorithm demonstrated the best performance among the three algorithms for all test levels.

Table 3. Comparison results of Scenario I: increasing obstacle and checkpoint counts

Level	Algo	Path Length	Energy	Runtime (s)
5 CP, 5 Obs	HCTS	279.46 ± 0.77	88,811.49 ± 259.82	0.27
	CS	279.49 ± 0.64	88,884.04 ± 394.07	0.22
	GA	281.53 ± 6.89	89,301.42 ± 1539.97	0.33
10 CP, 10 Obs	HCTS	302.37 ± 0.70	113,363.32 ± 230.28	0.75
	CS	367.39 ± 18.39	127,886.36 ± 4114.00	0.40
	GA	360.12 ± 27.29	126,223.09 ± 6088.19	0.64
15 CP, 15 Obs	HCTS	318.67 ± 0.78	136,036.33 ± 222.47	1.85
	CS	364.49 ± 16.06	127,227.80 ± 3589.51	0.40
	GA	468.50 ± 47.32	169,637.38 ± 10,569.67	1.07
20 CP, 20 Obs	HCTS	426.28 ± 2.12	179,271.71 ± 631.61	3.75
	CS	736.37 ± 39.32	248,680.95 ± 8847.84	1.06
	GA	663.42 ± 60.05	232,477.52 ± 13,380.62	1.60

However, the HCTS algorithm showed higher execution times for all test levels, while CS showed the lowest execution times. The higher execution time observed in HCTS can be attributed to the additional computational overhead introduced by the Tabu Search mechanism. Specifically, the processes of maintaining the tabu list and managing memory structures to avoid local optima require more processing cycles per iteration compared to the standard CS and GA.

In addition to the numerical values, the performance of the three algorithms was also evaluated using the three performance metrics, as mentioned in Section 5.2. Table 4 provides the statistical validation results used to determine whether the performance differences between HCTS and the baseline algorithms were statistically significant. As shown in the table, almost all *p*-values are lower than the significance level of 0.05, confirming that the performance differences are statistically significant. Overall, the *p*-values reported in Table 4 indicate that HCTS produced statistically significant improvements over CS and GA for both optimization objectives. The only exception is at level 1, with 5 checkpoints and 5 obstacles, where the results of HCTS were of comparable quality to those of CS. Specifically, the *p*-values for path length and energy were 0.4513 and 0.2078, respectively, both exceeding the threshold of 0.05. This indicates comparable performance in low-complexity environments, as visually confirmed by the overlapping distributions in Figure 3. However, as the search space expanded, the statistical superiority of HCTS became evident. This difference in performance, along with the clear advantage of HCTS in highly complex environments, is further demonstrated by the statistical distributions shown in Figure 4, which represents the level with 20 obstacles and 20 checkpoints.

Table 4. Statistical validation of Scenario I: increasing obstacle and checkpoint counts

Level	HCSTS vs. CS		HCSTS vs. GA	
	p-Value (Length)	*p*-Value (Energy)	*p*-Value (Length)	*p*-Value (Energy)
5 CP, 5 Obs	0.4513	0.2078	0.0271	0.0167
10 CP, 10 Obs	1.69×10^{-17}	1.69×10^{-17}	1.69×10^{-17}	1.69×10^{-17}
15 CP, 15 Obs	1.69×10^{-17}	1.69×10^{-17}	1.69×10^{-17}	1.69×10^{-17}
20 CP, 20 Obs	1.69×10^{-17}	1.69×10^{-17}	1.69×10^{-17}	1.69×10^{-17}

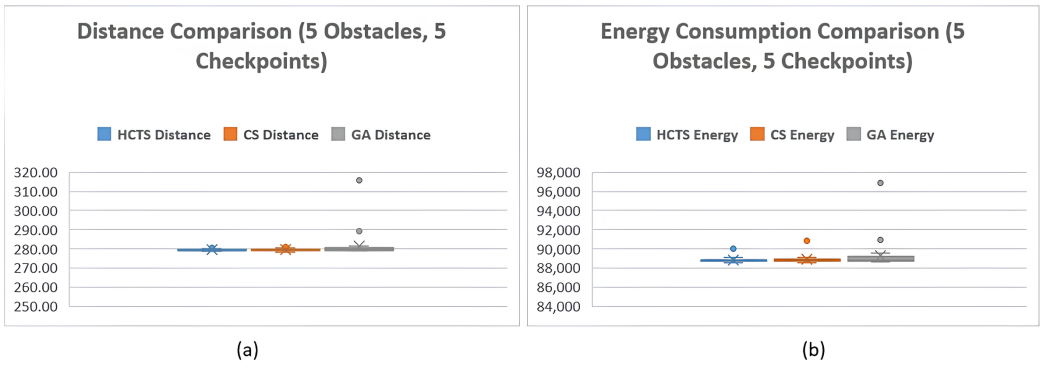

Figure 3. Statistical distribution and performance comparison of HCTS, CS, and GA under 5 obstacles and 5 checkpoints over 30 independent runs: (**a**) path length and (**b**) energy consumption.

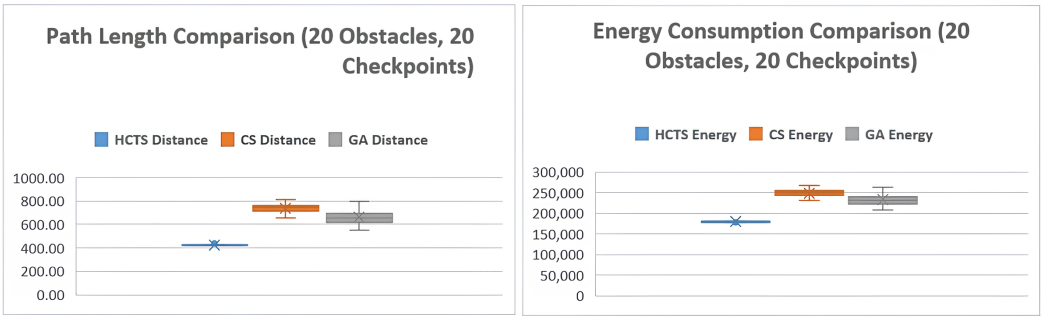

Figure 4. Statistical distribution and performance comparison of HCTS, CS, and GA under 20 obstacles and 20 checkpoints over 30 independent runs: (**a**) path length and (**b**) energy consumption.

With regard to the effectiveness measure, the results in Table 5 clearly indicate the superior performance of the HCTS algorithm, both in terms of the path length and energy consumption. Recall from Section 5.2 that a small value for the effectiveness measure is highly desirable. As such, the effectiveness values obtained by HCTS are below 0.6, whereas for CS and GA, these values are quite high.

Table 5. Effectiveness of Scenario I: increasing obstacle and checkpoint counts

Level	Effectiveness					
	HCTS		CS		GA	
	Length	Energy	Length	Energy	Length	Energy
5 CP, 5 Obs	0.31	0.29	0.40	0.40	0.94	0.77
10 CP, 10 Obs	0.24	0.23	16.55	9.89	16.64	9.91
15 CP, 15 Obs	0.38	0.26	8.69	5.64	22.01	12.37
20 CP, 20 Obs	0.53	0.43	12.27	7.85	19.61	11.60

For the success rate metric, the results in Table 6 also clearly indicate the best performance of HCTS among all the three algorithms. For all four levels, HCTS almost reached the best solution. That is, for the 30 independent runs executed, the HCTS algorithm achieved success between 29 and 30 runs for the path length and energy consumption objectives. In contrast, CS and GA showed an inferior performance for almost all test levels. The only exception is the level of 5 checkpoints and 5 obstacles where both HCTS and CS had the same success rate. Furthermore, among the three algorithms, GA had the worst performance.

Table 6. Success rate of Scenario I: increasing obstacle and checkpoint counts

Level	Success Rate					
	HCTS		CS		GA	
	Length	Energy	Length	Energy	Length	Energy
5 CP, 5 Obs	96%	96%	96%	96%	86%	86%
10 CP, 10 Obs	100%	100%	3%	3%	3%	3%
15 CP, 15 Obs	96%	100%	6%	6%	3%	3%
20 CP, 20 Obs	96%	96%	13%	13%	3%	3%

5.3.4. Comparative Analysis of Scenario II: Increasing Obstacles with Fixed Checkpoint Counts

In this section, the performance of the algorithms is evaluated under an environment where one factor remains fixed and another varies. Specifically, the number of checkpoints

is kept constant, while the number of obstacles is gradually increased from low to high levels (5, 10, 15, and 20). This design isolates the effect of environmental complexity, ensuring that any changes in path length, energy consumption, or runtime can be attributed primarily to obstacle density rather than to changes in mission size. The 20-checkpoint setting is selected to represent a relatively complex mission, allowing the scalability of the algorithms to be examined under demanding routing conditions across different obstacle levels. In addition, the 10-checkpoint setting is considered as a balanced case that is neither overly simple nor highly complex. This intermediate level enables a clearer analysis of algorithm behavior under moderate mission difficulty, providing a more practical view of performance. Together, these two settings offer a more comprehensive understanding of how the algorithms behave under both balanced and complex tasks as obstacle density progressively increases.

Case 1: 20 fixed checkpoints and increasing obstacle counts

Based on Table 7, as the number of obstacles increases from 5 to 15, a general upward trend is observed across all three algorithms in terms of path length, energy consumption, and runtime, which is expected given that higher obstacle density forces longer detours and more complex replanning. However, HCTS presents a notable exception, as its path length slightly decreases at 10 obstacles (409.83) compared to 5 obstacles (412.92), before rising again at 15 obstacles (421.02). This non-monotonic behavior may suggest that HCTS benefits from a moderate level of environmental complexity, where the additional obstacles introduce structural constraints that inadvertently guide the algorithm toward more efficient routing decisions. CS and GA, by contrast, follow a more consistent increasing trend across all obstacle levels, indicating a more predictable but less adaptive response to environmental changes.

In terms of path length, HCTS consistently achieves the shortest values across all obstacle configurations. At 5 obstacles, HCTS records 412.92 compared to 720.79 for CS and 661.42 for GA, representing a substantial advantage. This gap is maintained at 10 and 15 obstacles, where HCTS records 409.83 and 421.02 respectively, while CS and GA remain considerably higher. A similar pattern is observed in energy consumption, where HCTS consistently records the lowest values across all three obstacle levels, with readings of 176,474.93, 175,883.53, and 178,195.96 at 5, 10, and 15 obstacles respectively. CS produces the highest energy consumption throughout, peaking at 246,664.01 at 15 obstacles, while GA falls in between. The relatively stable energy figures for HCTS further reinforce its efficiency under increasing environmental pressure. Regarding runtime, CS achieves the fastest execution across all configurations, while HCTS incurs the highest computational cost, as discussed in the Section 5.3.3. Despite this, the superior path quality and energy efficiency demonstrated by HCTS reflect a meaningful performance trade-off that favors solution quality over computational efficiency.

The statistical significance of the observed performance differences is confirmed through pairwise comparisons between HCTS and each of the competing algorithms. As presented in Table 8, for HCTS vs. CS, all p-values for both path length and energy consumption are uniformly recorded at 1.69×10^{-17} across all obstacle configurations (5, 10, and 15 obstacles). For HCTS vs. GA, p-values for path length range from 3.38×10^{-17} to 3.21×10^{-16}, and for energy consumption range from 1.69×10^{-17} to 3.38×10^{-17}. All values fall substantially below the significance threshold of 0.05, providing strong statistical evidence that the performance advantages demonstrated by HCTS over CS and GA are not attributable to random variation but rather reflect genuine and consistent algorithmic superiority.

Table 7. Comparison results of Scenario II: 20 fixed checkpoints and increasing obstacle counts

Level	Algo	Path Length	Energy	Runtime (s)
20 CP, 5 Obs	HCSTS	412.92 ± 31.32	176,474.93 ± 6984.89	2.46
	CS	720.79 ± 28.42	245,197.18 ± 6300.70	0.73
	GA	661.42 ± 75.09	231,964.62 ± 16,765.84	0.92
20 CP, 10 Obs	HCSTS	409.83 ± 34.36	175,883.53 ± 7655.79	2.94
	CS	722.95 ± 31.25	245,709.44± 6926.22	0.81
	GA	636.23 ± 56.40	226,377.15 ± 12,585.28	1.176
20 CP, 15 Obs	HCSTS	421.02 ± 38.58	178,195.96 ± 8546.63	3.29
	CS	727.06 ± 44.62	246,664.01 ± 9933.93	0.93
	GA	661.13 ± 54.83	231,899.81 ± 12,146.01	1.35

Table 8. Statistical validation of Scenario II: 20 fixed checkpoints and increasing obstacle counts

Scenario	HCTS vs. CS		HCTS vs. GA	
	p-Value (Length)	*p*-Value (Energy)	*p*-Value (Length)	*p*-Value (Energy)
20 CP, 5 Obs	1.69×10^{-17}	1.69×10^{-17}	3.38×10^{-17}	3.38×10^{-17}
20 CP, 10 Obs	1.69×10^{-17}	1.69×10^{-17}	1.69×10^{-17}	1.69×10^{-17}
20 CP, 15 Obs	1.69×10^{-17}	1.69×10^{-17}	3.21×10^{-16}	2.03×10^{-16}

The relatively high standard deviation values observed in Table 7, along with the elevated effectiveness rates reported in Table 9, can be attributed to the stochastic nature inherent in all three algorithms under evaluation. As metaheuristic algorithms, HCTS, CS, and GA rely on probabilistic search mechanisms that introduce an element of randomness into the solution generation process across independent runs. This variability becomes particularly pronounced in the 20-checkpoint configuration, where the expanded mission size significantly enlarges the solution search space.

The effectiveness rates reported in Table 9 reflect this variability across all obstacle levels. At 5 obstacles, HCTS records effectiveness rates of 16.35 for length and 8.08 for energy, while CS records 7.8 and 4.9, and GA records 33.6 and 18.9 respectively. At 10 obstacles, HCTS records 15.15 and 7.48, CS records 11.4 and 7.3, and GA records 17.5 and 10.4. At 15 obstacles, HCTS records 18.5 and 8.9, CS records 29.2 and 17.4, and GA records 22.8 and 13.27. Notably, GA exhibits the highest effectiveness rates at lower obstacle levels, while CS shows a considerable increase as obstacle density rises to 15, suggesting that different algorithms are affected by search space expansion in distinct ways.

As the number of required checkpoints increases, the combinatorial complexity of the routing problem grows substantially, making it increasingly difficult for stochastic algorithms to converge consistently to the same solution across repeated trials. Consequently, different runs may explore distinct regions of the search space, yielding solutions of varying quality and thus producing wider performance distributions.

Furthermore, when obstacle density is relatively low, the search space becomes less constrained, offering a greater number of feasible paths of comparable cost. This abundance of near-optimal alternatives amplifies inter-run variability, as the algorithms may converge to different locally optimal solutions without a definitive structural preference. Collectively, these factors (mission size, search space expansion, and reduced environmental constraints), provide a coherent explanation for the elevated standard deviation and effectiveness rates observed across this scenario, and are consistent with the well-documented behavioral characteristics of metaheuristic optimization under large-scale, loosely constrained problem instances.

Table 9. Effectiveness of Scenario II: 20 fixed checkpoints and increasing obstacle counts.

Level	Effectiveness					
	HCTS		CS		GA	
	Length	Energy	Length	Energy	Length	Energy
20 CP, 5 Obs	16.35	8.08	7.8	4.9	33.6	18.9
20 CP, 10 Obs	15.15	7.48	11.4	7.3	17.5	10.4
20 CP, 15 Obs	18.5	8.9	29.2	17.4	22.8	13.27

Further insight into algorithmic consistency is provided by the success rate analysis presented in Table 10. As outlined in Section 5.2, the success rate criterion and its conditions have been previously defined. The results reveal uniformly low success rates across all algorithms and obstacle configurations, which is consistent with the high standard deviation values previously discussed. At 5 obstacles, HCTS achieves a success rate of 3% for both length and energy, while CS records 6% for both metrics, and GA records 3% for both. At 10 obstacles, HCTS improves to 6% for length and 10% for energy, CS maintains 6% across both metrics, and GA records 6% for length and 3% for energy. At 15 obstacles, HCTS records 6% for length and 10% for energy, CS remains at 6% for both, and GA records 6% for length and 10% for energy. The consistently low success rates observed across all configurations reflect the inherent difficulty of repeatedly converging to near-optimal solutions, particularly given the large and loosely constrained search space associated with the 20-checkpoint mission. These results further corroborate the stochastic variability discussed in the context of standard deviation and effectiveness rates, and collectively underscore the challenge that metaheuristic algorithms face in achieving stable convergence under complex, large-scale routing conditions.

Table 10. Success rate for of Scenario II: 20 fixed checkpoints and increasing obstacle counts.

Level	Success Rate					
	HCTS		CS		GA	
	Length	Energy	Length	Energy	Length	Energy
20 CP, 5 Obs	3%	3%	6%	6%	3%	3%
20 CP, 10 Obs	6%	10%	6%	6%	6%	3%
20 CP, 15 Obs	6%	10%	6%	6%	6%	10%

Case 2: 10 fixed checkpoints and increasing obstacle counts

This case examines algorithm performance under a balanced mission setting, where the number of checkpoints is fixed at 10 and the number of obstacles is incrementally increased from 5 to 20, providing a moderately complex evaluation environment that complements the high-demand conditions analyzed in the preceding case.

Table 11 presents the performance results for the balanced scenario comprising 10 fixed checkpoints with obstacle counts increasing from 5 to 20. Across all obstacle configurations, HCTS consistently achieves the shortest path length and lowest energy consumption. At 5 obstacles, HCTS records a path length of 308.07 and energy consumption of 114,737.9, compared to 362.8 and 126,903.9 for CS, and 354.06 and 124,933.14 for GA. At 15 obstacles, HCTS records 308.67 and 114,858.3, while CS records 370.6 and 128,547.78, and GA records 363.73 and 127,080.92. At 20 obstacles, HCTS records 306.19 and 114,299.5, CS records 361.63 and 126,580.16, and GA records 355.53 and 125,256.58.

A particularly noteworthy observation in this balanced scenario is that HCTS exhibits a non-monotonic trend in path length, where the value slightly decreases from 308.07 at

5 obstacles to 306.19 at 20 obstacles, with a marginal intermediate increase at 15 obstacles. This adaptive behavior suggests that HCTS is capable of exploiting the structural constraints introduced by higher obstacle density to identify more efficient routing solutions, even under moderate mission complexity. CS and GA, by contrast, display relatively stable but consistently higher path lengths across all obstacle levels, confirming their comparatively limited adaptability regardless of mission scale.

Regarding runtime, CS again achieves the fastest execution while HCTS incurs the highest computational cost across all obstacle levels, consistent with the trade-off observed in the preceding case and attributable to the additional overhead of the Tabu Search mechanism.

Table 11. Comparison of Scenario II: 10 fixed checkpoints and increasing obstacle counts.

Level	Algo	Path Length	Energy	Runtime (s)
10 CP, 5 Obs	HCTS	308.07 ± 9.70	114,737.9 ± 2185.16	0.65
	CS	362.8 ± 17.79	126,903.9 ± 3964.5	0.43
	GA	354.06 ± 29.91	124,933.14 ± 6638.76	0.53
10 CP, 15 Obs	HCTS	308.67 ± 10.06	114,858.3 ± 2238.29	0.89
	CS	370.6 ± 15.34	128,547.7819± 3374.6	0.60
	GA	363.73 ± 30.40	127,080.92 ± 6779.49	0.77
10 CP, 20 Obs	HCTS	306.19 ± 9	114,299.5 ± 2012.01	0.97
	CS	361.63 ± 18.72	126,580.16 ±4133.87	0.62
	GA	355.53 ± 36.25	125,256.58 ± 8083.77	0.89

The statistical validation results for the 10-checkpoint case, presented in Table 12, confirm that the performance advantages of HCTS over both CS and GA are statistically significant across all obstacle configurations. For HCTS vs. CS, p-values for path length range from 1.64×10^{-15} to 7.62×10^{-14}, and for energy consumption from 1.64×10^{-15} to 1.98×10^{-13} . For HCTS vs. GA, p-values for path length range from 7.96×10^{-11} to 4.13×10^{-10}, and for energy consumption from 9.30×10^{-11} to 4.13×10^{-10}. All p-values fall substantially below the 0.05 significance threshold across all obstacle levels. These results are consistent with those observed in the 20-checkpoint scenario, further reinforcing that the superiority of HCTS is statistically robust and not a product of random variation, regardless of mission complexity or obstacle density.

Table 12. Statistical validation of Scenario II: 10 fixed checkpoints and increasing obstacle counts.

Scenario	HCTS vs. CS		HCTS vs. GA	
	p-Value (Length)	p-Value (Energy)	p-Value (Length)	p-Value (Energy)
10 CP, 5 Obs	1.64×10^{-15}	1.64×10^{-15}	7.96×10^{-11}	9.30×10^{-11}
10 CP, 15 Obs	1.69×10^{-17}	1.69×10^{-17}	2.04×10^{-12}	1.67×10^{-12}
10 CP, 20 Obs	7.62×10^{-14}	1.98×10^{-13}	4.13×10^{-10}	4.13×10^{-10}

The effectiveness rates presented in Table 13 reveal a notably different pattern compared to those observed in the 20-checkpoint case in Table 9. Under the 10-checkpoint configuration, HCTS records considerably lower effectiveness rates across all obstacle levels, with values of 2.04 and 1.39 at 5 obstacles, 2.39 and 1.47 at 15 obstacles, and 1.55 and 1.03 at 20 obstacles for length and energy respectively.

These values are substantially lower than those recorded in the 20-checkpoint case, where HCTS recorded rates as high as 16.35 and 18.5 for length. A similar reduction is observed for CS and GA, where effectiveness rates in the 10-checkpoint scenario are generally lower than their counterparts in the more complex mission setting. This consistent

reduction in effectiveness rates across all algorithms as mission size decreases from 20 to 10 checkpoints directly supports the earlier argument regarding search space size as a primary driver of algorithmic variability. When the mission scope is reduced, the search space contracts accordingly, enabling the algorithms to converge more consistently to near-optimal solutions across independent runs. These findings collectively validate the hypothesis that metaheuristic performance stability is strongly influenced by the dimensionality of the problem, and that a more constrained search space yields more reproducible and reliable outcomes.

Table 13. Effectiveness of Scenario II: 10 fixed checkpoints and increasing obstacle counts

Level	Effectiveness					
	HCTS		CS		GA	
	Length	Energy	Length	Energy	Length	Energy
10 CP, 5 Obs	2.04	1.39	12.54	7.73	16.13	9.56
10 CP, 15 Obs	2.39	1.47	8.21	5.04	22.52	12.26
10 CP, 20 Obs	1.55	1.03	19.43	11.52	18.06	10.76

The success rate results presented in Table 14 further corroborate the improvement in algorithmic consistency observed in the 10-checkpoint case compared to the 20-checkpoint case in Table 10. HCTS demonstrates a remarkable increase in success rates, recording 56% and 60% at 5 obstacles, 73% and 73% at 15 obstacles, and 80% and 80% at 20 obstacles for length and energy respectively. These values stand in stark contrast to the success rates recorded in the 20-checkpoint case, where HCTS achieved a maximum of only 6%, highlighting the substantial impact of reduced mission complexity on convergence stability. CS and GA, however, maintain comparatively low success rates across all obstacle levels, with CS recording 6%, 6%, and 3% for length and 6%, 10%, and 3% for energy, while GA records 6%, 3%, and 6% for length and 13%, 3%, and 6% for energy across the three obstacle levels.

Table 14. Success rate of Scenario II: 10 fixed checkpoints and increasing obstacle counts

Level	Success Rate					
	HCTS		CS		GA	
	Length	Energy	Length	Energy	Length	Energy
10 CP, 5 Obs	56%	60%	6%	6%	6%	13%
10 CP, 15 Obs	73%	73%	6%	10%	3%	3%
10 CP, 20 Obs	80%	80%	3%	3%	6%	6%

This divergence suggests that although the reduction in search space benefits all algorithms to some extent, HCTS gains greater advantage from the more constrained problem setting. This is likely because the TS mechanism becomes more effective in consistently identifying near-optimal visiting sequences when the solution space is less expansive. These findings reinforce the conclusion that problem dimensionality plays a critical role not only in solution quality, but also in the reproducibility and convergence reliability of metaheuristic algorithms.

This behavior also explains why HCTS outperforms standard CS and GA. In standard CS, Lévy-flight-based updates provide broad exploration; however, when CS is used alone, the generated changes may be too wide and may not sufficiently refine promising visiting orders. GA, on the other hand, relies on crossover and mutation, which may disrupt useful partial visiting sequences and delay convergence toward efficient routes. In contrast, HCTS

combines the diversification ability of CS with the local refinement capability of TS. This allows promising checkpoint orders to be improved through neighborhood-based moves, reducing unnecessary detours between consecutive checkpoints and leading to shorter paths and lower energy consumption.

From a practical perspective, the above results indicate that improving the checkpoint-visiting order can directly reduce the total travel distance and energy consumption of UAV missions. This is reflected in the lower distance and energy values achieved by the proposed HCTS compared with the benchmark algorithms. Such improvement is particularly important in inspection and delivery applications, where UAVs are often constrained by limited battery capacity and need to visit multiple locations within a single mission. Therefore, HCTS can support more efficient mission planning in urban environments by producing shorter and more energy-aware visiting sequences while satisfying obstacle-avoidance constraints.

5.4. Computational Complexity

The computational complexity of HCTS is also calculated and compared with CS and GA. Let us assume that p represents the number of nests (i.e., the population size), T is the number of iterations, m is the number of checkpoints in the UAV path, and K is the number of tabu search perturbations per iteration. Table 15 provides the step-by-step breakdown of this calculation.

Table 15. Computational complexity of HCTS.

Step	Operation	Cost
Initialization	Generate and evaluate p random tours	(O(p•m))
Lévy flight generation	p new tours × (perturbation + evaluation)	(O(p•m))
Replacement step	Compare and replace p tours	(O(p•m))
Abandonment step	Possibly regenerate and evaluate tours	(O(p•m))
Sorting and update	Sort p tours by fitness	(O(p•log p))
Tabu Search	K perturbations, each costing (O(m))	(O(K•m))
Total per iteration	Sum of all above	O(p•(m + log p) + K•m)

Using the calculations given in Table 15, the total complexity for the T iterations is O(T•[p•(m + log p) + K•m]). In contrast, the total complexity of the CS algorithm is O(T•p•(m + log p)). Furthermore, the computational complexity of GA is O(T•p•($f(L)$+ L)), where L is the length of a chromosome and $f(L)$ is the time to evaluate the fitness of one individual.

The increase in complexity in HCTS arises from the incorporation of TS, which introduces additional memory usage and search operations to help avoid local minima. This shows that while HCTS has a higher computational complexity due to its hybrid nature, the extra computational cost is justified by its ability to find better solutions, as discussed in the previous sections. Therefore, the trade-off between computational complexity and solution quality should be carefully considered when choosing between these algorithms for UAV path planning missions.

6. Conclusions

This study introduced the HCTS algorithm, a hybrid metaheuristic designed to address the complex multi-objective challenges of UAV path planning in obstacle-dense urban environments. By integrating Cuckoo Search with Tabu Search and employing fuzzy logic, the proposed approach provides a robust approach for optimizing path length and energy efficiency simultaneously.

The primary strength of HCTS lies in its superior exploration and exploitation capabilities. Unlike conventional metaheuristics that often suffer from premature convergence in complex terrains, the hybrid nature of HCTS allows it to maintain a high level of reliability. Compared with the two baseline algorithms, HCTS achieved reductions in path length ranging from 0.01% to 42.11% and reductions in energy consumption ranging from 0.08% to 27.91%, depending on scenario complexity. This is further supported by the success rate results, where HCTS maintained a high success rate of 96–100% in the main fixed-complexity scenarios. In contrast, the baseline algorithms showed a clear decline in more complex environments, with success rates dropping to 3–13%. These results indicate that HCTS is more reliable in generating feasible solutions under increasing mission complexity. Moreover, the performance gap between HCTS and the competing algorithms becomes more evident as obstacle density and checkpoint requirements increase, highlighting the scalability of the proposed approach for large-scale UAV mission planning.

Despite these strengths, certain trade-offs and limitations were identified. A notable weakness is the increased computational overhead associated with the Tabu Search refinement process, resulting in higher execution times compared to the basic Cuckoo Search. While this trade-off is often acceptable for pre-flight path planning where path quality and safety are paramount, it may pose challenges for real-time applications requiring instantaneous re-routing.

Furthermore, when the number of checkpoints is large and the environmental constraints are relatively limited, the combinatorial search space becomes more expansive, which may hinder consistent convergence. This is reflected in the reduced success rates observed under high-checkpoint and low-obstacle configurations. In such cases, the algorithm may explore different regions of the solution space across independent runs, leading to higher variability in solution quality and larger standard deviation values. Nevertheless, HCTS maintained lower average path length and energy consumption compared with the baseline algorithms across the tested configurations. This indicates that although the expanded search space affects convergence consistency, the hybrid search mechanism remains effective in guiding the search toward shorter and more energy-efficient visiting sequences. Additionally, the current model assumes a static environment, which does not fully account for the unpredictable dynamics of real-world urban airspace.

Moving forward, future research will focus on optimizing the computational efficiency of HCTS to bridge the gap between path quality and execution speed. Potential directions include the integration of parallel computing or adaptive parameter tuning. Moreover, extending the model to incorporate dynamic obstacle avoidance and real-time sensor data will be a crucial step toward deploying this algorithm in autonomous UAV systems operating in evolving and uncertain environments.

Author Contributions: Conceptualization, G.A., A.H. and A.A.; methodology, G.A., A.H., A.A., and L.M.; software, G.A.; validation, G.A., A.H. and A.A.; formal analysis, G.A., A.H. and A.A.; investigation, G.A.; resources, G.A.; data curation, G.A., A.H. and A.A.; writing—original draft preparation, G.A.; writing—review and editing, G.A., A.H., A.A. and L.M.; visualization, G.A.; supervision, A.H. and A.A.; project administration, G.A., A.H. and A.A.; funding acquisition, A.H. and A.A. All authors have read and agreed to the published version of the manuscript.

Funding: This Project was funded by the Deanship of Scientific Research (DSR) at King Abdulaziz University, Jeddah, Saudi Arabia under grant no.(IPP: 990-612-2025). The authors, therefore, acknowledge with thanks DSR for technical and financial support.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The code and data are available with the authors and can be provided upon request after indicating the purpose of use.

Acknowledgments: During the preparation of this manuscript, the authors used Grammarly for grammar checking and Claude (Anthropic) to assist in editing and polishing the written text. All content was originally written by the authors, and they reviewed all suggestions and take full responsibility for the final content of this publication.

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

Abbreviations

The following abbreviations and symbols are used in this manuscript:

UAV	Unmanned aerial vehicle
TSP	Traveling salesman problem
CS	Cuckoo search
TS	Tabu search
HCTS	Hybrid cuckoo search–tabu search
WSM	Weighted sum method
ACO	Ant colony optimization
NSGA II	Non-dominated sorting genetic algorithm II
FM^2	Fast marching square
DRL	Deep reinforcement learning
DDQN	Double deep Q-network
IPO	Improved parrot optimization
ZAMOPSO	Zaslavskii chaotic multi-objective particle swarm optimization
i	A checkpoint on the path
j	Another checkpoint on the path
n	Number of checkpoints
$rand$	Random number operator
x_{L1} and x_{U1}	Boundary values of the plane area
B_{ci}	Center of the ith point
B_{hi}	Height of the ith point
h	Height limitation for points
L_i	Path length of the mission
E_i	Energy used during mission
p_{min}	Minimum power
$\hat{v}$	Induced velocity required for a given thrust T
T	Thrust
v	Average ground speed of UAV
β	Pitch angle
m	Mass of UAV (body + battery)
g	Gravitational constant
f_d	Drag force
ρ	Air density
r	UAV rotor diameter
q	Number of UAV rotors
d	Traversed distance
$E_f(k)$	Total flying energy of UAV k
η_k	Power efficiency of UAV k
$p_h(k)$	Power consumption of hovering of UAV k
p_h^{min}	Theoretical minimum power for hovering
$E_h(k)$	Hovering energy per stop

$t_h(k)$	Hovering time per stop
$E_{h\_total}(k)$	Total hovering energy
$E(k)$	Total energy consumption
d_{safe}	Safe distance
O_i	Obstacles
d_{UAV}	Maximum diameter of UAV
p_d	UAV trajectory segment between two consecutive checkpoints
$\mu_R(x)$	Membership function for route
$\mu_L(x)$	Membership function for path length
$\mu_E(x)$	Membership function for energy consumption
w_1	Weight associated with $\mu_L(x)$
w_2	Weight associated with $\mu_E(x)$
L_{max}	Upper limit for $\mu_L(x)$
L_{min}	Lower limit for $\mu_L(x)$
E_{max}	Upper limit for $\mu_E(x)$
E_{min}	Lower limit for $\mu_E(x)$
D_{env}	Maximum Euclidean distance across the environment
p_f^{nom}	Nominal forward flight power
v_{nom}	Nominal UAV velocity
E_{hover}	Hovering energy consumed at each checkpoint
α	Step size in HCSTS and CS algorithms
λ	Control parameter for *Lévy* flight
p_a	Probability of abundance
TLS	Tabu list size
$p_{average}$	Average fitness of solutions
p_{opt}	Optimum fitness
$\eta_{quality}$	Success rate
$N_{successful}$	Number of good solutions
N_{sim}	Total number of simulations performed

References

1. Zhou, X.; Gao, F.; Fang, X.; Lan, Z. Improved Bat Algorithm for UAV Path Planning in Three-Dimensional Space. *IEEE Access* **2021**, *9*, 20100–20116. [CrossRef]
2. Ahmed, F.; Mohanta, J.C.; Keshari, A.; Yadav, P.S. Recent Advances in Unmanned Aerial Vehicles: A Review. *Arab. J. Sci. Eng.* **2022**, *47*, 7963–7984. [CrossRef]
3. Wu, Q.; Su, Y.; Tan, W.; Zhan, R.; Liu, J.; Jiang, L. UAV Path planning trends from 2000 to 2024: A bibliometric analysis and visualization. *Drones* **2025**, *9*, 128. [CrossRef]
4. Bashir, N.; Boudjit, S.; Dauphin, G.; Zeadally, S. An obstacle avoidance approach for UAV path planning. *Simul. Model. Pract. Theory* **2023**, *129*, 102815. [CrossRef]
5. Sonny, A.; Yeduri, S.R.; Cenkeramaddi, L.R. Q-learning-based unmanned aerial vehicle path planning with dynamic obstacle avoidance. *Appl. Soft Comput.* **2023**, *147*, 110773. [CrossRef]
6. Pehlivanoglu, Y.V.; Pehlivanoglu, P. An enhanced genetic algorithm for path planning of autonomous UAV in target coverage problems. *Appl. Soft Comput.* **2021**, *112*, 107796. [CrossRef]
7. Ren, Q.; Yao, Y.; Yang, G.; Zhou, X. Multi-objective path planning for UAV in the urban environment based on CDNSGA-II. In Proceedings of the 2019 IEEE International Conference on Service-Oriented System Engineering (SOSE), San Francisco, CA, USA, 4–9 April 2019; pp. 350–3505.
8. Yang, X.S.; Deb, S. Cuckoo search via Lévy flights. In Proceedings of the 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC), Coimbatore, India, 9–11 December 2009; pp. 210–214.
9. Glover, F. Tabu search—Part I. *ORSA J. Comput.* **1989**, *1*, 190–206. [CrossRef]
10. Meng, W.; Zhang, X.; Zhou, L.; Guo, H.; Hu, X. Advances in UAV Path Planning: A Comprehensive Review of Methods, Challenges, and Future Directions. *Drones* **2025**, *9*, 376. [CrossRef]
11. Rehman, S.; Khan, S.A. Fuzzy logic based multi-criteria wind turbine selection strategy—A case study of Qassim, Saudi Arabia. *Energies* **2016**, *9*, 872. [CrossRef]

12. Bahwini, T.; Zhong, Y.; Gu, C. Path planning in the presence of soft tissue deformation. *Int. J. Interact. Des. Manuf. (IJIDeM)* **2019**, *13*, 1603–1616. [CrossRef]
13. Zhong, Y.; Shirinzadeh, B.; Yuan, X. Optimal robot path planning with cellular neural network. In *Advanced Engineering and Computational Methodologies for Intelligent Mechatronics and Robotics*; IGI Global Scientific Publishing: Hershey, PA, USA, 2013; pp. 19–38.
14. Zhou, Q.; Gao, S.; Qu, B.; Gao, X.; Zhong, Y. Crossover recombination-based global-best brain storm optimization algorithm for uav path planning. *Proc. Rom. Acad. Ser. A* **2022**, *23*, 209–218.
15. Hills, J.; Zhong, Y. Cellular neural network-based thermal modelling for real-time robotic path planning. *Int. J. Agil. Syst. Manag.* **2014**, *7*, 261–281. [CrossRef]
16. Primatesta, S.; Guglieri, G.; Rizzo, A. A risk-aware path planning strategy for UAVs in urban environments. *J. Intell. Robot. Syst.* **2019**, *95*, 629–643. [CrossRef]
17. Muñoz, J.; López, B.; Quevedo, F.; Monje, C.A.; Garrido, S.; Moreno, L.E. Multi UAV coverage path planning in urban environments. *Sensors* **2021**, *21*, 7365. [CrossRef]
18. Sadallah, N.; Yahiaoui, S.; Bendjoudi, A.; Nouali-Taboudjemat, N. Multi-objective offline and online path planning for UAVs under dynamic urban environment. *Int. J. Intell. Robot. Appl.* **2022**, *6*, 119–138. [CrossRef]
19. Tang, H.; Zhu, Q.; Qin, B.; Song, R.; Li, Z. UAV path planning based on third-party risk modeling. *Sci. Rep.* **2023**, *13*, 22259. [CrossRef]
20. Rienecker, H.; Hildebrand, V.; Pfifer, H. Energy optimal 3D flight path planning for unmanned aerial vehicle in urban environments. *CEAS Aeronaut. J.* **2023**, *14*, 621–636. [CrossRef]
21. Song, J.; Zhou, R. Improved UAV Path Planning in Urban Environment Based on A-Star Method. In Proceedings of the 2025 2nd International Conference on Mechanics, Electronics Engineering and Automation (ICMEEA 2025), Toronto, ON, Canada, 16–18 May 2025; pp. 1157–1167.
22. Tian, S.; Li, Y.; Zhang, X.; Zheng, L.; Cheng, L.; She, W.; Xie, W. Fast UAV path planning in urban environments based on three-step experience buffer sampling DDPG. *Digit. Commun. Netw.* **2024**, *10*, 813–826. [CrossRef]
23. Zhu, Y.; Tan, Y.; Chen, Y.; Chen, L.; Lee, K.Y. Uav path planning based on random obstacle training and linear soft update of drl in dense urban environment. *Energies* **2024**, *17*, 2762. [CrossRef]
24. Sarhan, S.; Rinaldi, M.; Primatesta, S.; Guglieri, G. Noise-aware UAV path planning in urban environment with reinforcement learning. *Eng. Proc.* **2025**, *90*, 3.
25. Wu, J.; Wang, H.; Li, N.; Yao, P.; Huang, Y.; Yang, H. Path planning for solar-powered UAV in urban environment. *Neurocomputing* **2018**, *275*, 2055–2065. [CrossRef]
26. Hu, X.; Pang, B.; Dai, F.; Low, K.H. Risk assessment model for UAV cost-effective path planning in urban environments. *IEEE Access* **2020**, *8*, 150162–150173. [CrossRef]
27. Majeed, A.; Hwang, S.O. A multi-objective coverage path planning algorithm for UAVs to cover spatially distributed regions in urban environments. *Aerospace* **2021**, *8*, 343. [CrossRef]
28. Wei, X.; Xu, J. Distributed path planning of unmanned aerial vehicle communication chain based on dual decomposition. *Wirel. Commun. Mob. Comput.* **2021**, *2021*, 6661926. [CrossRef]
29. Yu, X.; Luo, W. Reinforcement learning-based multi-strategy cuckoo search algorithm for 3D UAV path planning. *Expert Syst. Appl.* **2023**, *223*, 119910. [CrossRef]
30. Cao, H.; Li, S.; Li, X.; Liu, Y. A UAV Path-Planning Approach for Urban Environmental Event Monitoring. *Comput. Mater. Contin.* **2025**, *83*, 5575. [CrossRef]
31. Ma, H.; Gao, Y.; Zhou, Z. Three-Dimensional UAV Path Planning in Urban Environments Based on an Improved Parrot Optimization Algorithm. *Inf. Technol. Control* **2025**, *54*, 821–843. [CrossRef]
32. Lin, C.; Xu, H.; Chen, X. Research on Urban UAV Path Planning Technology Based on Zaslavskii Chaotic Multi-Objective Particle Swarm Optimization. *Symmetry* **2025**, *17*, 1711. [CrossRef]
33. Cheng, Q.; Zhang, Z.; Du, Y.; Li, Y. Research on particle swarm optimization-based UAV path planning technology in urban airspace. *Drones* **2024**, *8*, 701. [CrossRef]
34. Debnath, D.; Vanegas, F.; Boiteau, S.; Gonzalez, F. An Integrated Geometric Obstacle Avoidance and Genetic Algorithm TSP Model for UAV Path Planning. *Drones* **2024**, *8*, 302. [CrossRef]
35. Shivgan, R.; Dong, Z. Energy-efficient drone coverage path planning using genetic algorithm. In Proceedings of the 2020 IEEE 21st International Conference on High Performance Switching and Routing (HPSR), Newark, NJ, USA, 11–14 May 2020; pp. 1–6.
36. Monwar, M.; Semiari, O.; Saad, W. Optimized path planning for inspection by unmanned aerial vehicles swarm with energy constraints. In Proceedings of the 2018 IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, United Arab Emirates, 9–13 December 2018; pp. 1–6.
37. Stolaroff, J.K.; Samaras, C.; O'Neill, E.R.; Lubers, A.; Mitchell, A.S.; Ceperley, D. Energy use and life cycle greenhouse gas emissions of drones for commercial package delivery. *Nat. Commun.* **2018**, *9*, 409. [CrossRef]

38. Miettinen, K. Some methods for nonlinear multi-objective optimization. In Proceedings of the International Conference on Evolutionary Multi-Criterion Optimization, Zürich, Switzerland, 7–9 March 2001; pp. 1–20.
39. Gao, L.S. The fuzzy arithmetic mean. *Fuzzy Sets Syst.* **1999**, *107*, 335–348. [CrossRef]
40. Pavlačka, O.; Pavlačková, M.; Hetfleiš, V. Fuzzy weighted average as a fuzzified aggregation operator and its properties. *Kybernetika* **2017**, *53*, 137–160. [CrossRef]
41. Tawfeek, M.; Alrashdi, I.; Alruwaili, M.; Talaat, F. A fuzzy multi-objective framework for energy optimization and reliable routing in wireless sensor networks via particle swarm optimization. *Comput. Mater. Contin.* **2025**, *83*, 2773. [CrossRef]
42. Seyyedabbasi, A.; Tareq Tareq, W.Z.; Bacanin, N. An effective hybrid metaheuristic algorithm for solving global optimization algorithms. *Multimed. Tools Appl.* **2024**, *83*, 85103–85138. [CrossRef]
43. Mahmood, A.; Khan, S.A.; Albalooshi, F.; Awwad, N. Energy-aware real-time task scheduling in multiprocessor systems using a hybrid genetic algorithm. *Electronics* **2017**, *6*, 40. [CrossRef]
44. Sivanandam, S.; Deepa, S. Genetic algorithms. In *Introduction to Genetic Algorithms*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 15–37.
45. Sadiq, S.; Habib, Y. *Iterative Computer Algorithms with Applications in Engineering: Solving Combinatorial Optimization Problems*; IEEE: Los Alamitos, CA, USA, 1999; p. 387.
46. Derrac, J.; García, S.; Molina, D.; Herrera, F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm Evol. Comput.* **2011**, *1*, 3–18. [CrossRef]
47. Campuzano, G.; Obreque, C.; Aguayo, M.M. Accelerating the Miller–Tucker–Zemlin model for the asymmetric traveling salesman problem. *Expert Syst. Appl.* **2020**, *148*, 113229. [CrossRef]
48. Mora-Meliá, D.; Iglesias-Rey, P.L.; Martínez-Solano, F.J.; Ballesteros-Perez, P. Efficiency of evolutionary algorithms in water network pipe sizing. *Water Resour. Manag.* **2015**, *29*, 4817–4831. [CrossRef]
49. Liang, J.J.; Runarsson, T.P.; Mezura-Montes, E.; Clerc, M.; Suganthan, P.N.; Coello, C.C.; Deb, K. Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization. *J. Appl. Mech.* **2006**, *41*, 8–31.