robotics

MDPI

Article

Improved Multi-Objective Cuckoo-Catfish Optimizer for Smooth and Collision-Free Mobile Robot Path Planning

Jaafar Ahmed Abdulsaheb * and Mohanad Azeez Joodi

Department of Electrical Engineering, College of Engineering, University of Baghdad, Baghdad 10001, Iraq; m.mohammad1702d@coeng.uobaghdad.edu.iq

* Correspondence: jaafar.ahmed@coeng.uobaghdad.edu.iq

Abstract

In this paper, an Adaptive Improved Cuckoo-Catfish Optimizer (AICCO) is proposed for smooth and collision-free mobile robot path planning in static and dynamic environments. The proposed AICCO enhances the original Cuckoo-Catfish Optimizer (CCO) by integrating chaotic opposition-based initialization, nonlinear adaptive control, elite-guided search, strong elite preservation, memetic local refinement, and stagnation-based opposition repair. A weighted-sum scalarized fitness function is formulated to minimize path length and turning-angle variation while maximizing obstacle clearance. The proposed method is evaluated using 23 benchmark functions and further validated in static, dynamic, and nonlinear/reactive obstacle navigation scenarios. The benchmark results show that AICCO achieves the best overall rank among the optimizers compared. In the static planning scenario, the method achieves zero collision penalty and a minimum clearance of 0.785565. In the dynamic online replanning scenario, it achieves zero collisions with a minimum clearance of 1.514250. An additional nonlinear/reactive dynamic scenario further demonstrates that the proposed method can maintain collision-free navigation under more complex obstacle motion.

Keywords: mobile robot path planning; Cuckoo-Catfish Optimizer; Adaptive Improved CCO; weighted-sum scalarization; multi-objective optimization; obstacle avoidance; path smoothness; collision-free navigation; dynamic obstacles; online replanning

1. Introduction

Mobile robot path planning is a fundamental problem in autonomous navigation, where a robot must move from a start point to a target position through a feasible and collision-free route. In practical applications, the generated path should not only avoid obstacles but should also be short, safe, and smooth enough for execution by a real robotic platform. Therefore, optimizing only the path length is usually insufficient, because path length, obstacle clearance, and path smoothness are often conflicting objectives in mobile robot navigation. Recent studies have therefore formulated mobile robot path planning as a multi-objective optimization problem that simultaneously considers path length, path safety, and path smoothness [1–4]. Classical planners such as A*, Dijkstra, RRT, and artificial potential field are widely used because of their simplicity and ability to generate feasible paths [5–8].

Much attention has been given to metaheuristic and swarm-intelligence algorithms since they are capable of addressing nonlinear, complex, and multi-objective path-planning problems. Path length, safety, and smoothness have been optimized using

Check for updates

Academic Editor: Shahram Payandeh

Received: 8 May 2026
Revised: 7 June 2026
Accepted: 11 June 2026
Published: 15 June 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

algorithms like PSO, ABC, ACO, NSGA-II and others, which are evolutionary algorithms. However, most of these approaches remain prematurely converged, poorly exploited, computationally expensive, or trapped in local minima in cluttered spaces. These constraints are further aggravated when the planner needs to come up with a path that is not only free of collisions, but also one which is smooth and maintains sufficient clearance from obstacles [9–12].

The Cuckoo-Catfish Optimizer (CCO) is a new metaheuristic method of optimization that is based on the behavior of the cuckoo catfish. The original CCO contains a variety of search mechanisms, including surround search, compressed-space search, transition search, chaotic predation, parasitic behavior, and death mechanisms. The algorithm demonstrated competitive performance on numerical benchmarks and engineering optimization problems, including one path-optimization problem. Nevertheless, the original CCO was intended as a general-purpose optimizer, and has not been adequately specialized to support multi-objective path planning of mobile robots simultaneously considering path length, collision avoidance, and smoothness [13].

Path-planning techniques can generally be classified into classical search-based methods, sampling-based methods, artificial potential field methods, cell-decomposition methods, swarm-intelligence methods, and evolutionary optimization methods. However, the raw paths produced by these methods may contain redundant points, sharp turns, or segments that pass too close to obstacles. Therefore, additional smoothing or optimization is often required before such paths can be executed safely by a mobile robot. Recent studies have combined classical planners with curve-fitting and optimization techniques, such as Bezier curves, B-splines, and PSO, to improve path smoothness and obstacle avoidance [14–16].

The present study was triggered by these gaps, where an improved multi-objective Cuckoo-Catfish Optimizer for smooth and collision-free mobile robot path planning is developed. The proposed method improves population diversity, exploitation, and incorporates path quality objectives, improving the original CCO. It tries to create shorter, safer, and smoother paths while taking into account both global and local detail.

The novelty of this work is the original Cuckoo-Catfish Optimizer (CCO) that has been modified into an AICCO framework for smooth and collision-free navigation of mobile robots in path planning applications. Unlike the original CCO, which was developed as a general-purpose optimizer, the proposed AICCO integrates chaotic opposition-based initialization, nonlinear adaptive control, elite-guided search, strong elite preservation, memetic local refinement, and stagnation-based opposition repair specifically for waypoint-based path planning. These mechanisms are intended to increase initial waypoint diversity, better balance global corridor exploration with local path refinement, maintain higher-quality collision-free paths, and repair poor-quality and infeasible candidate paths near obstacles. Moreover, the proposed framework integrates path objectives with weighted-sum scalarization, adaptive obstacle expansion, and online dynamic replanning to tackle static, dynamic, and nonlinear/reactive obstacle environments. The conceptual overview of the proposed multi-objective mobile robot path-planning framework is shown in Figure 1.

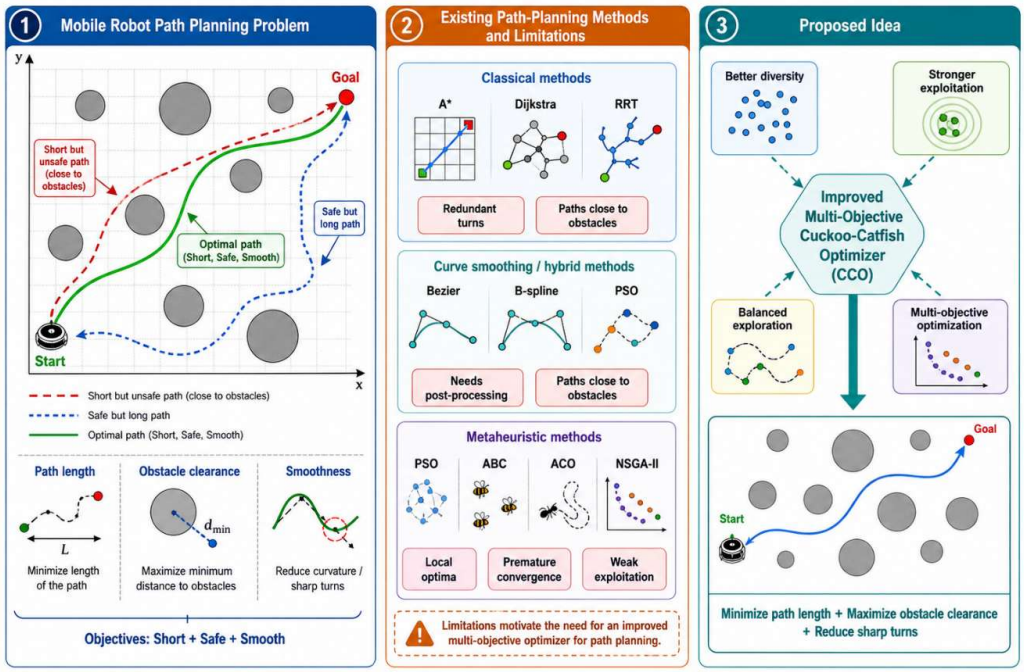


Figure 1. Conceptual overview of the proposed multi-objective mobile robot path-planning framework. The different colors are used only to visually distinguish the main stages/modules of the proposed framework.

2. Related Works

Recent studies have also begun to approach the problem of mobile robot path planning as a multi-objective optimization problem, as opposed to a single-objective shortest-path problem. Duan et al. proposed a better NSGA-II algorithm of mobile robot path planning where path length, path safety, and path smoothness were optimized concurrently. Their approach involved hybridization of initialization, problem-specific evolutionary operators, adaptive variable neighborhood local search, and hybrid global search to enhance both exploitation and exploration. This paper affirmed that path planning ought to take into consideration multiple conflicting goals other than the shortest path [1].

Another algorithm by Liu et al. is an NSGA-II-based algorithm that can be used to plan the paths of mobile robots to help solve issues like local optima and non-smooth paths. Their research also substantiates the relevance of Pareto-based evolutionary optimization in the balancing of various path-quality measures. Nonetheless, even solutions based on NSGA-II might still be highly dependent on the design of encoding schemes, crossover operators, mutation operators, and local search strategies which can diminish their overall adaptability to different environments [2].

Algorithms of swarm-intelligence have also been extensively developed to achieve smooth and safe navigation of the robots. Ye et al. suggested a hybrid algorithm of an artificial bee colony with genetic augmented exploration mechanisms for safe and smooth mobile robot path planning. Their technique developed path planning with three objectives; path length, path safety, and path smoothness. It presented redesigned path crossover, adaptive obstacle-aware mutation, dynamic elite selection, objective-guided optimization, and restart mechanisms to enhance the search performance. In another recent work, Ye et al. have proposed a higher artificial bee colony algorithm with a self-learning optimization mechanism to solve multi-objective path-planning problems with a focus on path length and safety using Pareto dominance and better population optimization strategies [3,4].

Other more recent research has been directed at enhancing safety and fluidity by integrating classical planners and optimization or control techniques. Guo et al. suggested

a safe path-planning approach, which relies on a better PSO and control barrier functions, wherein a safety velocity adjustment mechanism was applied to minimize local-minimum issues and enhance obstacle avoidance. Ahmad and Ab Wahab suggested a framework which is a combination of the Dijkstra algorithm and dynamical waypoint assignment, and optimization of piecewise cubic Bezier curves to enhance both safety and smoothness in complex settings. Jo and Choi further demonstrated that the RRT-generated paths were often in need of smoothing since the straight line segments in the path might not meet robot turning-radius specifications; hence, they used B-splines and PSO to come up with smoother, collision-free paths [14–16].

New smart and learning-based optimization techniques have also been proposed to be used in multi-objective path planning. Ran et al. came up with a multi-objective RIME algorithm, called DRL-MORIME, to be used by mobile robots to optimize multiple objectives simultaneously. Their approach maximized path length, smoothness, and safety, as well as applying adaptive operator selection and archive optimization to enhance the Pareto-front diversity and search performance. Their reported results included an improvement in the path length, total turning angle, and collision risk, which confirms the significance of adaptive multi-objective optimization in recent studies in robot navigation [17].

Recently, there has been advanced research in path planning that has indicated the necessity of combining global search, local obstacle avoidance, and path smoothing. For instance, smoothing-optimized A*-guided Dynamic Window Approach (DWA) methods have been proposed to achieve redundancy reduction in A*-based global paths and better local obstacle avoidance performance in DWA [18]. Other studies based on the DWA added obstacle-behavior prediction to the local planning to improve dynamic obstacle avoidance [19]. Model predictive control (MPC)-based trajectory planners have also been introduced to handle dynamic obstacles by considering prediction uncertainty and robot kinodynamic constraints [20]. In addition, some planners in recent years based on swarm-intelligence have formulated mobile robot path planning as a multi-objective problem of path length, path safety, and path smoothness, such as the hybrid artificial bee colony and enhanced artificial bee colony methods [3,4].

In comparison to these recent methods, the proposed AICCO is a population-based multi-objective planner which targets generating paths based on waypoints with an explicit trade-off between path length, obstacle clearance, and path smoothness. The A*-DWA and MPC-based approaches are primarily planner-control and local-trajectory approaches, while AICCO search is directly performed in waypoint solution space, and the complete candidate path can be evaluated by the proposed multi-objective fitness function. Furthermore, unlike other swarm-intelligence approaches, the CCO structure offers more than one way of searching, which can be matched to various path-planning tasks such as the widespread exploration of viable corridors, refinement around promising paths, and discarding of infeasible or poor-quality path candidates.

Despite the above measures that have made great strides, there are still a number of constraints. Classical planners are often in need of more smoothing and safety measures; some metaheuristic methods are prone to local optima and high levels of randomness; and many multi-objective evolutionary algorithms rely on problem-specific operators or post-processing. Furthermore, although the initial CCO has been proven to have a good general optimization ability, it has not yet been fully adapted to dedicated multi-objective mobile robot path planning. Thus, the paper will suggest a better CCO framework, which explicitly includes path length, obstacle clearance, and smoothness into the optimization process, with the aim of generating shorter, safer, and smoother paths free of collisions.

Limitations of past research may be summarized as the following:

1. The raw paths produced by classical graph-search planners may consist of redundant points, sharp turns, or hazardous obstacle proximity.
2. Numerous traditional planners continue to consider path length primarily, but safety and smoothness must also be taken into account to enable real robot navigation.
3. Certain metaheuristic algorithms are more flexible but can be characterized by high computational cost, local optima, or premature convergence.
4. There are multiple multi-objective evolutionary algorithms that rely heavily on problem-specific operators and can exhibit weak exploitation.
5. The original CCO is a powerful general optimizer, though it has not been appropriately customized to smooth and collision-free multi-objective mobile robot path planning.

3. Problem Statement and Preliminaries

Suppose that a mobile robot has an initial starting position SP and must move to a goal position GP in a two-dimensional workspace. The work area is filled with a number of fixed and moving barriers. The primary goal of the path-planning problem is to produce a short, smooth, safe, and collision-free optimal or near-optimal path.

In a static environment, only one complete path can be created, starting from the initial point and extending to the goal point. But in a dynamic world, a pre-calculated path might be unsafe due to the changing positions of moving obstacles over time. Hence, the path-planning problem is posed in this work as an online dynamic replanning problem. At every decision time t_k, the robot updates its current position as well as the states of the surrounding obstacles, replans a new local path to the goal, executes only a short movement step, and repeats the process until the goal is achieved.

In the case of the planned path, a sequence of waypoints is used to represent the planned path:

$$P = \{SP, p_1, p_2, \ldots, p_n, GP\}, \tag{1}$$

where each waypoint is defined as

$$p_i = (x_i, y_i), \quad i = 1, 2, \ldots, n. \tag{2}$$

For the online dynamic replanning case, the local path generated at the k-th replanning step is represented as:

$$P^{(k)} = \left\{P_R(t_k), p_1^{(k)}, p_2^{(k)}, \ldots, p_n^{(k)}, GP\right\}. \tag{3}$$

$P_R(t_k)$ is the current robot position at the decision time t_k, and $p_i^{(k)}$ is the i-th waypoint that was generated at the k-th replanning step.

The suggested path-planning model takes into account the following assumptions:

Assumption 1. *The obstacles are represented as circular shapes. The description of each obstacle is as follows: the center position and the radius.*

Assumption 2. *The mobile robot is assumed to be a physical body with radius r_{MR}. Increasing the size of the robot and its safety requirements makes collision checking easier. In such a way, the mobile robot can be considered as a point without putting oneself at risk of collision with obstacles.*

Assumption 3. *There can be both static and dynamic obstacles in the workspace. The obstacles that are dynamic change their positions with time according to some predefined linear, circular, nonlinear, or reactive motion models, whereas static obstacles are fixed during navigation.*

Assumption 4. *The mobile robot is assumed to be omnidirectional, possessing the capability of moving in any direction. Therefore, the primary focus of the path planning algorithm is on the quality of the path, obstacle avoidance, and the generation of smooth paths.*

Assumption 5. *In dynamic environments, the robot follows an online replanning strategy. Each time the decision time t k occurs, the new local start point is the current robot position, and a new path is generated between the current robot position and the goal. The robot is only allowed to travel a limited portion of the generated path before it must replan.*

3.1. Adaptive Obstacle Expansion

To ensure safe navigation, an adaptive radius is added to each obstacle to ensure safe navigation. In contrast to the traditional fixed expansion approach, the proposed approach takes into account the robot radius, safety margin, sensing uncertainty, and relative motion between the robot and the obstacle. Such expansion enables the robot to be considered a point, even with its physical size and the amount of safety clearance necessary.

The general extended radius of obstacle O_j is defined as:

$$r_{exp,j} = r_{obs,j} + r_{MR} + \delta_0 + \gamma\sigma + k_v \parallel v_{rel,j} \parallel, \tag{4}$$

where $r_{exp,j}$ is the expanded radius of obstacle O_j, $r_{obs,j}$ is the original obstacle radius, r_{MR} is the mobile robot radius, δ_0 is the basic safety margin, σ represents sensing or localization uncertainty, $v_{rel,j}$ is the relative velocity of the robot obstacle O_j, and γ and k_v are weighting coefficients.

To consider the relative velocity term in the case of static obstacles, since the position of the obstacle is not changing with time, then the relative velocity term is ignored. Thus, the increased radius is obtained:

$$r_{exp,j}^{static} = r_{obs,j} + r_{MR} + \delta_0 + \gamma\sigma. \tag{5}$$

In the case of dynamic obstacles, relative motion between the robot and the obstacle is taken into account. In this way, the extended radius is

$$r_{\mathrm{exp},j}^{\mathrm{dynamic}} = r_{obs,j} + r_{MR} + \delta_0 + \gamma\sigma + k_v\|v_R - v_j\|, \tag{6}$$

where v_R is the robot velocity and v_j is the velocity of the j-th dynamic obstacle.

The proposed online replanning framework examines the expansion of obstacles in a time-dependent manner. At a sampled point q of a candidate path, the obstacle is considered at the predicted arrival time of the robot at that point. Let s_q be the cumulative path distance of the current robot position $P_R(t_k)$ to the sampled point q. The forecasted arrival time is estimated as follows:

$$\tau_q = t_k + \frac{s_q}{v_R}, \tag{7}$$

where v_R is the speed of the robot. Thus, the dynamic expanded radius at the forecasted arrival time can be expressed as:

$$r_{exp,j}(\tau_q) = r_{obs,j} + r_{MR} + \delta_0 + \gamma\sigma + k_v\|v_R(\tau_q) - v_j(\tau_q)\|. \tag{8}$$

This adaptive expansion plan offers a safer boundary in reality. The faster the obstacle travels or the greater the uncertainty, the larger the safety boundary. Conversely, when the obstacle is not moving or has little risk, the expansion is smaller, which helps to eliminate

unnecessary detours. The proposed adaptive obstacle expansion concept is illustrated in Figure 2.

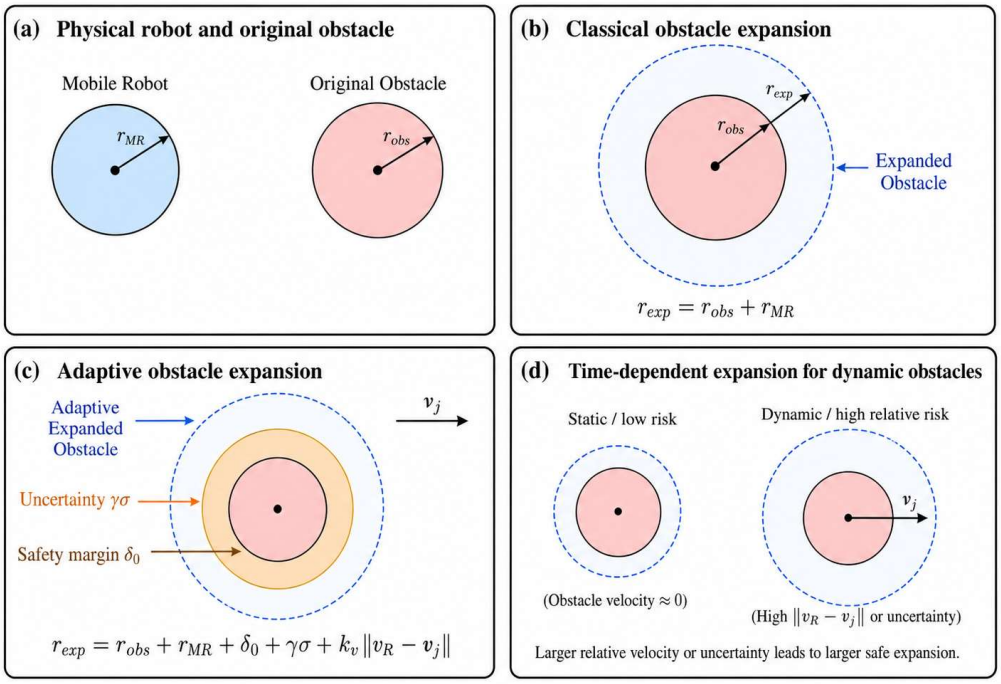


Figure 2. Adaptive obstacle expansion according to robot size, safety margin, uncertainty, and obstacle motion.

3.2. Dynamic Obstacle Motion Model

Dynamic obstacles are assumed to change their location at every time step. Obstacle movement is taken into account in two different ways: linear and circular.

3.2.1. Linear Movement

In linear movement, the obstacle moves with a velocity v_{obs} in a straight direction at the direction angle ϕ_{obs}. The obstacle position at the next time step is changed to:

$$x_{obs}(t + \Delta t) = x_{obs}(t) + v_{obs}\Delta t \cos(\phi_{obs}), \tag{9}$$

$$y_{obs}(t + \Delta t) = y_{obs}(t) + v_{obs}\Delta t \sin(\phi_{obs}), \tag{10}$$

where ϕ_{obs} is the direction angle of the obstacle and Δt is the sampling time.

Alternatively, the center of the obstacle at a time t can be written as

$$c_{obs}(t) = c_{obs}(0) + v_{obs}t \begin{bmatrix} \cos(\phi_{obs}) \\ \sin(\phi_{obs}) \end{bmatrix}. \tag{11}$$

3.2.2. Circular Movement

In circular movement, the obstacle moves around a fixed central point (x_c, y_c) in the circular trajectory with radius (r_c). The position of the obstacle is specified as:

$$x_{obs}(t) = x_c + r_c \cos(\theta(t)), \tag{12}$$

$$y_{obs}(t) = y_c + r_c \sin(\theta(t)). \tag{13}$$

As such, the angular position is updated to be

$$\theta(t + \Delta t) = \theta(t) + \omega \Delta t, \tag{14}$$

where ω is the angular velocity. The angle is between

$$0 \leq \theta < 2\pi.$$

The updated obstacle position is used for the adaptive obstacle expansion model and for obstacle detection and avoidance. In the proposed replanning framework in an online environment, these positions are also utilized during path evaluation by estimating the position of the obstacle at the expected arrival time τ_q of the robot at each sample point. The obstacle motion models include both linear and circular types, as illustrated in Figure 3.

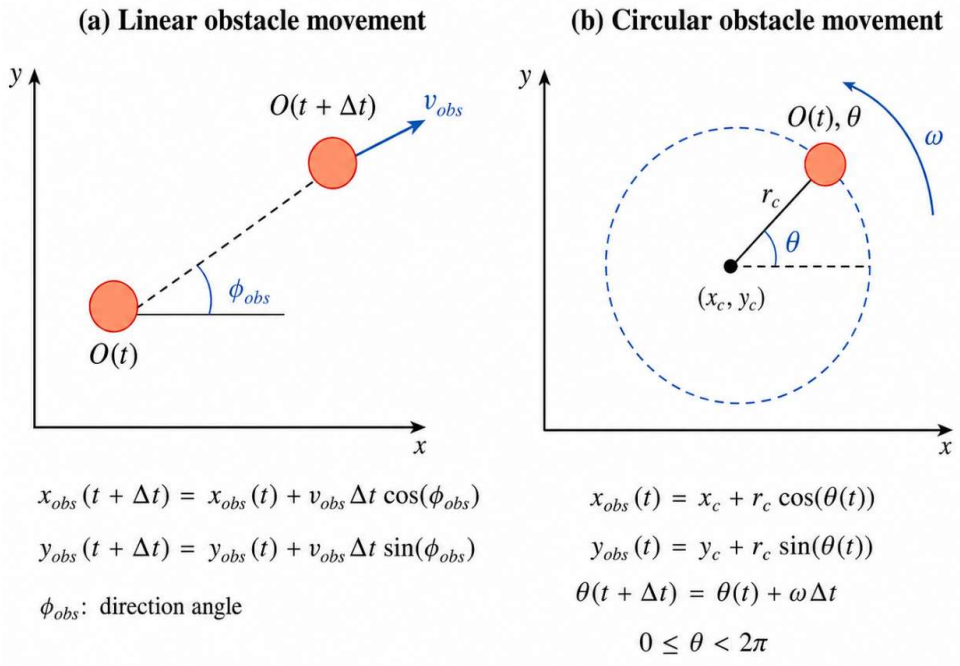


Figure 3. Motion models of dynamic obstacles: linear and circular movements. The orange circles represent the dynamic obstacle positions, while the dashed blue circle represents the circular trajectory followed by the obstacle in the circular motion model.

3.2.3. Nonlinear and Reactive Obstacle Motion

In addition to the linear and circular obstacle motion models, a nonlinear and reactive obstacle motion model is introduced to represent more complex dynamic environments. In this model, the obstacle velocity is not constant. Instead, it changes with time due to a nonlinear acceleration component. Moreover, the obstacle can react to the robot position when the robot enters a predefined interaction region.

Let the position of the j-th dynamic obstacle at time t be represented as:

$$\mathrm{c}_j(t) = \begin{bmatrix} x_j(t) \\ y_j(t) \end{bmatrix}, \tag{15}$$

and let its velocity be represented by

$$\mathrm{v}_j(t) = \begin{bmatrix} v_{x,j}(t) \\ v_{y,j}(t) \end{bmatrix}. \tag{16}$$

The obstacle position is updated at each sampling time Δt as:

$$\mathrm{c}_j(t+\Delta t) = \mathrm{c}_j(t) + \mathrm{v}_j(t)\Delta t, \tag{17}$$

where $\mathrm{c}_j(t)$ is the obstacle center position, $\mathrm{v}_j(t)$ is the obstacle velocity, and Δt is the sampling time.

The obstacle velocity is updated according to the nonlinear and reactive acceleration components as:

$$\mathrm{v}_j(t+\Delta t) = \mathrm{v}_j(t) + \left[\mathrm{a}_j^{nl}(t) + \mathrm{a}_j^{\mathrm{react}}(t)\right]\Delta t, \tag{18}$$

where $a_j^{nl}(t)$ is the nonlinear acceleration component and $a_j^{\mathrm{react}}(t)$ is the reactive acceleration component.

The nonlinear acceleration component is defined as:

$$\mathrm{a}_j^{nl}(t) = A_j \begin{bmatrix} \sin(\omega_j t + \phi_j) \\ \cos(\omega_j t + \phi_j) \end{bmatrix}, \tag{19}$$

where A_j is the acceleration amplitude, ω_j is the angular frequency, and ϕ_j is the phase angle. This component allows the obstacle to move along a nonlinear and time-varying trajectory.

The distance between the robot and the j-th obstacle is calculated as:

$$d_{R,j}(t) = \left\|\mathrm{c}_j(t) - \mathrm{P}_R(t)\right\|, \tag{20}$$

where $P_R(t)$ is the robot position at time t.

The reactive acceleration component is activated only when the robot is closer than a predefined interaction distance (d_{react}). It is defined as:

$$\mathrm{a}_j^{\mathrm{react}}(t) = \begin{cases} k_r \frac{\mathrm{c}_j(t) - \mathrm{P}_R(t)}{d_{R,j}(t)+\varepsilon}, & d_{R,j}(t) < d_{\mathrm{react}}, \\ 0, & d_{R,j}(t) \geq d_{\mathrm{react}} \end{cases} \tag{21}$$

where k_r is the reactive acceleration gain and ε is a small positive constant used to avoid division by zero.

This model is more challenging than purely linear and circular obstacle motion models because the obstacle trajectory changes with time and may also depend on the robot position. Therefore, it provides a more realistic test for evaluating the robustness of the proposed online replanning strategy in complex dynamic environments.

3.3. Objective Function Formulation

The improved multi-objective Cuckoo-Catfish Optimizer is proposed to generate a path that satisfies three main objectives: minimum path length, improved path smoothness, and maximum safety. These goals are merged into a weighted cost function, which is minimized by the optimizer.

3.3.1. Path Length Objective

The first goal is to find the shortest path possible from the robot's starting position to the goal's position. The path length objective for the local path generated at replanning step k is:

$$f_1\left(P^{(k)}\right) = \sum_{i=0}^{n} d\left(p_i^{(k)}, p_{i+1}^{(k)}\right), \tag{22}$$

where

$$d\left(p_i^{(k)}, p_{i+1}^{(k)}\right) = \sqrt{\left(x_{i+1}^{(k)} - x_i^{(k)}\right)^2 + \left(y_{i+1}^{(k)} - y_i^{(k)}\right)^2} \tag{23}$$

with:

$$p_0^{(k)} = P_R(t_k), p_{n+1}^{(k)} = GP. \tag{24}$$

A smaller value of f_1 indicates a shorter path.

3.3.2. Path Smoothness Objective

The second objective is to minimize sharp turns along the path. The smoothness goal is based on the angle between successive path segments. Let us define the heading angle of the i-th segment as $\theta_i^{(k)}$:

$$\theta_i^{(k)} = \text{atan2}\left(y_{i+1}^{(k)} - y_i^{(k)}, x_{i+1}^{(k)} - x_i^{(k)}\right). \tag{25}$$

The smoothness objective is formulated as:

$$f_2\left(P^{(k)}\right) = \sum_{i=1}^{n} \left|\theta_i^{(k)} - \theta_{i-1}^{(k)}\right|. \tag{26}$$

A lower f_2 makes the generated path smoother with less sharp turns, which is convenient for robot motion. The formulation of the overall objective function is summarized in Figure 4.

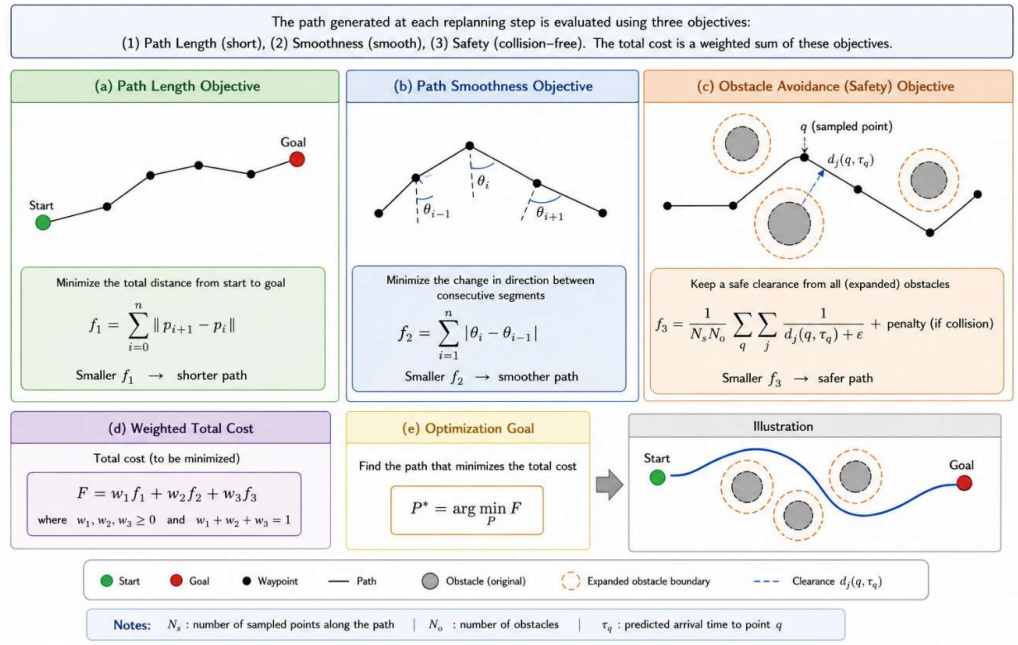


Figure 4. Objective function formulation for the proposed multi-objective path-planning problem. The symbol P* denotes the optimal path that minimizes the total weighted cost function.

3.3.3. Obstacle Avoidance Objective

The third objective is to keep the robot's path away from the expanded obstacles. The difference between waypoint-only collision checking and the proposed approach is that the proposed approach checks obstacle clearance with sampled points on each path segment. This ensures more dependable collision detection, particularly if a path segment is in the immediate vicinity of an obstacle between two waypoints.

The arrival time at each sampled point q on the path is predicted as:

$$\tau_q = t_k + \frac{s_q}{v_R}, \tag{27}$$

where s_q is the total distance that a robot has traveled from its current position $P_R(t_k)$ to the sampled point q, and v_R is the robot velocity.

The clearance of sampled point q to obstacle O_j is:

$$D_j(q, \tau_q) = \|q - c_j(\tau_q)\|, \tag{28}$$

where $c_j(\tau_q)$ is the predicted center of obstacle O_j at time τ_q. Then, the safe clearance is computed to be:

$$d_j(q, \tau_q) = D_j(q, \tau_q) - r_{\text{exp},j}(\tau_q). \tag{29}$$

A sampled point is deemed to be "unsafe" when:

$$d_j(q, \tau_q) \leq 0.$$

A risk term is incorporated in the formulation of the safety objective, which increases as the path gets closer to the expanded obstacle boundary:

$$f_{\text{risk}} = \frac{1}{N_s N_o} \sum_{q=1}^{N_s} \sum_{j=1}^{N_o} \frac{1}{\max(d_j(q, \tau_q), \varepsilon)}, \tag{30}$$

where N_s is the number of sampled path points, N_o is the number of obstacles, and ε is a small positive number to prevent division by zero errors. If any sampled point is in a large obstacle, then a large collision penalty is applied:

$$f_{\text{penalty}} = \sum_{q=1}^{N_s} \sum_{j=1}^{N_o} \Phi(q, O_j), \tag{31}$$

where

$$\Phi(q, O_j) = \begin{cases} \lambda_c (1 + |d_j(q, \tau_q)|)^2, & d_j(q, \tau_q) \leq 0 \\ 0, & d_j(q, \tau_q) > 0 \end{cases}.$$

λ_c is a positive constant, large enough. Thus, the obstacle avoidance goal is:

$$f_3\left(P^{(k)}\right) = f_{\text{risk}} + f_{\text{penalty}}. \tag{32}$$

The smaller the value of f_3, the safer the path, with higher clearance of obstacles.

3.3.4. Scalarized Multi-Objective Fitness Function

The total objective function is represented in the form of a weighted sum:

$$F\left(P^{(k)}\right) = w_1 f_1\left(P^{(k)}\right) + w_2 f_2\left(P^{(k)}\right) + w_3 f_3\left(P^{(k)}\right). \tag{33}$$

It is important to point out that the suggested formulation is a weighted-sum scalarization of several objectives pertaining to path planning. Hence, the optimizer minimizes a single scalar fitness function, which is a combination of the path length, path smoothness, and an obstacle avoidance function. In this form, the proposed approach is not a multi-objective Pareto path planner, but a weighted multi-objective path planner. The reason for this choice was that weighted-sum scalarization is simple, not too heavy to compute, and is well-suited for repeated online replanning, in which only one of a set of candidate paths is chosen at each decision step.

For the main experiments, weighting coefficients (w_1), (w_2), and (w_3) are determined and kept constant for a stable and fair evaluation environment of all compared algorithms.

These weights indicate the relative dominance of the path length, path smoothness, and obstacle avoidance, respectively. The chosen baseline setting provides a balance between the three objectives, with a slight preference for path length. The proposed formulation is not, however, restricted to a fixed set of weights. The weights can be changed depending on the application's navigation priority. As an illustration, the higher the (w_3) is, the more important obstacle clearance and safety will be, the higher the (w_2) is, the more important a smoother path will be, and the higher the (w_1) is, the more important a shorter path will be.

In order to assess the impact of the weighting coefficients, a weight sensitivity analysis was included in Section Weight Sensitivity Analysis. The sensitivity analysis involves the four different configurations—balanced, length-priority, smoothness-priority, and safety-priority—with identical static obstacle maps. This analysis sheds light on the different weight settings and their impact on path length, smoothness, clearance, and collision avoidance.

In this, w_1, w_2, and w_3 are path length, smoothness, and safety weighting coefficients, respectively. These weights satisfy the following:

$$w_1 + w_2 + w_3 = 1. \tag{34}$$

All the objective terms are normalized/scaled before being aggregated in the weighted fitness function so as to enhance the generalizability of the map configurations. Also, sensitivity analysis on the weight is performed in the experimental part to investigate the performance of the proposed AICCO with different navigation preferences.

In this work, the total path-planning cost is directly minimized by the optimizer:

$$P_{best}^{(k)} = \arg\min_{P^{(k)}} F\left(P^{(k)}\right). \tag{35}$$

The optimum route is therefore the one that has minimal distance, minimal velocity change, and no collisions.

4. Obstacle Detection and Avoidance (ODA)

In the proposed framework, adaptive obstacle detection and avoidance strategy has been proposed to enhance robot navigation in static and dynamic environments. The proposed ODA method integrates adaptive obstacle expansion, sector-based sensing, risk evaluation, safe-gap selection, and online dynamic replanning.

At every decision time t_k, the robot updates its belief of the current location and the locations of the dynamic obstacles with the motion models presented in Section 3.2. Next, each obstacle is expanded using the adaptive obstacle expansion model. Then, the sensing area of the mobile robot is subdivided into virtual areas. The sectors are the possible movement directions.

The proposed method differs from conventional binary obstacle detection techniques by assigning a risk value to each sector. The obstacle distance, expanded obstacle radius, sensing range, uncertainty, and relative obstacle motion are factors that affect the risk value. The probability of value of a sector S_i is determined by:

$$R_i = \alpha_1\left(1 - \frac{d_i}{SR}\right) + \alpha_2\left(\frac{r_{\exp,i}}{SR}\right) + \alpha_3\left(\frac{v_{rel,i}}{v_{\max}}\right), \tag{36}$$

where d_i is the nearest distance between the robot and the expanded obstacle detected in sector S_i, SR is the sensing range, $r_{\exp,i}$ is the expanded obstacle radius, $v_{\text{rel},\,i}$ is the relative

velocity between the robot and the detected obstacle, $v_{\max}$ is the maximum expected obstacle velocity, and α_1, α_2, and α_3 are weighting coefficients.

A free-sector vector is created in the following manner:

$$V_s(i) = \begin{cases} 1, & R_i < R_{th} \\ 0, & R_i \geq R_{th} \end{cases}, \tag{37}$$

when R_{th} is a predetermined safety margin. Here, $V_s(i) = 1$ means that the sector i is safe, and $V_s(i) = 0$ means the sector i is blocked or risky.

Back-to-back safe sectors are combined together to create candidate safe gaps:

$$G = \{G_1, G_2, \ldots, G_m\},$$

where m is the number of gaps available that are deemed safe.

Every candidate gap is assessed based on its angular size, the average risk, the direction of the goal, and the smoothness of the gap in relation to the last heading direction. The gap G_l of the candidate is calculated thus:

$$J_l = \beta_1 |\theta_l - \theta_g| + \beta_2 \frac{1}{W_l + \varepsilon} + \beta_3 \bar{R}_l + \beta_4 |\theta_l - \theta_{\text{prev}}|, \tag{38}$$

where θ_l is the center angle of gap G_l, θ_g is the angle between the robot and the goal point, W_l is the angular width of the gap, $\bar{R}_l$ is the average risk inside the gap, θ_{prev} is the previous robot heading, and β_1, β_2, β_3, and β_4 are weighting coefficients.

The optimum safe gap chosen is:

$$G^* = \arg\min_{G_l} J_l. \tag{39}$$

Once the most favorable gap is chosen, a provisional local target is created in the chosen gap in the central direction:

$$P_{\text{local}} = P_R(t_k) + L_d \begin{bmatrix} \cos(\theta_{G^*}) \\ \sin(\theta_{G^*}) \end{bmatrix}, \tag{40}$$

where L_d is the look-ahead distance, and $P_R(t_k)$ is the position of the robot at the current time.

This local target or chosen safe direction is not the final desired plane in the proposed online replanning framework. Rather, it helps direct the optimizer away from unsafe or high-risk directions to more secure areas during replanning. The proposed AICCO, for every replanning step, computes a new local path from the current robot location to the goal, taking into account the new positions of obstacles and adaptive safety boundaries.

The online replanning process is formulated as:

$$P^{(k)} = AICCO(P_R(t_k), GP, \mathcal{O}(t_k)), \tag{41}$$

where

$$\mathcal{O}(t_k) = \{O_1(t_k), O_2(t_k), \ldots, O_{N_o}(t_k)\}.$$

Once the local path $P^{(k)}$ is generated, the robot performs a small motion step only:

$$\Delta s = v_R \Delta t, \tag{42}$$

and adjusts its location to be

$$P_R(t_{k+1}) = P_R(t_k) + \Delta s \hat{u}^{(k)}, \tag{43}$$

where $\hat{u}^{(k)}$ is the unit direction along the first segment of the replanned path.

If it is impossible to leave a safe space, the robot switches to emergency avoidance mode. In this situation, the robot slows down or decreases the length of the step performed and approaches the sector with the lowest risk value:

$$S^* = \arg\min_i R_i. \tag{44}$$

This mechanism avoids collision and keeps the robot going until it finds a safe gap that is feasible.

The proposed obstacle detection and avoidance approach (ODA) is better than the traditional approach which relies on only binary obstacle information. Rather, it takes into account obstacle distance, robot size, the safety margin, uncertainty, and dynamic obstacle motion. In addition, the use of ODA and online replanning allows the robot to not just follow a predetermined path that is calculated only at the start of the mission. Rather, it constantly revises its path depending on the state of the robot and the estimated movement of other robots. This makes the proposed method more suitable to a realistic dynamic environment. Figure 5 illustrates the proposed ODA framework comprising sector-based risk evaluation, safe-gap selection, and online replanning.

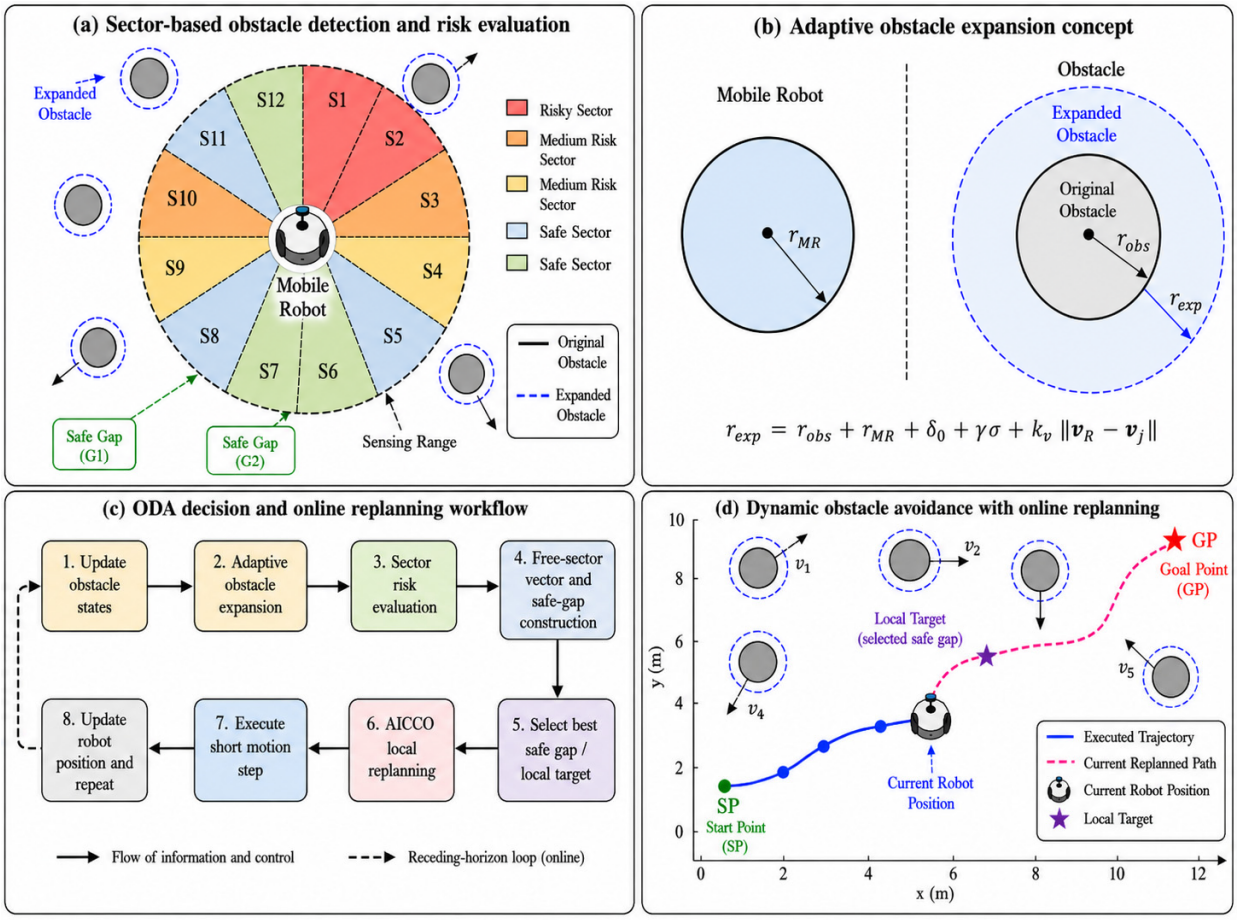


Figure 5. Proposed adaptive risk-sector obstacle detection and avoidance framework.

5. Proposed Optimization Algorithm

5.1. Original Cuckoo-Catfish Optimizer

Recently, Wang et al. [13] proposed a swarm-intelligence-based metaheuristic algorithm, namely the Cuckoo-Catfish Optimizer (CCO). The algorithm is based on the biological characteristics of the cuckoo catfish: searching, predation, and parasitic and survival mechanisms in interactions with cichlids. These are natural behaviors that are mathematically modeled to lead a group of candidate solutions towards optimal or near-optimal parts of the search space. The initial step in the original CCO is to randomly create a population of candidate solutions within a specified lower and upper limit. Every individual is a potential solution to the optimization problem, and the quality of the individual is judged by a fitness function. In the iterative process, individuals' positions are updated with the help of a number of search strategies that mimic the behavior of cuckoo catfish in nature. The original CCO mainly consists of three search stages: exploration, transition, and exploitation. During the initial phase, a compressed space and surround-search algorithm is used to improve the global exploration capacity and search the different regions of the solution space. Such mechanisms are used to diversify the population and minimize the chances of premature convergence. At the intermediate stage, a transition strategy is employed: half of the population move towards promising regions while the remainder of the population keep searching away from their best-found so far. In the latter phase, chaotic predation is applied to intensify exploitation about the best solution and boost the convergence accuracy. Furthermore, to increase the population-updating process, the original CCO incorporates parasitism and death mechanisms. Poor or failed individuals can be replaced by new ones, which helps to avoid poor regions of the search space and maintain the search diversity. These mechanisms enable the CCO to balance global search and local refinement. The original CCO has been successfully applied to numerical optimization problems, engineering design problems, photovoltaic parameter estimation cases and a path-optimization case, but it was mainly developed as a general-purpose optimization algorithm. Thus, it does not take into account the special needs of mobile robot path planning, including the representation of waypoints, the smoothness of the path, the clearance of obstacles, and the avoidance of collisions. In view of this, the original CCO is modified and improved so that it is more suitable to be used in mobile robot path planning with multiple objectives.

5.2. Proposed AICCO: Memetic Elite-Refined Adaptive Improved Cuckoo-Catfish Optimizer

The original Cuckoo-Catfish Optimizer (CCO) performs well in optimization, but it has some drawbacks when applied directly to mobile robot path planning, such as the random initial distribution, loss of elite solutions, premature stagnation, and the lack of local refinement around good solutions. Hence, this work aims to develop an Adaptive Improved Cuckoo-Catfish Optimizer (AICCO) to improve the optimization accuracy of CCO without altering its basic search structure.

The AICCO proposed is not a full withdrawal from the original CCO mechanism. Rather, it keeps the original CCO search behavior with several lightweight but effective improvement strategies. These strategies are chaotic opposition-based initialization, nonlinear adaptive control, elite-guided search enhancement, strong elite preservation, memetic local refinement, and stagnation-based opposition repair.

These changes aim at increasing the diversity of the population in the initial phase of the search, enhancing the tendency to exploit good solutions, and preventing premature convergence.

The choice of CCO as the basic optimizer is due to the similarity between its original search method and the requirements of path planning for mobile robots. For the waypoint-based path planner, the optimizer needs to search for various feasible corridors in the

workspace, select and optimize some promising waypoint sets, and preserve safe and smooth paths without falling into local optima around obstacles. These requirements are conveniently organized into the original CCO. Compressed space search allows the search area to be decreased around potential path corridors. The surround-search mechanism allows exploration around candidate waypoint configurations. Transition behavior helps make a gradual transition from global exploration to local exploitation. The chaotic predation mechanism enhances local refinement for better solutions. Furthermore, the parasitism and death mechanisms eliminate weak or infeasible path candidates like paths with poor clearance or collision penalties.

Table 1 provides more detailed information on the mapping of each improvement mechanism to its contribution to the path-planning problem.

Table 1. Path-planning-specific role of the proposed AICCO mechanisms.

AICCO Mechanism	General Optimization Role	Specific Role in Mobile Robot Path Planning
Chaotic opposition-based initialization	Improves initial population diversity	Generates diverse initial waypoint configurations and increases the probability of starting from feasible path corridors
Nonlinear adaptive control	Balances exploration and exploitation	Encourages wide workspace exploration in early iterations and supports fine path refinement in later iterations
Elite-guided search enhancement	Guides the population toward promising regions	Directs candidate paths toward safer and shorter waypoint configurations discovered by elite solutions
Strong elite preservation	Prevents loss of the best solution	Preserves the best collision-free path found during the search process
Memetic local refinement	Improves local exploitation accuracy	Refines waypoint positions to reduce unnecessary turning, improve clearance, and enhance path smoothness
Stagnation-based opposition repair	Escapes local stagnation	Repairs path candidates trapped near obstacles or poor local regions by generating alternative waypoint configurations

While some enhancement strategies presented in AICCO like chaotic initialization, elite preservation, adaptive control, greedy selection, memetic refinement, and opposition-based repair are also observed in improved PSO, GWO, ACO, ABC, and other swarms, the originality of the proposed method is based on combining these strategies in a CCO-specific and path-planning-oriented manner. A PSO-based planner primarily provides velocity-position updates depending on personal and global best solutions. In GWO-based planners, there are leader wolves to direct the search. For ACO-based planners, the search is primarily based on the construction of paths using pheromones. In ABC-based planners, searching is carried out in the employed bee phase, onlooker bee phase, and scout bee phase. The proposed AICCO, on the other hand, maintains the original CCO search structure such as compressed-space search, surround search, transition behavior, chaotic predation, parasitic behavior, and death mechanisms and uses these mechanisms in waypoint-based mobile robot path planning. The novelty of AICCO is hence not the individual usage of common enhancement operators but rather their systematic combination in the CCO framework, which enables waypoint diversity, collision-free path preservation, local smoothness refinement, and stagnation repair near obstacles.

Now, let X_i^t denote the position of the i-th individual at iteration t, $i = 1, 2, \ldots, N_p$ and N_p denote the population size. $T_{\max}$ is the maximum number of iterations. Each person is one possible solution. The problem of robot path-planning is a set of complete paths of

intermediate waypoints between the SP and the GP, with each one representing a solution. Hence, the best-so-far solution is calculated as follows:

$$X_{\text{best}}^{t} = \arg \min_{i=1,\ldots,N_p} F\left(X_i^t\right), \tag{45}$$

where $F\left(X_i^t\right)$ is the value of the objective function of the i-th solution.

The mean solution is computed from the best N_e individuals as:

$$X_{\text{elite}}^{t} = \frac{1}{N_e} \sum_{j=1}^{N_e} X_{j,\text{ best}}^{t}, \tag{46}$$

where $X_{j,\text{ best}}^{t}$ is the j-th best solution for the present population.

5.2.1. Chaotic Opposition-Based Initialization

All individuals of the population are randomly assigned to values between the lower and upper limits of the search space in the original CCO. However, the random initialization can result in the poor distribution of individuals, particularly in high-dimensional or constrained search spaces. So as to increase the diversity of the initial population, a chaotic sequence is generated using the logistic map in the proposed AICCO:

$$z_{i,d}^{t+1} = \mu z_{i,d}^{t}\left(1 - z_{i,d}^{t}\right), \tag{47}$$

where $z_{i,d}^{t} \in (0,1)$, d is the dimension index, and $\mu = 4$. The first solution is then created as:

$$X_{i,d}^{0} = LB_d + z_{i,d}^{0}(UB_d - LB_d), \tag{48}$$

where LB_d is the lower bound for the d-th decision variable, and UB_d is the upper bound for it.

The opposite solution is calculated as:

$$\bar{X}_{i,d}^{0} = LB_d + UB_d - X_{i,d}^{0}. \tag{49}$$

Both X_i^0 and $\bar{X}_i^0$ are calculated, and the better one is chosen:

$$X_i^0 = \begin{cases} X_i^0, & F\left(X_i^0\right) < F\left(\bar{X}_i^0\right) \\ \bar{X}_i^0, & \text{otherwise} \end{cases}. \tag{50}$$

This approach can improve the likelihood of beginning the optimization process in good areas and limit bad initial population distribution.

5.2.2. Nonlinear Adaptive Control Factor

The original CCO decreases the control factor to control the switch from exploration to exploitation. In the proposed AICCO, the transition is achieved with a nonlinear adaptive control factor, which results in a smoother, more flexible transition.

$$C(t) = C_{\min} + (C_{\max} - C_{\min})\left(1 - \frac{t}{T_{\max}}\right)^{\kappa}, \tag{51}$$

where $C_{\max}$ and $C_{\min}$ are the maximum and minimum values of the control factor, respectively, and κ is a nonlinear control parameter.

In early iterations, $C(t)$ is high, and this promotes world exploration. The more iterations, the lower the value of $C(t)$, and the more it is willing to exploit and enact local refinement. Further, the shrink factor that was inherited from the original CCO is used:

$$T(t) = \left(1 - \sin\left(\frac{\pi}{2}\frac{t}{T_{\max}}\right)\right)^{\frac{t}{T_{\max}}}. \tag{52}$$

Additionally, the adaptive surrounding radius factor is defined as:

$$w(t) = 1 - \frac{e^{\frac{t}{T_{\max}}} - 1}{e - 1}. \tag{53}$$

This is because of the adaptive factors that allow this algorithm to slowly shift from a wide range search to a finer local search.

5.2.3. Elite-Guided CCO Search Enhancement

The proposed AICCO maintains the search operators of the original CCO, such as compressed-space search, surround search, transition behavior, and chaotic predation. Nonetheless, there is added a correction term which is guided by the elite to enhance the exploitation ability of the algorithm.

Let $\mathcal{U}_{CCO}(X_i^t)$ be the candidate solution obtained with one of the original CCO search operators. The improved elite-guided update is defined as:

$$X_i^{\text{new}} = \mathcal{U}_{CCO}(X_i^t) + \eta C(t) r\left(X_{\text{elite}}^t - X_i^t\right), \tag{54}$$

where η is a control coefficient, r is a random number in the range [0, 1], and X_{elite}^t is the mean position of the elite solutions.

This change enables the original CCO search behavior to continue while directing the population to high-quality areas. Thus, the search process can take advantage of the exploration ability of the original CCO and the exploitation ability of elite guidance. The improved form of the equation for the compressed space search is:

$$X_i^{\text{new}} = X_i^t + Z|rd|\left(X_{\text{best}}^t + \frac{1}{2}X_{r1}^t - X_{r2}^t\right) + \frac{r^3}{2}\left(X_{r3}^t - X_{r4}^t\right) + \eta_1 C(t) r\left(X_{\text{elite}}^t - X_i^t\right), \tag{55}$$

where X_{r1}^t, X_{r2}^t, X_{r3}^t, and X_{r4}^t are randomly chosen individuals, Z is a binary random vector, and $|rd|$ is normally distributed random factor. The enhanced update for the surround-search stage is:

$$X_i^{\text{new}} = X_{\text{sur}}^t + \eta_2 C(t) r\left(X_{\text{best}}^t - X_i^t\right), \tag{56}$$

where X_{sur}^t is the result from the original surround-search spiral operator or spherical operator.

For the chaotic predation stage, the better update is provided by:

$$X_i^{\text{new}} = X_{\text{pre}}^t + \eta_3 (1 - C(t)) r\left(X_{\text{elite}}^t - X_i^t\right), \tag{57}$$

where X_{pre}^t is the solution that is produced by the original chaotic predation operator. The $(1 - C(t))$ term boosts the role of elite exploitation in subsequent rounds.

5.2.4. Greedy Acceptance Rule

The proposed AICCO adopts a greedy acceptance rule to reduce the amount of unnecessary randomness and maintain convergence stability. Only a new solution that is better than the current solution is accepted.

$$X_i^{t+1} = \begin{cases} X_i^{\text{new}}, & F\left(X_i^{\text{new}}\right) < F\left(X_i^t\right) \\ X_i^t, & \text{otherwise} \end{cases}. \tag{58}$$

This rule enhances the exploitation accuracy and avoids the algorithm wandering away from potential areas because of too many random accepts.

5.2.5. Strong Elite Preservation

In the search operation, high-quality solutions can be lost because of stochastic update operations. Hence, a robust elite preservation mechanism is put in place. At the end of each generation, the worst individual in population is replaced with the best solution ever found.

$$X_{\text{worst}}^{t+1} = X_{\text{best}}^t, \tag{59}$$

where

$$X_{\text{worst}}^t = \arg\max_{i=1,\ldots,N_p} F\left(X_i^t\right).$$

This mechanism allows the best solution in each iteration to always be maintained. This leads to a more stable convergence behavior and to a lower risk of losing high-quality solutions.

5.2.6. Memetic Local Refinement

In order to make the exploitation capability of the original CCO more powerful, a local refinement stage based on memetics is used around the best solution. This is performed in an alternate manner during the optimization process. The search step size is adaptively decreased as iterations go on in the local search:

$$\rho(t) = \rho_{\min} + (\rho_{\max} - \rho_{\min})\left(1 - \frac{t}{T_{\max}}\right), \tag{60}$$

with the maximum and minimum local search step size at this location being $\rho_{\max}$ and $\rho_{\min}$. A random local candidate around the best solution is generated as:

$$X_{\text{trial}} = X_{\text{best}}^t + \rho(t) R \odot (UB - LB), \tag{61}$$

where R is a random vector drawn from a normal distribution, and $\odot$ is an element-wise multiplication. The solution generated is accepted if it is better than the current best solution:

$$X_{\text{best}}^t = \begin{cases} X_{\text{trial}}, & F(X_{\text{trial}}) < F\left(X_{\text{best}}^t\right) \\ X_{\text{best}}^t, & \text{otherwise} \end{cases}. \tag{62}$$

Also, each dimension of the best solution is improved by coordinate-based refinement. Two candidate solutions are generated for the d-th dimension:

$$\begin{aligned} X_d^+ &= X_{best}^t + s_d e_d \\ X_d^- &= X_{best}^t - s_d e_d \end{aligned}, \tag{63}$$

where e_d is a unit vector in the d-th dimension, and s_d is the local step size in that dimension. The "best" one of the three X^t_{best}, X^+_d, and X^-_d is chosen:

$$X^t_{\text{best}} = \arg \min_{X \in \{X^t_{\text{best}}, X^+_d, X^-_d\}} F(X). \tag{64}$$

Memetic refinement enhances the local search accuracy and aids the algorithm in reaching more precise solutions, particularly in the later stages of the optimization process.

5.2.7. Stagnation-Based Opposition Repair

If the individual cannot be improved for a certain number of iterations, then it is said to be stagnant. The proposed AICCO has a stagnation-based opposition repair strategy besides its random replacement strategy. The opposition solution is generated as:

$$X^{opp}_i = LB + UB - X^t_i + \xi R \odot (UB - LB), \tag{65}$$

where ξ is a small perturbation coefficient and R is a normally distributed random vector. A second solution is produced close to the best solution:

$$X^{near}_i = X^t_{best} + \lambda C(t) R \odot (UB - LB), \tag{66}$$

where λ is a small control coefficient.

The solution is repaired, and the repaired solution is selected as:

$$X^{t+1}_i = \arg \min_{X \in \{X^t_i, X^{opp}_i, X^{\text{near}}_i\}} F(X). \tag{67}$$

This is to prevent the algorithm from getting stuck in a rut, but not from too much random regeneration.

5.2.8. Boundary Handling

The space of searched solutions is bounded after every change in the solution. If any decision variable falls outside of its range, it is repaired as:

$$X^{\text{new}}_{i,d} = \min\left(UB_d, \max\left(LB_d, X^{\text{new}}_{i,d}\right)\right). \tag{68}$$

This prevents any possible solution from being outside the searchable space.

5.2.9. Final AICCO Update Procedure

The following is a summary of the main update procedure of the proposed AICCO:

$$X^{t+1}_i = \mathcal{A}\left(\mathcal{U}_{CCO}(X^t_i), X^t_{\text{best}}, X^t_{\text{elite}}, C(t), T(t), w(t)\right), \tag{69}$$

where $\mathcal{A}(\cdot)$ is the proposed adaptive improvement process comprising elite guidance, greedy acceptance, elite preservation, memetic local refinement, and stagnation-based repair.

The end best solution is achieved as:

$$t^* = \arg \min_{t=1,\ldots,T_{\max}} F\left(X^t_{\text{best}}\right), X_{\text{best}} = X^{t^*}_{\text{best}}. \tag{70}$$

In the field of mobile robot path planning, the final X_{best} is translated to a list of waypoints which are considered to be the most optimal or near-optimal paths from the start point to the goal point. The proposed AICCO retains the core functionality of the original CCO, with the addition of stable and effective enhancement mechanisms. Chaotic opposition-based initialization enhances the diversity of the initial population. Nonlinear

adaptive control provides a smooth transition from exploration to exploitation. Convergence stability is improved by elite-guided search and good preservation of the elite. Memetic local refinement enhances the ability to exploit around the best solution, while stagnation-based opposition repair helps the robot to escape from poor search regions. These mechanisms make the proposed AICCO more effective at optimization problems and more appropriate to generate short, smooth, and collision-free paths in mobile robot path planning.

5.3. Pseudocode of the Proposed AICCO

The main steps of the proposed Adaptive Improved Cuckoo-Catfish Optimizer (AICCO) are summarized in Algorithm 1. The proposed AICCO preserves the original CCO search structure while enhancing it using chaotic opposition-based initialization, nonlinear adaptive control, elite-guided search enhancement, strong elite preservation, memetic local refinement, and stagnation-based opposition repair.

Algorithm 1. Pseudocode of the Proposed AICCO

Input: Population size Np, maximum number of iterations Tmax, lower bound LB, upper bound UB, elite size Ne, and objective function F(.)
Output: Best solution Xbest and best fitness value Fbest.
Initialize AICCO parameters and generate the initial population using chaotic opposition-based initialization.
Evaluate all individuals and determine the initial best solution Xbest.
for t = 1 to Tmax do
    Update the nonlinear adaptive control factors and compute the elite mean solution Xelite.
    for each individual Xi do
        Generate a new candidate solution using the original CCO search operators enhanced by elite guidance.
        Apply boundary handling and evaluate the new candidate solution.
        Apply greedy acceptance to update Xi and its stagnation counter.
        If Xi is stagnant, apply opposition-based and near-best repair, then keep the best repaired solution.
    end for
    Apply memetic local refinement around Xbest.
    Preserve the best solution by replacing the worst individual with Xbest.
    Update Xbest and store the convergence value.
end for
Decode Xbest into the final waypoint-based path.
return Xbest and Fbest.

Algorithm 1 presents the compact procedure of the proposed AICCO. The algorithm preserves the original CCO search operators while improving the search process through chaotic opposition-based initialization, nonlinear adaptive control, elite-guided enhancement, greedy acceptance, stagnation repair, memetic local refinement, and strong elite preservation. These mechanisms improve the balance between exploration and exploitation and support the generation of short, smooth, and collision-free paths. The flowchart of the proposed AICCO algorithm is shown in Figure 6.

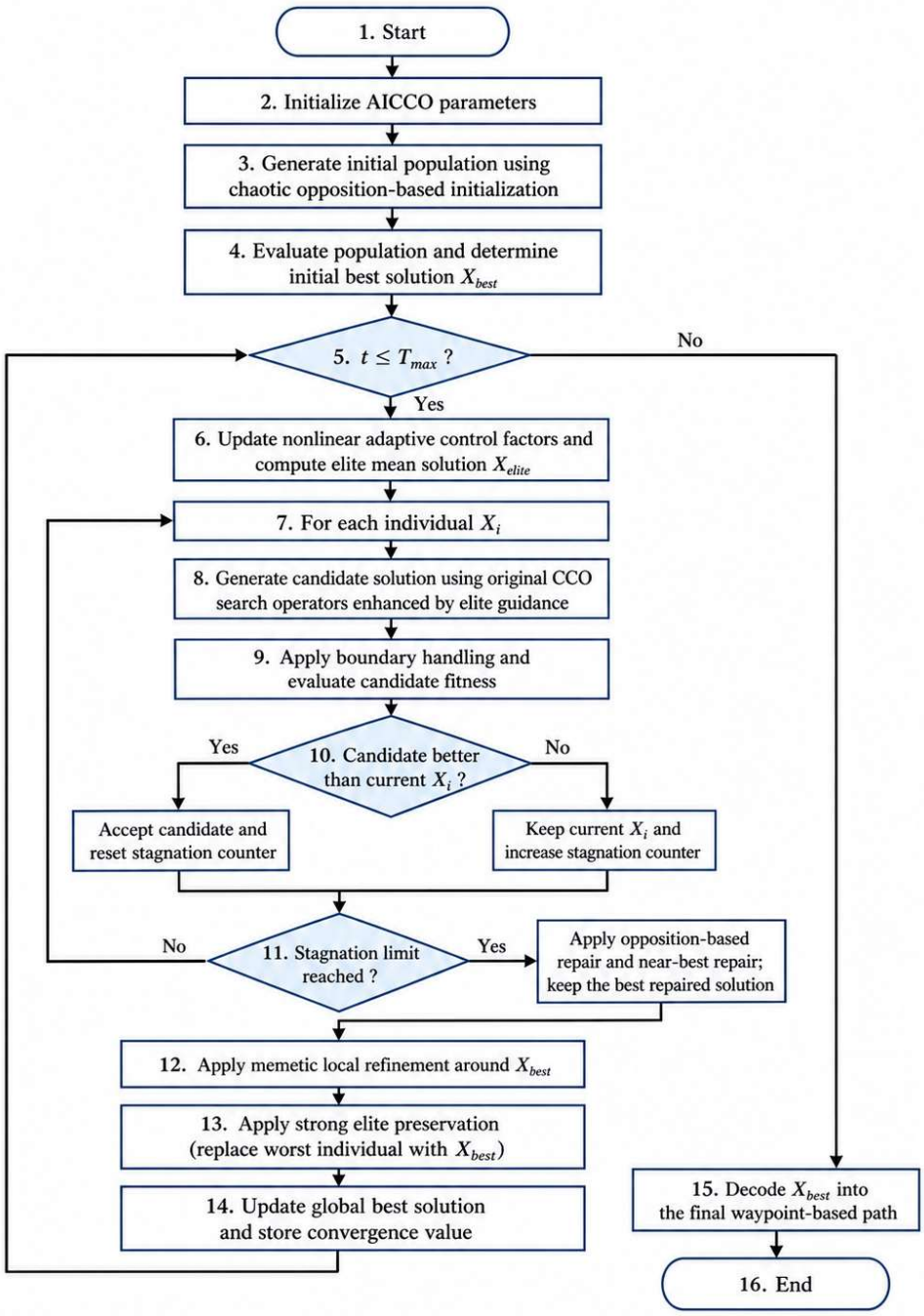


Figure 6. Flowchart of the proposed AICCO algorithm.

5.4. Computational Complexity Analysis

The main computational complexity of the proposed AICCO is based on the population size, number of iterations, size of a solution, fitness evaluation cost, calculation of the elite, and memetic local refinement. Suppose that N_p represents the population size, $T_{\max}$ the maximum number of iterations, D the dimension of each individual solution, and N_e the number of elite individuals.

During initialization phase, the proposed chaotic opposition-based initialization creates two different solutions for each individual: original chaotic solution and opposition solution. Hence, complexity of the initialization is:

$$O(2N_pD) \approx O(N_pD). \tag{71}$$

In every iteration, the original CCO search operators are used to change the location of each individual. Each solution has D dimensions, and the update process requires:

$$O(N_p D). \tag{72}$$

The objective function is used to assess the population. If the cost of evaluating one candidate solution is denoted by C_F, then the fitness evaluation cost per iteration is:

$$O(N_p C_F). \tag{73}$$

To sort the population to identify the best and elite solutions, one needs to:

$$O(N_p \log N_p). \tag{74}$$

The elite mean solution is computed by using the best N_e individuals, thus:

$$O(N_e D). \tag{75}$$

Memetic local refinement is also proposed around the best solution in the AICCO. If all dimensions are refined using coordinate data, its complexity is roughly as follows:

$$O(DC_F). \tag{76}$$

Thus, the overall iteration computational complexity is:

$$O(N_p D + N_p C_F + N_p \log N_p + N_e D + DC_F). \tag{77}$$

Therefore, the total computational complexity of the proposed AICCO with $T_{\max}$ iterations is:

$$O(T_{\max}[N_p D + N_p C_F + N_p \log N_p + N_e D + DC_F]). \tag{78}$$

The term $N_e D$ typically has a small effect on the total complexity since N_e is normally much smaller than N_p. Thus, the major computational expenses are the population update, fitness evaluation, sorting, and memetic local refinement.

In the mobile robot path-planning problem, the candidates are the different ways of generating a waypoint-based path. If there are n intermediate waypoints on the path, then:

$$D = 2n. \tag{79}$$

Path length, smoothness of path, and obstacle avoidance are combined in the fitness function. The path length goal is $O(n)$, as is the smoothness goal, and obstacle avoidance is $O(N_s * N_o)$ sampled path points and obstacles. Thus, the estimated cost for the fitness evaluation is

$$C_F = O(n + N_s N_o). \tag{80}$$

So, the computational complexity of the proposed AICCO for the robot path-planning problem can be expressed as:

$$O(T_{\max}[N_p D + N_p(n + N_s N_o) + N_p \log N_p + N_e D + D(n + N_s N_o)]). \tag{81}$$

The main memory complexity of the proposed AICCO is due to the storage of the population, fitness values, elite individuals, and auxiliary counters. Hence, the memory complexity is:

$$O(N_p D). \tag{82}$$

The proposed AICCO includes some new operations, including elite guidance, memetic local refinement, and stagnation-based repair, but these are lightweight operations compared to the entire population-based search process. Furthermore, they help to enhance the accuracy of convergence and also minimize the premature stagnation, thereby lowering the number of unproductive iterations needed to achieve good-quality solutions.

Thus, the proposed AICCO achieves an acceptable computational complexity while enhancing the path generated in terms of length, smoothness, and obstacle avoidance.

In addition to the above, the asymptotic computational complexity of the proposed AICCO, the platform used for implementation, the number of particles in the population, the number of iterations, the number of path points sampled, and the number of obstacles all influence the runtime performance of the algorithm. Hence, in Section 6.4.2 an empirical runtime comparison and scalability analysis is given for evaluating the actual computational cost of the proposed method in dynamic replanning scenarios.

6. Experimental Results and Discussion

This section outlines the experimental results and analysis, highlighting their significance for the project.

This section will examine the performance of the proposed Adaptive Improved Cuckoo-Catfish Optimizer (AICCO) from two angles. First, the optimization capability of the proposed algorithm is checked with 23 classical benchmark functions. Secondly, the proposed AICCO is adopted in the path planning of mobile robots in static and dynamic obstacle environments. Optimization accuracy, convergence behavior, path length, smoothness, obstacle clearance, collision avoidance, and computational time are compared.

6.1. Experimental Setup

MATLAB R2024a was used to evaluate the experiment. The proposed AICCO was compared to five popular optimization algorithms, namely Particle Swarm Optimization (PSO) [9], Gray Wolf Optimizer (GWO) [21], the Whale Optimization Algorithm (WOA) [22], Artificial Bee Colony (ABC) [10], and the original Cuckoo-Catfish Optimizer (CCO) [13] for benchmark testing. The comparison was made using 23 classical benchmark functions (F1–F23). To obtain statistically valid results, each algorithm was run separately 30 times for each function.

The population size was chosen to be 40 and the max number of iterations was chosen to be 500 for the benchmark experiments. The best, worst, mean, and standard deviation values from 30 independent runs were used to evaluate the performance. In addition, the average computation time and rank of each algorithm were recorded.

Two simulation scenarios were taken for robot path planning. In the first scenario, only static (circular) obstacles were placed and in the second scenario, dynamic (linear and circular) obstacles were placed. In both, a combination of path length, path smoothness, and obstacle avoidance was used in the objective function. The robot model was considered as a circular body and obstacles were inflated based on the robot radius, safety margin, uncertainty, and relative motion. The experimental parameter settings used in this study are summarized in Table 2.

Table 2. Experimental parameter settings.

Parameter	Value
Number of algorithms	6
Algorithms	PSO, GWO, WOA, ABC, Original CCO, Proposed AICCO
Test functions	F1–F23

Table 2. *Cont.*

Parameter	Value
Population size	40
Maximum iterations	500
Independent runs	30
Performance metrics	Best, Worst, Mean, Std, Rank, Time

6.2. Benchmark Results on F1–F23 Test Functions

To evaluate the proposed AICCO, 23 classical benchmark test functions (F1–F23) were used. Each of these functions possess a particular kind of searching property such as a unimodal landscape, multimodal landscape, and fixed-dimensional multimodal landscape. Hence, they can be used for evaluating the Exploration ability, Exploitation accuracy, Robustness, and Stability of the compared optimization algorithms.

Five popular optimization algorithms were used for comparison with the proposed AICCO, namely Particle Swarm Optimization (PSO) [9], Gray Wolf Optimizer (GWO) [21], Whale Optimization Algorithm (WOA) [22], Artificial Bee Colony (ABC) [10], and the original Cuckoo-Catfish Optimizer (CCO) [13]. The number of independent runs for each benchmark function for each algorithm is 30. The population size was defined as 40, and the maximum number of iterations as 500. Performance of each algorithm was measured by the best, worst, mean, standard deviation, rank, and average computation time.

Detailed statistical benchmark results are shown in three sub-tables to render tables more readable. Table 3a–c provides the best, worst, mean, standard deviation, rank, and average computation time for the algorithms compared over F1–F23. This restructuring maintains the full number of results, but simplifies the benchmark comparison.

Table 3. (**a**) Statistical comparison of the compared algorithms on benchmark functions F1–F8. (**b**) Statistical comparison of the compared algorithms on benchmark functions F9–F16. (**c**) Statistical comparison of the compared algorithms on benchmark functions F17–F23.

(**a**)

Fun	Algorithm	Best	Worst	Mean	Std	Rank	AvgTime
F1	PSO	0.010154947	0.231424512	0.096242405	0.066883774	6	0.028859267
	GWO	4.22615×10^{-36}	2.31728×10^{-33}	2.87298×10^{-34}	4.98108×10^{-34}	2	0.08606508
	WOA	1.32603×10^{-90}	2.16905×10^{-79}	9.35617×10^{-81}	3.98533×10^{-80}	2	0.018091653
	ABC	3.12291×10^{-6}	0.000171765	2.49203×10^{-5}	3.3746×10^{-5}	5	0.02218417
	Original CCO	2.36908×10^{-31}	4.5915×10^{-7}	3.06392×10^{-8}	9.16652×10^{-8}	4	0.06520362
	Proposed AICCO	1.66097×10^{-37}	3.12259×10^{-34}	3.92831×10^{-35}	6.83242×10^{-35}	2	0.06455466
F2	PSO	0.017295452	0.145732524	0.045524262	0.02392187	6	0.029031543
	GWO	1.56722×10^{-21}	1.04751×10^{-19}	2.44697×10^{-20}	2.32045×10^{-20}	2	0.088652593
	WOA	1.5772×10^{-59}	3.5568×10^{-51}	1.22426×10^{-52}	6.48711×10^{-52}	2	0.020727187
	ABC	0.003078662	0.011583643	0.005503666	0.001970935	5	0.024594733
	Original CCO	8.49254×10^{-17}	3.45005×10^{-5}	3.90451×10^{-6}	7.74841×10^{-6}	4	0.067198133
	Proposed AICCO	1.97451×10^{-20}	7.05498×10^{-19}	1.5044×10^{-19}	1.51429×10^{-19}	2	0.068832463
F3	PSO	209.6824605	1032.470693	501.3663847	231.2823739	4	0.140109357
	GWO	1.29709×10^{-9}	2.52839×10^{-5}	3.1716×10^{-6}	6.15208×10^{-6}	2	0.20160236
	WOA	14,712.17509	57,262.48819	34,499.01528	10,878.89473	6	0.1320324
	ABC	11,065.27706	23,716.55528	17,411.85465	3342.767236	5	0.129049957
	Original CCO	0.040160808	2.359611101	0.554534101	0.546058388	3	0.188335177
	Proposed AICCO	1.59195×10^{-19}	7.9157×10^{-17}	1.4818×10^{-17}	1.91366×10^{-17}	1	0.24974826

Table 3. *Cont.*

F4	PSO	2.376753481	5.691699651	3.875701808	0.818380024	4	0.028431917
	GWO	1.20858×10^{-8}	1.08669×10^{-6}	2.98295×10^{-7}	2.94062×10^{-7}	2	0.089687517
	WOA	0.68721831	89.46852332	45.68003168	27.79775284	6	0.01754155
	ABC	32.92161682	52.3974284	43.49875592	5.205085116	5	0.021615333
	Original CCO	0.001844734	0.574684477	0.29234704	0.147669376	3	0.097019273
	Proposed AICCO	5.21762×10^{-14}	6.45967×10^{-13}	2.27026×10^{-13}	1.73718×10^{-13}	1	0.063560397
F5	PSO	24.15037979	315.3298272	109.0455597	69.58211486	6	0.041080543
	GWO	25.97657169	28.31545463	26.83411123	0.753507796	2	0.102183887
	WOA	27.07188863	28.75818896	27.85365319	0.431023672	4	0.030954093
	ABC	0.468728693	146.7256059	47.04964994	39.11240986	5	0.032762717
	Original CCO	26.76643772	29.33653105	27.70828552	0.736693045	3	0.11299422
	Proposed AICCO	20.92919516	23.8807231	22.62847381	0.580281927	1	0.08708616
F6	PSO	0	3	1.133333333	0.776079152	6	0.030926267
	GWO	0	0	0	0	2.5	0.09221049
	WOA	0	0	0	0	2.5	0.019070603
	ABC	0	0	0	0	2.5	0.022645707
	Original CCO	0	1	0.033333333	0.182574186	5	0.126699733
	Proposed AICCO	0	0	0	0	2.5	0.0757692
F7	PSO	0.010836915	0.035286929	0.02037707	0.005112781	4	0.10966796
	GWO	0.000707484	0.004859138	0.00211853	0.001160163	1	0.28929282
	WOA	3.17265×10^{-5}	0.013702514	0.00270637	0.003476206	2	0.166208113
	ABC	0.111653126	0.327682485	0.225517472	0.052815358	6	0.174525873
	Original CCO	0.009059569	0.048273539	0.021279475	0.008037374	5	0.317273297
	Proposed AICCO	0.000594409	0.005162478	0.002768606	0.001243819	3	0.360030093
F8	PSO	−8075.255844	−5501.101753	−6962.851591	759.5585346	5	0.088282957
	GWO	−7784.440729	−3148.368073	−5784.2948	1289.559151	6	0.220711777
	WOA	−12,569.38026	−8400.836379	−10,603.75305	1620.75756	2	0.07949915
	ABC	−11,924.45371	−11,075.24487	−11,519.79496	198.9291652	1	0.067661913
	Original CCO	−9626.943808	−5895.978484	−7917.530305	848.013285	4	0.184759593
	Proposed AICCO	−9842.33	−5642.190337	−8345.645235	944.5339728	3	0.181433647

(**b**)

Fun	Algorithm	Best	Worst	Mean	Std	Rank	AvgTime
F9	PSO	16.95234219	82.60437145	43.92969095	16.0535322	5	0.077962673
	GWO	0	221.6926316	15.87373354	39.74226601	3	0.179244267
	WOA	0	5.68434×10^{-14}	1.89478×10^{-15}	1.03781×10^{-14}	1	0.048480127
	ABC	0.063469399	8.475084866	5.663320986	1.815145654	2	0.06081675
	Original CCO	1.125619821	71.99021474	32.99541904	18.48591465	4	0.198974393
	Proposed AICCO	37.80838231	117.404732	69.34847192	20.65685091	6	0.173104093
F10	PSO	0.028104026	0.997659054	0.161088507	0.201544592	5	0.09053803
	GWO	2.17604×10^{-14}	3.24185×10^{-14}	2.46025×10^{-14}	3.8918×10^{-15}	2	0.193573337
	WOA	4.44089×10^{-16}	7.54952×10^{-15}	3.52311×10^{-15}	2.23398×10^{-15}	2	0.056947743
	ABC	0.026330262	1.000729967	0.188468301	0.210440054	6	0.060743367
	Original CCO	7.54952×10^{-15}	0.000572441	5.37952×10^{-5}	0.000120276	4	0.082448573
	Proposed AICCO	3.9968×10^{-15}	1.03473×10^{-13}	1.01548×10^{-14}	1.7676×10^{-14}	2	0.128534983
F11	PSO	0.020173077	0.721996178	0.181626057	0.155208146	6	0.112948203
	GWO	0	0.027106336	0.003394417	0.007570747	1	0.20568694
	WOA	0	0.102067201	0.00340224	0.018634836	2	0.067940103
	ABC	0.000148848	0.08557384	0.018183854	0.022401885	4	0.076782457
	Original CCO	0	0.603186808	0.097478049	0.187355412	5	0.17670685
	Proposed AICCO	0	0.017236256	0.00427041	0.005710458	3	0.20980008
F12	PSO	1.62551×10^{-5}	0.105975844	0.011094634	0.031683995	3	0.467324547
	GWO	0.011998142	0.139643153	0.033367045	0.023576268	5	0.526573157
	WOA	0.002308999	0.096688908	0.012461589	0.016495179	4	0.379462517
	ABC	5.24117×10^{-7}	2.10848×10^{-5}	3.28259×10^{-6}	4.16825×10^{-6}	2	0.40804788
	Original CCO	0.004237312	0.429242834	0.069725859	0.102656998	6	0.478928503
	Proposed AICCO	1.10821×10^{-11}	1.41928×10^{-9}	4.49044×10^{-10}	3.96397×10^{-10}	1	0.787498293

Table 3. *Cont.*

F13	PSO	0.001087451	0.080131045	0.018769998	0.019292448	3	0.443083783
	GWO	0.09600126	1.148145001	0.481929107	0.22375547	6	0.304253213
	WOA	0.03931074	0.636900461	0.316961538	0.157034224	4	0.208820837
	ABC	1.87169×10^{-6}	0.000445341	4.13919×10^{-5}	8.62253×10^{-5}	2	0.21320072
	Original CCO	0.034682722	1.061048082	0.468476846	0.269841346	5	0.29892273
	Proposed AICCO	1.58039×10^{-9}	9.35543×10^{-6}	4.31795×10^{-7}	1.70903×10^{-6}	1	0.3996609
F14	PSO	0.998003838	6.903335694	2.381526768	1.893762074	5	0.25913898
	GWO	0.998003838	10.76318067	3.810948202	3.638815965	6	0.251253237
	WOA	0.998003838	10.76318068	2.209414508	2.666237567	4	0.248900127
	ABC	0.998003838	0.998003838	0.998003838	1.36973×10^{-14}	2	0.257497027
	Original CCO	0.998003838	0.998003838	0.998003838	5.00769×10^{-16}	2	0.286618707
	Proposed AICCO	0.998003838	0.998003838	0.998003838	1.48666×10^{-16}	2	0.342172977
F15	PSO	0.004361206	0.015366847	0.00546177	0.003358136	2	0.01916886
	GWO	0.004361207	0.037016492	0.016200693	0.014150752	6	0.020031683
	WOA	0.004363421	0.02297138	0.005707475	0.003741055	3	0.01388873
	ABC	0.004440625	0.007507708	0.005259741	0.000768536	1	0.01906142
	Original CCO	0.004361206	0.036414181	0.007630772	0.007030564	5	0.057370403
	Proposed AICCO	0.004361206	0.036414181	0.007598635	0.008517184	4	0.047523383
F16	PSO	−1.031628453	−1.031628453	−1.031628453	6.32085×10^{-16}	2	0.021850707
	GWO	−1.031628453	−1.031628424	−1.031628446	6.52283×10^{-9}	6	0.018575933
	WOA	−1.031628453	−1.03162845	−1.031628453	7.07226×10^{-10}	5	0.01754167
	ABC	−1.031628453	−1.031628453	−1.031628453	4.87871×10^{-16}	2	0.022738097
	Original CCO	−1.031628453	−1.031628453	−1.031628453	1.75881×10^{-10}	4	0.058911333
	Proposed AICCO	−1.031628453	−1.031628453	−1.031628453	6.32085×10^{-16}	2	0.049790867

(**c**)

Fun	Algorithm	Best	Worst	Mean	Std	Rank	AvgTime
F17	PSO	0.397887358	0.397887358	0.397887358	0	1.5	0.015343147
	GWO	0.397887372	0.398387484	0.397917362	0.000113138	6	0.011851213
	WOA	0.397887358	0.397907365	0.397890426	4.8273×10^{-6}	5	0.009278477
	ABC	0.397887358	0.397887574	0.397887365	3.94993×10^{-8}	4	0.015348687
	Original CCO	0.397887358	0.397887358	0.397887358	2.20179×10^{-11}	3	0.05097052
	Proposed AICCO	0.397887358	0.397887358	0.397887358	0	1.5	0.043207257
F18	PSO	3	3	3	1.3399×10^{-15}	1.5	0.015437753
	GWO	3.000000209	3.000109844	3.000027001	3.1239×10^{-5}	5	0.01137073
	WOA	3.000000004	3.000100582	3.00001192	2.20667×10^{-5}	4	0.009368817
	ABC	3	3.011543976	3.001459553	0.002708401	6	0.014736307
	Original CCO	3	3.000000024	3.000000001	4.30399×10^{-9}	3	0.053221237
	Proposed AICCO	3	3	3	2.33101×10^{-15}	1.5	0.042923037
F19	PSO	−3.862782148	−3.862782148	−3.862782148	2.66836×10^{-15}	2	0.026236177
	GWO	−3.862781983	−3.854900492	−3.862085193	0.002002729	5	0.02369658
	WOA	−3.862782038	−3.853204462	−3.860645488	0.002614527	6	0.019090483
	ABC	−3.862782148	−3.862782148	−3.862782148	6.9535×10^{-14}	2	0.02489608
	Original CCO	−3.862782148	−3.86278214	−3.862782147	1.5351×10^{-9}	4	0.06252063
	Proposed AICCO	−3.862782148	−3.862782148	−3.862782148	2.65431×10^{-15}	2	0.055902047
F20	PSO	−3.322368011	−3.203161918	−3.262764965	0.060621976	3.5	0.02772429
	GWO	−3.322367037	−3.134504825	−3.249548309	0.076897048	5	0.0327898
	WOA	−3.322153918	−3.063195662	−3.220856224	0.106848279	6	0.019741537
	ABC	−3.322368011	−3.322368011	−3.322368011	4.30668×10^{-14}	1	0.02658269
	Original CCO	−3.322368011	−3.203161551	−3.270711996	0.060080725	2	0.06795071
	Proposed AICCO	−3.322368011	−3.203161918	−3.262764965	0.060621976	3.5	0.05936142
F21	PSO	−10.15319968	−2.630471668	−6.242630622	3.746179293	6	0.025694243
	GWO	−10.15315266	−5.099895485	−9.479047604	1.746937744	2	0.0245393
	WOA	−10.15244965	−5.015845737	−7.906686699	2.544854189	4	0.017048273
	ABC	−10.15319968	−10.11814286	−10.15088291	0.007503218	1	0.024207337
	Original CCO	−10.15319968	−2.630471668	−8.552089665	2.524543186	3	0.06151102
	Proposed AICCO	−10.15319968	−2.630471668	−6.617049032	3.073694092	5	0.055254953

Table 3. *Cont.*

F22	PSO	−10.40294057	−2.751933564	−7.392810327	3.561544107	5	0.028319707
	GWO	−10.40286421	−5.128540996	−10.22669409	0.962887434	2	0.026730863
	WOA	−10.40254823	−1.837371999	−8.031310566	3.256146491	4	0.020313847
	ABC	−10.40294057	−10.13550257	−10.37529089	0.068653529	1	0.027230983
	Original CCO	−10.40294057	−3.724300345	−8.630692155	2.7801273	3	0.06558314
	Proposed AICCO	−10.40294057	−1.837592971	−5.739107985	3.311382235	6	0.05871098
F23	PSO	−10.53640982	−2.421734027	−7.994867237	3.683361513	4	0.032139
	GWO	−10.53629009	−2.421708142	−10.26541606	1.481439941	3	0.031454233
	WOA	−10.53593583	−2.421626862	−7.84246088	3.203186519	5	0.024554147
	ABC	−10.53640982	−10.35790367	−10.52701885	0.034043213	1	0.02991021
	Original CCO	−10.53640982	−3.835426801	−10.31304343	1.223426465	2	0.069760107
	Proposed AICCO	−10.53640982	−2.421734027	−6.667805185	3.576491989	6	0.063940323

Table 3a–c presents the statistical results of the proposed AICCO performance with respect to the 23 benchmark functions, demonstrating that the proposed AICCO performed well overall. The proposed AICCO achieved the best performance on seven benchmark functions (F3, F4, F5, F12, F13, F14, and F18) based on the mean fitness value. Furthermore, it was also competitive or near-best on a number of other functions. The results show that the proposed improvement mechanisms improve the optimization capability of the original CCO and enable it to work well on various search landscapes.

The proposed AICCO achieved better mean fitness values on 18 of the 23 benchmark functions compared to the original CCO. This shows that the proposed modifications led to a considerable improvement in the search behavior of the original algorithm. The improvement is due to the chaotic opposition-based initialization, nonlinear adaptive control, enhancement of search by the elite, strong elite preservation, memetic local refinement, and stagnation-based opposition repair. The mechanisms help to achieve greater initial population diversity, direct the search to promising areas, maintain high-quality solutions, and diminish the risk of premature stagnation.

The results of average ranking and computation time are shown in Table 4.

Table 4. Average rank and computation time of the compared algorithms.

Algorithm	Winning Functions	Average Rank	Overall Rank	Average Time (s)
Proposed AICCO	7	2.6957	1	0.1595
ABC	6	3.1087	2	0.0773
GWO	3	3.5870	3	0.1318
WOA	4	3.7174	4	0.0715
Original CCO	0	3.7391	5	0.1404
PSO	3	4.1522	6	0.0939

The ranking procedure was conducted using fitness value, which is computed by each algorithm on each benchmark function after 30 independent runs. All the benchmark functions were taken as minimization problems and the mean fitness value of each algorithm for a function was ranked: the best mean fitness value was ranked 1, the second best mean fitness value was ranked 2, and so on. If two or more algorithms had identical or numerically indistinguishable mean values, tied ranks were dealt with by assigning the average of the rank positions corresponding to the tied algorithms. For instance, two algorithms ranked first received a rank of 1.5.

For each algorithm, the average rank was calculated as:

$$\text{Average Rank}_j = \frac{1}{23}\sum_{f=1}^{23} R_{j,f}, \tag{83}$$

where $R_{j,f}$ is the rank of the j-th algorithm on the f-th benchmark function. The algorithms were then ranked in ascending order based on their average rank, which was used to calculate the overall rank. A lower average rank means that the overall benchmark performance is higher. As per this procedure, the proposed AICCO achieved a total rank sum ~62 for the 23 benchmark functions, giving the average rank value as 62/23 = 2.6957.

An average rank of 2.6957 was obtained for the proposed AICCO, which was also the best overall rank among all the algorithms compared in Table 4. ABC had the second best average rank, and it was followed by GWO, WOA, the original CCO, and PSO. The proposed AICCO also had the most winning functions; 7 out of 23 functions. The original CCO, on the contrary, failed to obtain the highest mean value on any of the benchmark functions. This clearly shows that the proposed AICCO is an enhancement of the original CCO and that the main search structure is retained.

The proposed AICCO had a slightly higher average computation time compared with some of the comparison algorithms, but this is expected since the proposed method employs more improvement mechanisms such as elite-guided enhancement, memetic local refinement, strong elite preservation, and stagnation-based opposition repair. This increased computational cost is justified by the improvement in the quality of the solution, robustness, and average rank.

There are 23 benchmark functions and 30 independent runs for the respective algorithms; Tables 3a–c and 4 provide statistical indicators, which fully test the optimization performance. Hence, emphasis is placed on best, mean, standard deviation, rank, and computation time and not on providing more convergence curves. This prevents too much 'fat' and gives a good, statistically valid comparison and clarity.

In general, the benchmark results show that the proposed AICCO outperforms PSO, GWO, WOA, ABC, and the original CCO in terms of general optimization performance. Based on these results, the proposed improvement mechanisms are proven to be effective, and the use of AICCO for the mobile robot path-planning problem is supported in the next experiments.

6.3. Static Obstacle Path-Planning Results

In an obstacle path-planning scenario that is static, the proposed AICCO was tested to produce short, smooth, and collision-free paths. The workspace of the simulation environment was set to be 100 × 100, 2D workspace. The robot was initially at the start point SP = (5, 5) and traveled towards the goal point GP = (95, 90). There were six round static obstacles in the environment, and each obstacle was expanded by using the robot radius, safety margin, and sensing uncertainty to achieve safe navigation. A path was constructed by using seven intermediate waypoints between the start and goal positions.

Three criteria that were chosen for the objective function were path length, smoothness of the path, and avoidance of obstacles. In this experiment, the weights were set as $w_{Len} = 0.40$, $w_{Smooth} = 0.25$, and $w_{Safe} = 0.35$. The number of individuals in the proposed AICCO was 45, with a maximum of 220 iterations. Collision checking was performed using sample points on each path segment instead of just checking waypoints, enhancing obstacle avoidance.

The results of the proposed AICCO are a successfully generated proper path from start point to goal point without crossing any static obstacles, as shown in Figure 7. This optimized path goes through the workspace without any obstacles and lies outside the

expanded obstacle boundary, indicating that the obstacle expansion method is able to ensure that the robot maintains a safe distance from the obstacles. The smoothness term also helps to create a more feasible path for the robot's motion, and the path is relatively smooth with fewer sharp turns.

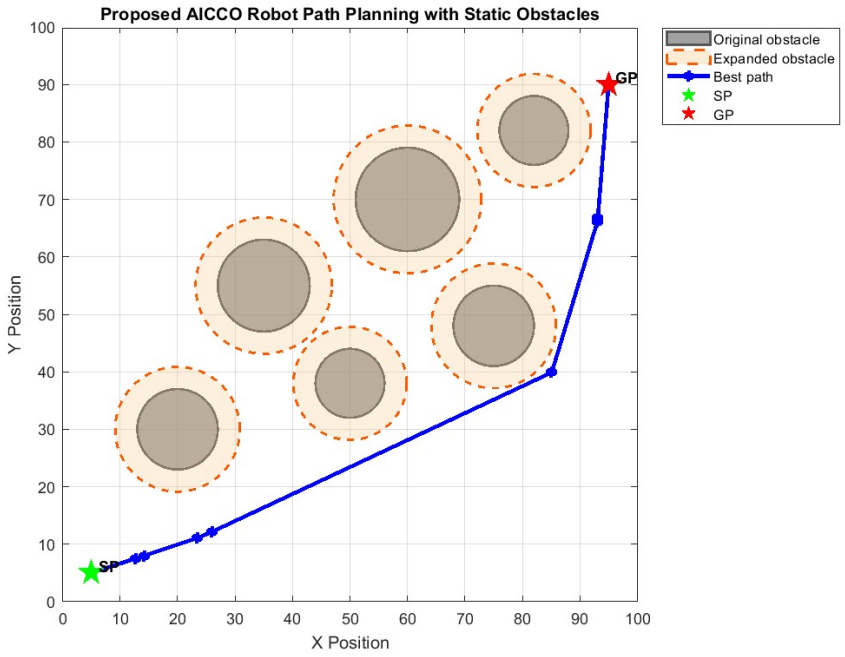


Figure 7. Static obstacle path planning result using the proposed AICCO.

The generated path is indicative of the fact that the optimizer is not just looking for the shortest possible geometric path. Rather, it takes into consideration the length of the trail along with the clearance of obstacles and smoothness. This is significant because the shortest route could be too near obstacles or have too many bends. Thus, the suggested AICCO is a safer and more feasible way for mobile robots to navigate in static environments.

Table 5 also presents numerical results which further show the effectiveness of the proposed AICCO in the static environment. The optimized path (in the image) was able to successfully pass through a feasible path while avoiding obstacle regions, with a total path length of 138.838369. The smoothness value of 1.173386 rad indicates that the path has relatively small variations in turning angle, and no overly sharp turns. The generated path is thus more appropriate for practical mobile robot motion.

Table 5. The output of obstacle path planning by the proposed AICCO.

Metric	Value
Best cost	0.4240775684
Path length	138.838369
Smoothness	1.173386 rad
Minimum clearance	0.785565
Collision penalty	0.000000
Computation time	28.064567 s

The minimum clearance value is 0.785565, which is a positive value and assures that the generated path is not within the expanded obstacle boundaries. Furthermore, the collision penalty is 0.000000, which suggests that there was no collision along the path. The results show that the proposed AICCO can effectively meet safety constraints and still have a reasonable path length and smoothness.

The computation time was 28.064567 s, which is acceptable for offline static path planning in which the path is computed once before it is executed. This extra computing overhead is primarily the result of using memetic local refinement, elite preservation, and dense collision checking along the path segments. However, these mechanisms help to make paths more feasible, safe and smooth.

In general, the static obstacle experiment shows that the proposed AICCO can produce a collision-free path whilst considering path length, smoothness and safety. This is a good result, which can be used to support the fact that the proposed optimizer can be used for mobile robot path planning in structured environments in which obstacles are fixed.

The convergence curve of the proposed AICCO for static obstacle path planning is shown in Figure 8. The best cost drops quickly in the initial few iterations, showing that the algorithm can quickly find good path regions. The cost function is reduced at a fast pace at first, then slowly reduces until it stabilizes at a steady value after the final few iterations. This behavior proves that the proposed AICCO has an excellent compromise between early-stage exploration and local refinement in the later stage. The obtained value for convergence is the same as the best cost in Table 5, indicating the convergence of the optimization process.

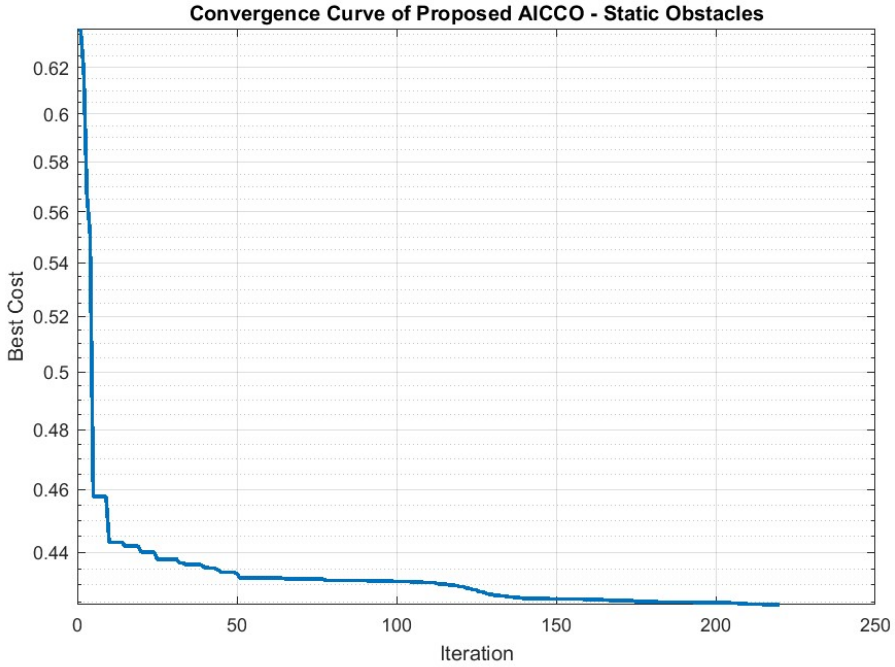


Figure 8. Convergence curve of the proposed AICCO for static obstacle path planning.

Weight Sensitivity Analysis

To better understand the impact of the weighting coefficients (w_1), (w_2), and (w_3) in the proposed weighted fitness function, a Weight Sensitivity Analysis was performed with the same static obstacle environment. Four favored configurations of weights were tried: balanced preference, length-priority preference, smoothness-priority preference, and safety-priority preference. In every case, the same AICCO parameters and obstacle map were applied, with the only difference being the objective weights, to test the ability of the proposed formulation to offer the user choices in navigation preferences within collision-free path generation.

The results of the weight sensitivity analysis are shown in Table 6. The balanced case resulted in a path length of 137.345802, a smoothness of 0.969008 rad, and a minimum clearance of 1.028402. The case with length priority had the shortest path of the cases

tested, with a path length of 137.085852 and collision penalty of 0. The safety-priority case produced the widest minimum clearance of 1.509797, which suggests that the greater the safety weight (w_3), the more paths will be found further away from the obstacles. The smoothness-priority case did not yield the lowest value of smoothness, but still resulted in a collision-free path with acceptable smoothness and positive clearance. This is expected as path length, smoothness, and obstacle clearance are competing objectives and changes to one can impact the others.

Table 6. Weight sensitivity analysis.

Case	(w_1)	(w_2)	(w_3)	Path Length	Smoothness (rad)	Minimum Clearance	Collision Penalty
Balanced	0.40	0.30	0.30	137.345802	0.969008	1.028402	0
Length-priority	0.48	0.27	0.25	137.085852	1.007037	0.733558	0
Smoothness-priority	0.32	0.43	0.25	137.877989	1.168401	0.975687	0
Safety-priority	0.28	0.25	0.47	137.989572	1.011196	1.509797	0

In general, all of the tested weight configurations were able to navigate without collision, and had a positive minimum clearance and no collision penalty. The above findings suggest that the proposed AICCO is not limited to one specific weight setting. Instead, the same weighted formulation is adapted based on the different priorities of navigation and map conditions. Hence, fixed weights were employed in the main comparative experiments to allow for a fair and reproducible evaluation, and the sensitivity analysis shows that the proposed objective function could be flexible.

These results support the use of fixed weights in the main comparative experiments for fairness and reproducibility, while showing that the same scalarized formulation can be adjusted to different navigation priorities without losing collision-free behavior.

6.4. Dynamic Online Replanning Results

For the greater assessment of the proposed AICCO under more realistic navigation conditions, a dynamic obstacle scenario was considered. The robot, unlike the static case, does not take a single path which is calculated only at the start of the navigation. Instead, an online replanning strategy is used. At every decision step, the current robot position is considered as a new local start point, the local positions of dynamic obstacles are updated, and the proposed AICCO generates a new local path to the goal. Then, the robot performs only a small motion step on the newly generated trajectory and replans again.

The dynamic environment was represented as a workspace of 100 × 100, which is a 2D space. The robot was initially located at SP = (5, 8) and was moving towards GP = (95, 90). Moving circular obstacles of linear motion and circular motion were present in the environment. In this case, the obstacle locations were varied to see how well the proposed method could adjust to the obstacle locations and still navigate without collision.

The proposed AICCO was used as a local planner in the online replanning process. The optimizer used waypoints to create a new path from the robot's current location to the goal every time the replanning was performed, taking into account the new positions of the obstacles and the larger safety margins. This generated path was not fully followed, unlike offline planning. Instead, only a small portion of the newly created path was carried out and a new replanning step was taken. In this way, the robot trajectory can be adjusted continuously to the dynamic environment.

The proposed AICCO was able to generate an adaptive executed trajectory in the dynamic environment, as shown in Figure 9. The robot traversed from the starting position towards the goal, while keeping out of the enlarged regions of the moving obstacles. The trajectory indicates that the robot was not using a predetermined trajectory. It rather continuously modified its motion based on the updated location of the obstacles. This

behavior is crucial since a path can be safe at one time and unsafe at another because of moving obstacles. Thus, with the online replanning strategy, the robot is better able to circumvent moving obstacles while still making progress towards the goal.

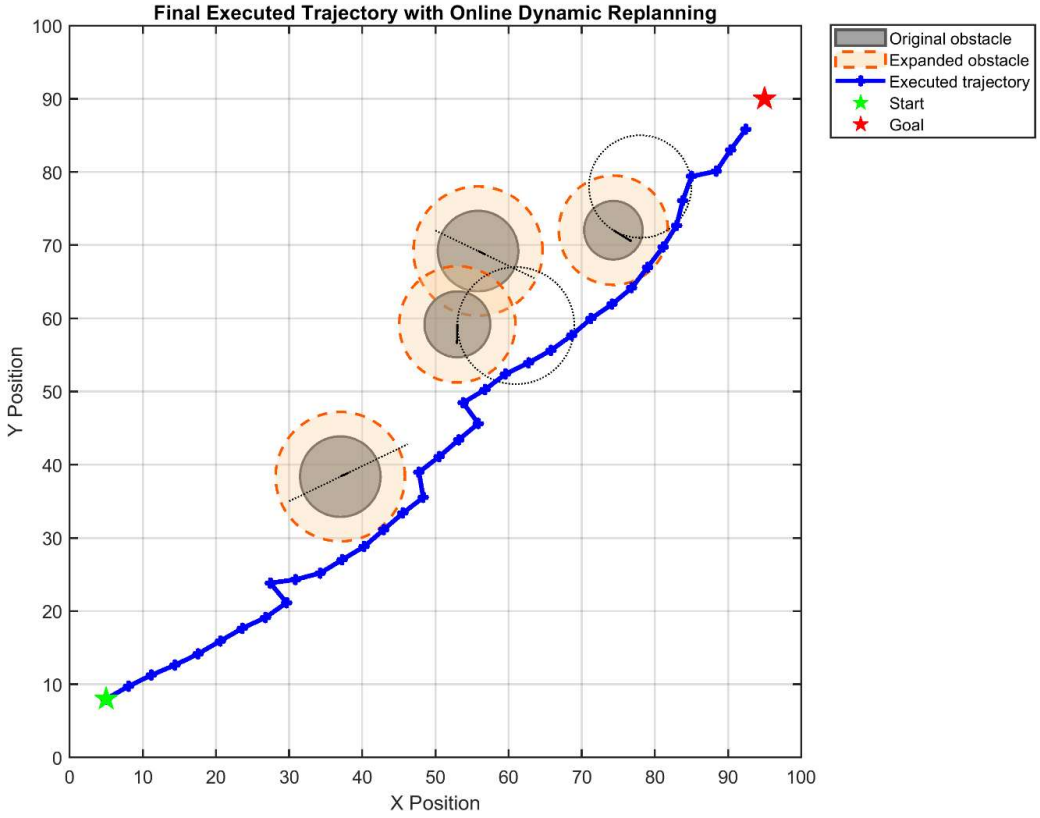


Figure 9. Final executed trajectory using the proposed AICCO with online dynamic replanning. The gray filled circles represent the original obstacle positions, the orange dashed circles represent the expanded obstacle boundaries, the black dashed circles/lines indicate intermediate or previous positions of dynamic obstacles during the replanning process, and the blue curve represents the executed trajectory from the start point to the goal point.

Table 7 also shows the numerical results to further validate the proposed AICCO performance in the dynamic scenario. The robot made it through the navigation process with 37 online replanning steps and finished with an executed path length of 129.500000. The collision count was 0, which shows that during the navigation, no collision with any dynamic obstacle was detected, and hence the proposed method succeeded in avoiding collisions with such obstacles. The minimum actual clearance was 1.514250, which is a positive value, meaning that the actual trajectory performed was outside the expanded obstacle boundaries.

Table 7. Dynamic online replanning results.

Metric	Value
Actual executed path length	129.500000
Actual executed smoothness	13.707685 rad
Minimum actual clearance	1.514250
Collision count	0
Total navigation time	25.900000 s
Mean replanning solve time	3.924900 s
Number of replanning steps	37

The value of the smoothness that was executed was 13.707685 rad. This value is greater than the result of the static path-planning case due to the fact that the robot's path is dynamically replanned based on dynamically changing obstacles. Dynamic online replanning is a case where more heading changes may be necessary as the robot needs to adjust to evolving obstacles, instead of moving along a single globally optimized static path. The smoothness value, therefore, indicates the adaptiveness of the online navigation process.

It should be noted that the mean replanning solve time of 3.924900 s was based on a MATLAB R2024a implementation. Hence, the current implementation should be viewed as a validation of the proposed online replanning concept based on simulation, and not as a complete real-time deployment for high-speed mobile robots. This computation time might be tolerated for low-speed robotic platforms, or for near-online navigation scenarios and offline evaluation, but it might be too high for high-speed mobile robots demanding control frequency. The computational expense is primarily due to the repeated population-based optimization at each replanning step, fitness evaluation performed at sampled path points, and repeated collision checking with the updated locations of obstacles.

The online replanning steps are shown regarding the local planning cost, predicted minimum clearance, and replanning computation time in Figure 10. The cost in the locality is generally low in most of the replanning steps but increases slightly at the end. This behavior can happen when the robot is moving towards the goal location with moving obstacles nearby, impacting the safe directions list. The clearance is determined by the planner and is kept positive during navigation, thus ensuring that safe distances are kept from the dynamic obstacles. The computation time curve further indicates that the replanning process does not drastically change throughout most steps.

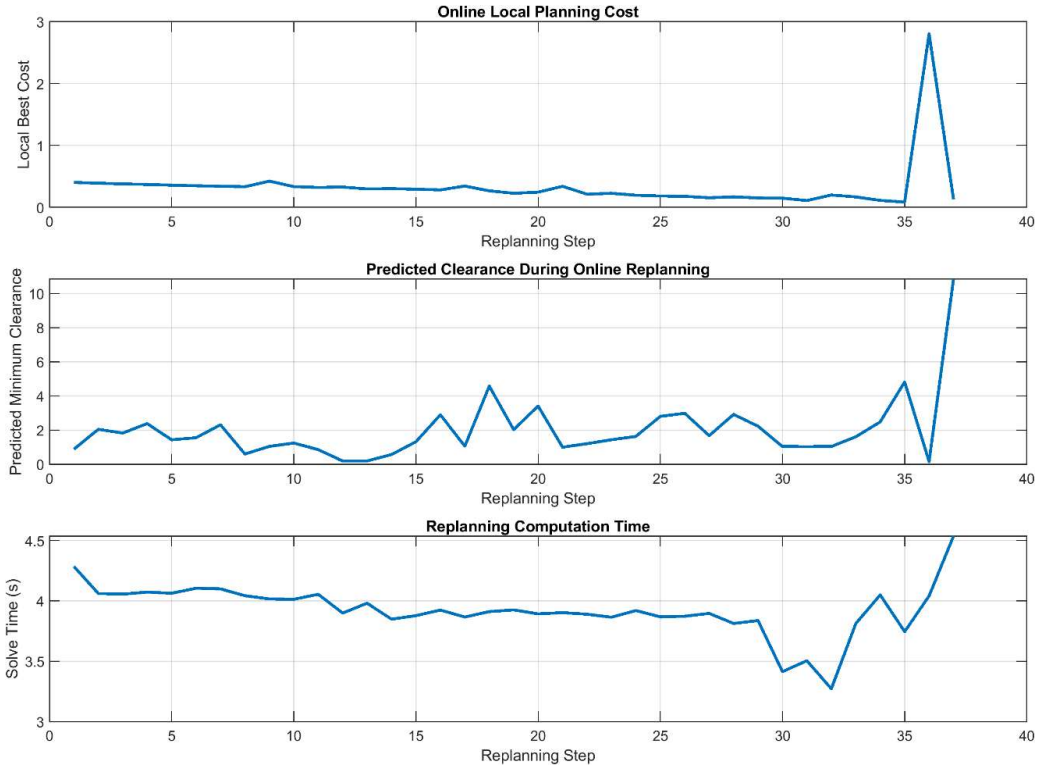


Figure 10. Online replanning performance metrics using the proposed AICCO.

In summary, the dynamic online replanning experiment shows that the proposed AICCO can produce a safe and adaptive trajectory in changing environments. The robot's navigation is collision-free through the use of dynamic obstacle prediction, adaptive obstacle expansion, and repeated local replanning as it progresses towards the goal. These

results justify the applicability of the proposed AICCO for static path planning not only for static environments, but also for dynamic environments where positions of obstacles change over time.

6.4.1. Additional Nonlinear/Reactive Dynamic Obstacle Scenario

To test the proposed AICCO in more demanding dynamic situations, another nonlinear/reactive obstacle scenario was tested. In contrast to the previous dynamic scenario where the moving obstacles were following linear and circular motion, in this scenario objects were moving with time-varying nonlinear motion and were reactive. Some obstacles alter their velocity based on sinusoidal acceleration components; others alter their direction when the robot enters a predefined interaction distance.

This extra scenario was created to verify the safety of the proposed online replanning approach in case of more unpredictable obstacle behavior. The robot performs replanning, updates the obstacle positions, calculates the extended safety regions, and replans a local path with AICCO at every replanning step. The robot then only follows a small portion of the generated path and plans again.

The trajectory that the robot executed in the nonlinear/reactive dynamic scenario is presented in Figure 11. It shows that the robot can be steered by the proposed AICCO to adapt to varying obstacle behavior without causing any collisions. The quantitative results of the nonlinear/reactive dynamic obstacle scenario are summarized in Table 8.

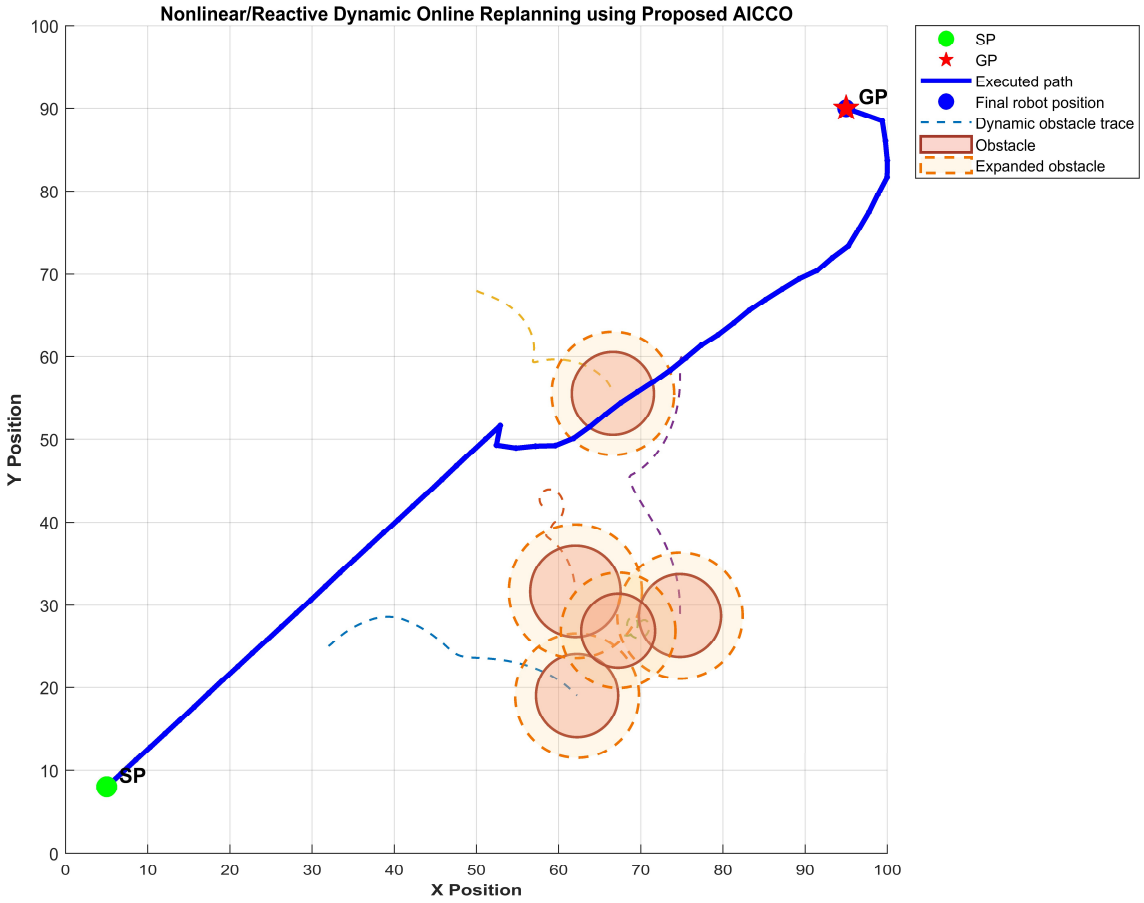


Figure 11. Executed trajectory under nonlinear/reactive dynamic obstacle motion using the proposed AICCO. The orange and purple dashed lines indicate the motion traces of the nonlinear/reactive dynamic obstacles during the replanning process.

Table 8. Nonlinear/reactive dynamic obstacle results.

Metric	Value
Actual executed path length	138.6437
Actual executed smoothness	8.3530
Minimum actual clearance	0.4563
Collision count	0
Total navigation time	34.2
Mean replanning solve time	7.0769
Number of replanning steps	57
Final distance to goal	0

The extra nonlinear/reactive scenario is a more complex dynamic situation than the simple linear/circular motion scenario. The robot was able to adapt to the time-varying and reactive obstacle motions and reach the goal, as demonstrated in Figure 11. The path length was 138.643784 and the smoothness was 8.353065 rad. The minimum actual clearance was still positive at 0.456385, and the number of collisions was still zero. These results show that the proposed AICCO can still ensure safe navigation, even when the motion of obstacles is less predictable. The number of replanning steps, however, rose to 57 and the mean replanning solve time rose to 7.076918 s, which shows that the more complex the dynamic scenario, the more complex the computation.

6.4.2. Runtime Comparison and Scalability Analysis

To support the theoretical complexity analysis of computation, an empirical time comparison was performed in the dynamic online replanning scenario. The aim of this experiment was to compare the practical computational expense of the proposed AICCO with that of the baseline optimizers for the same replanning situation. The same number of waypoints, the same objective function, the same number of obstacles, and the same obstacle-update procedure were used for all algorithms. A reduced computational setting was used for this runtime experiment to ensure a fair and fast comparison between the algorithms. The mean replanning time was defined as the mean optimization time needed to compute a new local path at each replanning.

The runtimes of the compared algorithms for the dynamic online replanning scenario are shown in Table 9. The outcomes indicate that WOA takes the least mean replanning time of 0.240458 s compared to all other algorithms, which consist of the original CCO, GWO, PSO, ABC, and the proposed AICCO. The replanning time for the proposed AICCO was the most expensive, with a mean of 0.639730 s. This increment is anticipated due to the extra mechanisms implemented in AICCO like chaotic opposition-based initialization, elite-guided enhancement, strong elite preservation, memetic local refinement, and stagnation-based opposition repair.

Table 9. Runtime comparison in dynamic online replanning.

Algorithm	Mean Replanning Time (s)	Path Length	Min Clearance	Collisions	Success
PSO	0.302524	133.792125	−1.870310	5	1
GWO	0.300012	175.000000	0.191677	0	0
WOA	0.240458	157.079212	0.540617	0	1
ABC	0.603579	126.685965	−5.337902	11	1
Original CCO	0.292062	126.725071	−5.623481	5	1
Proposed AICCO	0.639730	176.237731	0.780129	0	1

The proposed AICCO was more costly to compute but yielded goal-reaching success without any collisions and a minimum clearance of 0.780129. AICCO and WOA were the

two successful collision-free methods with the highest minimum clearance and the fastest computation time. Therefore, there is a compromise between runtime and path safety. For applications where safety and obstacle clearance are major concerns, the proposed AICCO is more appropriate for the applications, while for very low computation time applications, simpler optimizers can be used.

The proposed AICCO was also tested for scalability by varying the population number and the number of iterations. The mean replanning time for each of the three computational settings is given in Table 10. As expected, there was an increase in the average replanning time from 0.245651 s in the small setting to 0.520872 s in the large setting. This trend is in accordance with the theoretical complexity analysis: the dominant computational cost is primarily dependent on the population size, the maximum number of iterations, and the cost of evaluating candidate paths.

Table 10. Runtime scalability analysis of AICCO.

Case	Population	Iterations	Mean Replanning Time (s)
Small	12	25	0.245651
Medium	18	35	0.405192
Large	22	45	0.520872

These results confirm that the practical runtime of AICCO increases as the computational budget increases. For real-time implementation, acceleration techniques like parallel fitness evaluation, vectorized implementation, MEX/C++ acceleration, GPU computation, warm-start replanning, adaptive stopping criteria, and hybridization with faster local planners should be considered.

6.5. Discussion

Experimental results show that the proposed AICCO can work well in the numerical optimization and mobile robot path-planning applications. The results of the benchmark presented in F1–F23 indicate that the proposed AICCO obtained the highest overall rank when compared with the other algorithms. It achieved the best mean fitness value on seven benchmark functions, and an average rank of 2.6957. The results indicate that the proposed AICCO achieved better mean fitness values on the majority of the benchmark functions compared to the original CCO, which can be interpreted as the effect of the introduction of the improvement mechanisms that bolstered the search behavior of the original algorithm.

The combined effect of the proposed enhancement strategies is the reason for the improvement in the benchmark results. Chaotic opposition-based initialization enhances the initial population distribution and boosts the chances of its starting from good zones. The nonlinear adaptive control factor is used to make a smoother transition from exploration to exploitation. Elite-guided enhancement prioritizes the search towards high-quality areas and retains the original CCO search behavior. If the elite preservation is strong, then the best solution will not be lost during stochastic updating. Furthermore, memetic local refinement is capable of enhancing the exploitation ability around the best solution, while the stagnation-based opposition repair can help to regenerate stagnant individuals and prevent premature convergence.

The static path-planning results also show the appropriateness of the proposed AICCO for mobile robot navigation. The generated path was successful in avoiding all expanded obstacle regions and the collision penalty was zero. The minimum clearance value was 0.785565 which shows that the path was never inside the extended obstacle boundaries. Furthermore, the smoothness value was 1.173386 rad, which means that there was not a lot of variation in the turning angle of the generated path, and no excessive sharp turns were

taken. From these results, it can be observed that the proposed optimizer can achieve a balance between the three major path quality objectives; these are path length, smoothness, and obstacle avoidance.

In the dynamic online replanning scenario, the proposed AICCO also showed adaptive behavior with moving obstacles. The robot took 37 replanning steps to complete the navigation process—the actual path length executed was 129.500000 and the number of collisions was zero. The positive minimum actual clearance of 1.514250 verifies that the robot was able to keep a safe distance from the expanded dynamic obstacles during navigation. This is significant because it means that dynamic environments do not need to be precomputed, but can rather be continuously adapted to. In the proposed online replanning strategy, the robot replanned its trajectory based on the detected robot's position and the estimated motion of the obstacles around it.

This repeated replanning and local safety handling in the varying environments is also in line with previous motion-planning and collision-avoidance research. Sampling-based planners like PRM and RRT* help in increasing coverage of search space and improving the quality of its outcome [23,24], and incremental planners like D* and D* Lite help in efficient replanning in partially known or changing environments [25,26]. Velocity-space search, local sensing, and continuous obstacle avoidance are also important features of reactive local methods like the dynamic window approach and the vector field histogram [27,28]. In addition, the concept of safety based on inevitable collision states and trajectory-optimization methods like CHOMP also encourage the consideration of dynamic safety and smoothness during navigation [29,30].

The dynamic results also indicate that online replanning imparts some extra computational burden as well as potentially greater values for trajectory smoothness than static planning. The actual executed smoothness was 13.707685 rad, which was more than the static smoothness value as the robot changed its direction multiple times due to moving obstacles. This is acceptable in dynamic navigation situations where the robot's safety and adaptability are more important than maintaining a smooth path. The mean replanning solve time was 3.924900 s, which shows that the proposed method is more computationally demanding compared to the single shot static planning. However, this expense is tolerable for simulation validation and can be minimized in future research by employing parallel implementation, adaptive iteration limits, or faster local replanning methods.

The proposed AICCO was able to navigate without collision in static, dynamic, and nonlinear/reactive simulation cases; however, the present validation was performed in MATLAB. The results should not, therefore, be regarded as a real-robot real-time validation. From the mean replanning solve time of 3.924900 s in the dynamic scenario, it can be seen that the current implementation is computationally intensive. This restriction is more pronounced in more complex dynamic scenarios, including the nonlinear/reactive obstacle scenario in which there is more obstacle prediction and replanning, requiring more computation.

Therefore, the current implementation is more appropriate for simulation-based validation, low-speed mobile robots, and planning scenarios with replanning frequencies that are not too high. There are several acceleration strategies that can be used in order to improve real-time applicability. The first is that the fitness of each member of the population can be parallelized since each candidate path is evaluated independently. Second, the MATLAB implementation can be optimized by vectorizing, writing MEX/C++ functions, or using GPU acceleration. Thirdly, the optimizer can be warm-started with the best path found from the last replanning step, rather than starting the search process from the beginning. Fourth, adaptive stopping criteria are available to stop the optimization process as soon as a safe and acceptable path is identified. Fifth, a hybrid global-local approach can be

obtained, with AICCO producing good-quality global or medium-horizon waypoint paths and a faster local planner providing short-term collision avoidance.

The runtime comparison illustrates that the proposed AICCO brings extra computation expenses in contrast to less intricate baseline optimizers. This is normal due to the fact that AICCO contains a number of improvement mechanisms and multiple local refinement operations. However, the extra expense comes with increased path feasibility, obstacle avoidance, and collision avoidance. Thus, the proposed solution is better suited for situations where satisfactory path quality and safety levels are more critical than extremely frequent replanning. In the case of high-speed real-time robots, acceleration strategies and hybrid global local implementation are needed.

The overall results show that the proposed AICCO is an improvement over the original CCO and a good optimization structure for smooth and collision-free mobile robot path planning. Validating the benchmark results show its generic optimization ability, and the static and dynamic path-planning experiments prove that it can be applied to robot navigation problems. The proposed method is very useful when several different and contradictory criteria are used to assess the quality of the path, such as path length, smoothness, and avoidance of obstacles.

6.6. Summary of Experimental Findings

Based on the experimental results, the main findings can be summarized as follows:

1. Benchmark optimization performance

On the F1–F23 benchmark function, the proposed AICCO has the best overall rank among PSO, GWO, WOA, ABC, the original CCO, and the proposed AICCO. It attained an average rank of 2.6957 and the highest mean fitness value on seven functions, which showed its good general optimization ability.

2. There is an overall improvement on the original CCO

The proposed AICCO performed better than the original CCO in the overall benchmark comparison. The proposed AICCO obtained the highest number of winning functions and the original CCO did not obtain the best mean result on any benchmark function. This verifies that the proposed improvement mechanisms are a significant improvement to the original CCO search process.

3. Static path-planning performance

The proposed AICCO was able to find a feasible path which was collision-free from SP = (5, 5) to GP = (95, 90) in the static obstacle scenario. Path length obtained was 138.838369, smoothness value obtained was 1.173386 rad, minimum clearance obtained was 0.785565, and collision penalty obtained was 0.000000. The achieved results demonstrate the capability of the proposed method to balance path length, smoothness, and safety in static environments.

4. Dynamic online replanning performance

In the dynamic scenario, the robot managed to navigate by using online replanning. The path length executed was 129.500000, minimum actual clearance was 1.514250, and collision count was 0. This validates the proposed AICCO's ability to cope with dynamically changing obstacles and ensure safe navigation.

5. Safety and obstacle avoidance

Navigation in static and dynamic experiments was collision-free. It can be observed that positive clearance values were obtained in both cases, which indicate that the generated trajectories were all outside the expanded obstacle boundaries. This shows that the adaptive obstacle expansion collision-checking strategy works well.

6. Computational considerations

The computational load of the proposed AICCO arises from the elite-guided enhancing process, the memetic local refinement process, stagnation repair, and replanning in the dynamic case. However, this added expense is well worth the greater optimization accuracy, collision avoidance, and adaptive navigation.

To sum up, the experimental results show that the proposed AICCO can be used to solve the numerical optimization problem and can be used for mobile robot path planning. The algorithm can effectively optimize compared with the original CCO, and can generate safe, smooth, and collision-free paths in static and dynamic obstacle environments.

7. Conclusions

An Adaptive Improved Cuckoo-Catfish Optimizer (AICCO) has been introduced for the smooth and collision-free path planning of mobile robots in this paper. The proposed method enhances the original CCO, utilizing chaotic opposition-based initialization, nonlinear adaptive control, strong elite preservation, memetic local refinement, and stagnation-based opposition repair. They were added to enhance the diversity of the population, sharpen the exploitation around better solutions, maintain good-quality paths, and prevent premature stagnation.

A multi-objective path-planning formulation was derived to optimize three important aspects of navigation based on the path length, path smoothness, and path crossing behavior with respect to obstacles. The objective function is proposed to be based on the length of the path, turning-angle variations over the path, and obstacle clearance through sampled points on the path. Moreover, an adaptive obstacle expansion method was applied to take the size of the robot, safety margin, uncertainty, and dynamic obstacle movement into account. In dynamic environments, an online replanning approach was used, in which the robot replans continuously from its current position, while keeping the obstacle locations up to date.

The proposed AICCO was tested with the 23 classical benchmark functions and compared with PSO, GWO, WOA, ABC, and the original CCO. Experimental results revealed that AICCO had the highest overall ranking compared with the other algorithms, and the best mean fitness value on seven benchmark functions. This validates the fact that the proposed improvement mechanisms improve the optimization performance of the original CCO.

The proposed method was also examined in static and dynamic mobile robot path planning. AICCO was able to create a collision-free path in the static environment with a collision penalty of zero and minimum clearance 0.785565. In the dynamic online replanning scenario, the robot was able to move through moving obstacles without collision, with actual clearance of at least 1.514250. These results show that the proposed method is able to produce safe and adaptive paths while balancing the length of the path, the smoothness, and the avoidance of obstacles.

Overall, the experimental results verify that the proposed AICCO is useful for numerical optimization and mobile robot path planning based on simulation. In all three test scenarios—static, dynamic, and nonlinear/reactive MATLAB simulation scenarios—the method was able to produce both safe paths and collision-free paths while minimizing path length, maximizing smoothness, and avoiding obstacles. Real-robot validation, however, was not covered in the present study. Further, the dynamic replanning time reported shows that there is still need for greater efficiency before it can be directly deployed on fast mobile robots. Future work will be directed towards enhancing the real-time performance by parallel implementation, adaptive iteration control, warm-start replanning, code-level

acceleration, and integration with faster local planners. High-fidelity robotic simulators and physical mobile robot platforms for further validation will be taken into account.

Author Contributions: J.A.A.: Writing—original draft, Visualization, Validation, Software, Methodology, Conceptualization. M.A.J.: Writing—review and editing, Validation, Conceptualization. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The MATLAB code and simulation data supporting the findings of this study are available from the corresponding author upon reasonable request.

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

Abbreviations

The following abbreviations are used in this manuscript:

ABC	Artificial Bee Colony
ACO	Ant Colony Optimization
AICCO	Adaptive Improved Cuckoo-Catfish Optimizer
APF	Artificial Potential Field
A*	A-Star Algorithm
CCO	Cuckoo-Catfish Optimizer
CHOMP	Covariant Hamiltonian Optimization for Motion Planning
D*	Dynamic A-Star
DRL	Deep Reinforcement Learning
DRL-MORIME	Deep-Reinforcement-Learning-Based Multi-Objective RIME
GP	Goal Point
GWO	Gray Wolf Optimizer
MATLAB	Matrix Laboratory
MORIME	Multi-Objective RIME
NSGA-II	Non-dominated Sorting Genetic Algorithm II
ODA	Obstacle Detection and Avoidance
PRM	Probabilistic Roadmap
PSO	Particle Swarm Optimization
RIME	Rime Optimization Algorithm
RRT	Rapidly Exploring Random Tree
RRT*	Rapidly Exploring Random Tree Star
SP	Start Point
Std	Standard Deviation
WOA	Whale Optimization Algorithm

References

1. Duan, P.; Yu, Z.; Gao, K.; Meng, L.; Han, Y.; Ye, F. Solving the multi-objective path planning problem for mobile robot using an improved NSGA-II algorithm. *Swarm Evol. Comput.* **2024**, *87*, 101576. [CrossRef]
2. Liu, S.; Tian, Q.; Tang, C. Mobile Robot Path Planning Algorithm Based on NSGA-II. *Appl. Sci.* **2024**, *14*, 4305. [CrossRef]
3. Ye, F.; Duan, P.; Meng, L.; Xue, L. A hybrid artificial bee colony algorithm with genetic augmented exploration mechanism toward safe and smooth path planning for mobile robot. *Biomim. Intell. Robot.* **2025**, *5*, 100206. [CrossRef]
4. Ye, F.; Duan, P.; Meng, L.; Sang, H.; Gao, K. An enhanced artificial bee colony algorithm with self-learning optimization mechanism for multi-objective path planning problem. *Eng. Appl. Artif. Intell.* **2025**, *149*, 110444. [CrossRef]
5. Hart, P.E.; Nilsson, N.J.; Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybern.* **1968**, *4*, 100–107. [CrossRef]
6. Dijkstra, E.W. A note on two problems in connexion with graphs. *Numer. Math.* **1959**, *1*, 269–271. [CrossRef]
7. LaValle, S.M. *Rapidly-Exploring Random Trees: A New Tool for Path Planning*; Technical Report TR 98-11; Computer Science Department, Iowa State University: Ames, IA, USA, 1998.

8. Khatib, O. Real-time obstacle avoidance for manipulators and mobile robots. *Int. J. Robot. Res.* **1986**, *5*, 90–98. [CrossRef]
9. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95—International Conference on Neural Networks, Perth, Australia, 27 November–1 December 1995; pp. 1942–1948. [CrossRef]
10. Karaboga, D.; Basturk, B. A powerful and efficient algorithm for numerical function optimization: Artificial Bee Colony (ABC) algorithm. *J. Glob. Optim.* **2007**, *39*, 459–471. [CrossRef]
11. Dorigo, M.; Gambardella, L.M. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Trans. Evol. Comput.* **1997**, *1*, 53–66. [CrossRef]
12. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2002**, *6*, 182–197. [CrossRef]
13. Wang, T.-L.; Gu, S.-W.; Liu, R.-J.; Chen, L.-Q.; Wang, Z.; Zeng, Z.-Q. Cuckoo catfish optimizer: A new meta-heuristic optimization algorithm. *Artif. Intell. Rev.* **2025**, *58*, 326. [CrossRef]
14. Guo, B.; Sun, Y.; Chen, Y. Safe path planning of mobile robot based on improved particle swarm optimization. *Trans. Inst. Meas. Control* **2025**, *47*, 1715–1724. [CrossRef]
15. Ahmad, J.; Ab Wahab, M.N. Enhancing the safety and smoothness of path planning through an integration of Dijkstra's algorithm and piecewise cubic Bezier optimization. *Expert Syst. Appl.* **2025**, *289*, 128315. [CrossRef]
16. Jo, S.; Choi, J. Enhancing robot path planning with B-Spline and PSO for obstacle avoidance. *Discov. Comput.* **2026**, *29*, 213. [CrossRef]
17. Ran, J.; Zhang, J.; Wu, F.; Ren, A.; Zhou, P.; Wang, L. DRL-MORIME: A multi-objective path planning algorithm for mobile robots with adaptive operator selection and archive optimization. *J. King Saud Univ. Comput. Inf. Sci.* **2026**, *38*, 172. [CrossRef]
18. Cao, L.; Tang, L.; Cao, S.; Sun, Q.; Zhou, G. Smooth Optimised A*-Guided DWA for Mobile Robot Path Planning. *Appl. Sci.* **2025**, *15*, 6956. [CrossRef]
19. Hahn, B. Enhancing Obstacle Avoidance in Dynamic Window Approach via Dynamic Obstacle Behavior Prediction. *Actuators* **2025**, *14*, 207. [CrossRef]
20. Schöneberg, E.; Schröder, M.; Görges, D.; Schotten, H.D. Trajectory Planning with Model Predictive Control for Obstacle Avoidance Considering Prediction Uncertainty. *IFAC-PapersOnLine* **2025**, *59*, 349–354. [CrossRef]
21. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey Wolf Optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
22. Mirjalili, S.; Lewis, A. The Whale Optimization Algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
23. Kavraki, L.E.; Svestka, P.; Latombe, J.-C.; Overmars, M.H. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Autom.* **1996**, *12*, 566–580. [CrossRef]
24. Karaman, S.; Frazzoli, E. Sampling-based algorithms for optimal motion planning. *Int. J. Robot. Res.* **2011**, *30*, 846–894. [CrossRef]
25. Stentz, A. Optimal and efficient path planning for partially known environments. In *Intelligent Unmanned Ground Vehicles*; Hebert, M.H., Thorpe, C., Stentz, A., Eds.; Springer: Boston, MA, USA, 1997; pp. 203–220. [CrossRef]
26. Koenig, S.; Likhachev, M. D* Lite. In Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI), Edmonton, AB, Canada, 28 July–1 August 2002; pp. 476–483.
27. Fox, D.; Burgard, W.; Thrun, S. The dynamic window approach to collision avoidance. *IEEE Robot. Autom. Mag.* **1997**, *4*, 23–33. [CrossRef]
28. Borenstein, J.; Koren, Y. The vector field histogram—Fast obstacle avoidance for mobile robots. *IEEE Trans. Robot. Autom.* **1991**, *7*, 278–288. [CrossRef]
29. Fraichard, T.; Asama, H. Inevitable collision states—A step towards safer robots? *Adv. Robot.* **2004**, *18*, 1001–1024. [CrossRef]
30. Zucker, M.; Ratliff, N.; Dragan, A.D.; Pivtoraiko, M.; Klingensmith, M.; Dellin, C.M.; Bagnell, J.A.; Srinivasa, S.S. CHOMP: Covariant Hamiltonian optimization for motion planning. *Int. J. Robot. Res.* **2013**, *32*, 1164–1193. [CrossRef]

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