SCIENTIFIC Publishing Limited

Journal of Intelligent Communication
https://ojs.ukscip.com/index.php/jic

Article

Hybrid Evolutionary Reinforcement Learning for UAV Path Planning: Genetic Programming and Soft Actor Critic Integrations

Muhammad Umer Mushtaq [1,2,*], Hein Venter [1], Pule Nxasana [1], Fhulufhelo Tshivhula [1], Katleho Junior Modise [1], Tamoor Shafique [3] and Owais Muhammad [4]

[1] Department of Computer Science, University of Pretoria, Pretoria 0028, South Africa
[2] Department of Computer Science, Bahria University Islamabad Campus, Islamabad 44230, Pakistan
[3] School of Digital Technology, Innovation and Business, University of Staffordshire, Stoke-on-Trent ST4 2DF, UK
[4] Department of Electrical and Electronic Engineering, University of Johannesburg, Johannesburg 1619, South Africa
[*] Correspondence: mu.mushtaq@up.ac.za

Received: 9 July 2025; **Revised:** 12 August 2025; **Accepted:** 16 August 2025; **Published:** 2 September 2025

Abstract: Unmanned Aerial Vehicle (UAV) path planning in unknown environments continues to pose a significant challenge, as Deep Reinforcement Learning (DRL) solutions are often severely hampered by slow convergence rates as well as unstable training dynamics. To address this gap, we introduce a Genetic Programming–seeded Soft Actor–Critic (GP+SAC) approach in which Genetic Programming produces high-quality trajectories that are introduced into the replay buffer of SAC as a "warm-start" policy to prevent wasteful early exploration. Through experiments in three benchmark grid environments, we demonstrate that GP+SAC converges significantly more rapidly than the FA-DQN baseline, achieving superior returns in fewer episodes while capitalizing on the same reward design. We show that in large environments, GP+SAC achieved a mean path length of 30.55 units as compared to FA-DQN's 28.38, thus validating that rapid convergence has no tradeoff in path efficiency. Observably, results also show that as much as GP+SAC obtains superior cumulative rewards, there is a visible fluctuation in the level of training that is indicative of instabilities under very constrained environments. Numerical evaluations show that the proposed GP+SAC agent converges significantly faster than the FA-DQN baseline, achieving higher episodic returns within only a few episodes. In terms of path efficiency, GP+SAC yields an average path length of 30.55 units, which is comparable to the FA-DQN's 28.38 units, demonstrating that accelerated convergence is achieved without sacrificing path optimality.

Keywords: UAV-Assisted WSNs; UAV Flight Path Scheduling; Soft Actor-Critic (SAC); Reinforcement Learning; Genetic Programming

1. Introduction

Integration of Unmanned Aerial Vehicles (UAVs) and Wireless Sensor Networks (WSNs) is a revolutionary solution for precision farming, facilitating effective data collection, environmental observation, and on-demand decision making [1]. For extensive agricultural applications, WSNs encounter energy-limited sensor nodes, extensive communication distances, and environmental limitations that discourage long-term operation. UAV-aided WSNs overcome them by decreasing ground node energy consumption, enhancing coverage, and facilitating flexible on-demand data collection. Nevertheless, optimal performance is realized only when attention is given to UAV flight

path planning, energy optimization, and fault-tolerant communication mechanisms [2].

Latest studies point out the promise of machine learning-based techniques, specifically Reinforcement Learning and metaheuristic optimization, to overcome these challenges. As for classic path planning methods such as A, RRT, and Dijkstra, although providing deterministic solutions, these methods fail to be responsive to non-static scenes and cannot optimize multiple criteria such as data throughput, collision evasion, and energy [3]. By integrating RL and Genetic Programming, hybrid methods promise the production of good initial solutions and adapt them through iterative learning, leading to smoother, shorter, and adaptable trajectories for UAVs.

We propose a hybrid system comprising a Genetic Programming (GP) and a Soft Actor-Critic (SAC) system for energy-optimal data collection via UAVs for agricultural Wireless Sensor Networks (WSNs). The GP part is employed offline to find near-optimum waypoint sequences with obstacle and smoothness constraints. The resulting paths are used to initialize the SAC component's replay buffer, accelerating convergence and end-path quality under dynamic conditions. The system was evaluated using custom Python environments (map1, map2, and map3) [4], and its performance was tested against the FA-DQN framework in terms of convergence speed, path length, and qualitative path characteristics such as smoothness.

2. Literature Review

A hybrid deep neural architecture combining Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks has been proposed to solve UAV path planning in dynamic environments [5]. The CNN extracts spatial features from aerial images, and the LSTM extracts temporal relations to make decisions in sequences. The system learns to cope with real-time obstacle and terrain changes. The model is also trained in simulated environments on sequences of images and compared against A3C and DQN baselines. It demonstrates higher path efficiency, fewer collisions, and quicker convergence. The approach improves UAV navigation by using spatial perception and memory. Performance shows robust generalization on unseen dynamic scenes. Real-world validation and energy-awareness are, however, not present. The system also does not consider swarm-level coordination or flight [5].

A reinforcement learning (RL)-based framework has been proposed to improve routing in Flying Ad-Hoc Networks (FANETs), addressing the shortcomings of conventional routing algorithms in highly dynamic UAV networks [6]. The authors formulate a multi-objective optimisation problem that jointly considers UAV trajectories, energy efficiency, and communication range to enhance packet delivery ratio (PDR) and minimise transmission delays. The solution suggests the deployment of a Data Forwarding Agent (DFA) per UAV to make next-hop decisions based on real-time environmental feedback, sensor data generation models, and cooperative UAV networking. The system allows for probabilistic modelling of data generation (normal and exponential distributions) and UAV mobility, enabling adaptive, context-aware routing. The RL-based DFA is acquired using Q-learning with an ε-greedy exploration strategy, which allows UAVs to learn and adapt their forwarding policies from experience replay and reward functions that trade successful delivery, energy consumption, and delay [7]. Comparative simulations prove that the proposed method outperforms greedy, random, and no-forwarding methods in minimizing delay and maximizing PDR while maintaining computational feasibility for real-time implementation. Assessed in variable coverage radius scenarios, the framework achieves PDR above 0.95 and significantly smaller delays than baseline methods. The article highlights the applicability of RL in UAV-aided WSNs in fields such as environmental monitoring, disaster relief, and public safety communication, with scalability, flexibility, and low-latency decision-making being the key benefits. Future research directions include multi-agent coordination, deep reinforcement learning (RL) algorithms, and the integration of realistic hardware experimentation to enhance robustness and performance [7].

Optimal path planning for large agricultural WSNs is realized via a scalable Q-learning (QL)–based multi-UAV data-gathering approach [7]. The researchers overcome major limitations of single UAV systems: low coverage, energy inefficiencies, and lack of fault tolerance. The researchers present QLUR, i.e., a three-stage approach consisting of sorted k-means clustering of sensor nodes, optimal positioning of data Collection Points (CPs), and QL-based optimal-path computation framed as a Traveling Salesman Problem (TSP). The approach prioritizes residual energy, priority of CPs, and UAV distance from the corresponding sensor nodes in the reward function to allow for resilient and adaptive decision-making. The system has a fault-tolerant scheme to replace CHs and task redeployment on UAV failure. The QLUR system, with extensive NS-3 simulations, is compared with GA and ACO. The results indicate that four UAV deployments yield a 67% reduction in mission completion time and an 80% gain in resid-

ual energy compared with single UAV systems. The approach outperforms others in terms of inference speed and memory efficiency with balanced CP distribution and task accomplishment with failures. The limitations include the assumption of uniform terrain and the fixed assignment of CPs. The future extensions include dynamic routing with obstacles, integration with zone priorities, and edge computing [7].

A lightweight, real-time UAV path planning approach is presented, which avoids collisions using a mathematically guaranteed safety margin and models obstacles as rectangles [8]. Diverging from numerous reinforcement learning (RL) or neural network-based solutions, this is a graph-based approach with the configuration space reduced by the replacement of grid-based tessellations with well-placed interest points near corners of the obstacles, followed by the generation of safe paths through the Dijkstra algorithm. This significantly reduces the computational burden, making it eligible for execution in real time. Simulation results with the UAV Toolbox for MATLAB verify the proposed approach, generating shorter, linear, and collision-free paths even for dense obstacles. Diverging from the possibilities of RL-based solutions to exhibit unpredictable behaviour or to need detailed training, this has mathematically specified guarantees, along with tolerating sensor uncertainty through a threshold-based "disk" safety model. Nonetheless, the UAV kinematics are not included within the scope of the model yet. Reducing memory usage as well as achieving quick convergence remains the focus of the paper as compared to techniques involving Artificial Potential Fields (APF), PRM, or RRT, etc. The paper further provides performance information on the basis of UAV turning angles along with speed deviations. The simplicity, although, remains the strength of the approach, with it being excellent for robustness along with the feasibility of executing it within a short time, even though it doesn't take into account moving obstacles or energy-aware planning [8].

More recent work in the field of deep neural policies places it as a more compelling route and thus a likely direction for the future of navigating dynamic partially observable environments. Convolutional Neural Networks (CNNs) extract spatial structure from aerial images while Long Short-Term Memory (LSTM) layers encode temporal dependencies, which then enables a unified perception-to-action stack that thus reacts to terrain and obstacle nonstationarities across sequences [5]. After having been evaluated against other baseline methods such as A3C and DQN, these CNN-LSTM systems reported higher path efficiency, fewer collisions and faster convergence, and better generalization to unseen environments. The strengths mentioned here, although enticing, are confined to simulations; moreover, energy budgeting and flight endurance are not modelled, latency and sensing are assumed ideal, and no mechanisms are provided for inter-UAV coordination or communication bottlenecks [5]. The observed gaps seem to reflect the broad survey observations that learning-based planners are prone to neglect energy-aware decision-making, communication borders, and safety analysis aimed at certification [8]. In more practical deployments and missions, these omitted aspects of energy-efficient path planning matter; energy state and harvesting potential modulate feasible trajectories and loitering policies, while communication reliability and bandwidth shape when and where data can be offloaded. Emerging directions therefore argue for augmenting such hybrids with (i) explicit energy models and harvesting-aware objectives [9], (ii) cognition/safety layers that wrap the learned policy with constraint monitors and human-in-the-loop overrides [10], and (iii) channel-aware planning that anticipates link quality variations along the route [11], aligning perception–memory strengths with field constraints. A second cluster focuses on the learning landscape itself. DRL-PP modifies DQN with a cubic reward to sharpen action discriminability, a double network structure for stability, and an ϵ-greedy schedule to balance exploration and exploitation, improving success rates and reducing steps on grid maps relative to vanilla DQN and Actor–Critic [12].

Complementary work tackles sparse rewards and deceptive local optima by (a) shaping rewards with cumulative obstacle density and goal distance and (b) segmenting the environment into connected regions to avoid dead ends, producing faster convergence with both discrete (DQN) and continuous (DDPG) action spaces [13]. Collectively, these demonstrate that geometry-aware shaping and structural priors can regularize DRL training without heavy architectures. But even here, real-world blockers persist: energy constraints, vehicle kinematics, and hardware trials are typically deferred [12,13]. Surveys echo the need for standardized metrics, GPS-denied robustness, and communication-aware, energy-aware formulations, as opposed to planner-only benchmarks [14]. Pushing toward deployment will likely require coupling such shaping with energy-harvesting-conditioned objectives [15], verifiable safety envelopes around learned policies [10], and channel-forecast-informed exploration/exploitation decisions [11,16].

When UAVs act as networked agents, routing and trajectory planning are coupled. A Q-learning Data Forwarding Agent (DFA) for FANETs learns next-hop policies that jointly optimize PDR delay and energy by using proba-

bilistic models of traffic and mobility; simulations report PDR > 0.95 and lower delays than greedy/random forwarding across coverage scenarios [17]. This reframes the objective: where to fly cannot be separated from what to forward and when. It also surfaces the role of channel predictability and network capacity. Here, two recent threads are complementary: (I) EMD-empowered neural predictors for non-stationary air-to-ground channels provide spatial-temporal link forecasts to anticipate fading/handovers [11], and (II) capacity/deployment analyses quantify spectral-efficiency/coverage trade-offs to guide altitude, placement, and fleet sizing decisions [11]. Folding these into learning (reward terms, constraints, or hierarchical policies) connects the DFA idea [18] to path planning under real communication risk, while survey work underscores the need for common evaluation settings that include communication load, interference, and scheduling—not just geometry.

In large Agricultural WSNs, the structure helps in restricting the DRL search space. Meanwhile, QLUR breaks down the mission into sorted *k*-means grouping of sensors, placing the data Collection Points (CPs) at optimal places and a Q-learning tour (TSP Framing) that has a reward that balances between node energy, CP priority, and the UAV travel cost. NS-3 results are clear. Four UAVs shorten mission time by 67 percent and raise residual energy by 80 percent when compared with a single UAV. The system also swaps failed CHs and redeploys tasks when needed [19]. Deterministic planners use a different idea. They limit the space to the corner points and run Dijkstra with a disk margin. The outcome is linear, safe paths that need little memory and run in real time. They do not handle moving obstacles or energy limits [20]. BSPOP goes further by encoding inputs as B-splines. That lowers the number of variables, trims constraints, and keeps paths smooth [21]. Together, the methods point to three styles: network-first, safety-first, and compute-first. The lesson is simple. Cluster tasks, add safety guards, and use smooth control to cut compute costs. Application studies in precision agriculture highlight the role of temporal memory and task-aware rewards. BL-DQN adds a Bi-LSTM head and links the reward to coverage, guidance, collision, and efficiency [22]. A U-Net is used for segmentation. Across six farm trials, coverage went up by a little over 41 percent, redundancy dropped compared with DQN and DFS, and training ran more steadily [22]. BiLG-D3QN builds on this by joining Bi-LSTM with Bi-GRU inside a duelling DQN. It also brings in payload energy and recharge stations as part of the model [23].

On 15×10 grids, coverage gains ranged from about 12–22%, and redundancy fell to around 2.5%, giving it an edge over DDQN, D3QN, Duelling DQN, A* (a technique/algorithm), and PPO [22]. Evaluations across ground vehicles, manipulators, and UAVs report that Double-DQN, SAC, Rainbow, and RL, combined with optimizers such as DQN plus PSO, improve tracking and adaptability. The edge devices' computational time is heavy, and many policies that would work in other simulations start to fail when doing real tests [24]. Classical hybrids remain useful. GA plus QuickNav is one example, where waypoint order is evolved and deterministic saving is enforced [23]. This setup beats A*, RRT, and ACO in both distance and time, and it has been tested on UAVs, though only at fixed height with static obstacles, without any energy or swarm features [23]. Surveys of WSN routing point to duty cycling, hierarchical or data-centric designs, blockchain trust, and federated learning [20]. They find that RL-adaptive MAC and routing perform better than static ones when the load shifts, but UAV swarm integration has not been proven in practice [25]. One crop recommendation study shows how models, once validated, were delivered through mobile and web tools. That step linked research with real farm use. DRL-based UAV systems should then follow the same path [26]. Mission results depend on timeliness and lifetime, not only path length. One design joined clustering with routing and used Monte Las Search with Triangle Path Optimization. Lifetime went up by around 360%. UAV path length dropped by about 56% [27]. GA parameters were tuned with RL. Extra schemes were added to guard emergency data and handle replenishment [27]. DRL families such as DQN, DDPG, SAC, and multi-agent models also help by adjusting trajectory, scheduling, and energy in random settings. Training load is still heavy. However, edge limits are still considered a problem [28].

Across the studies, three points keep showing up. Memory and perception with CNN, LSTM, Bi-LSTM, and GRU let systems react better to change. Then structure-based steps like reward shaping, segmentation, clustering, and CP placement shrink the search space and speed training. Mission-aware goals tied to PDR, delay, AoI, and lifetime give stronger results. To move from simulation to field use, energy harvesting and storage must be part of the state and the objectives. That way, policies adjust to gaps in power. Safety checks also need verifiable limits. Planning then has to factor in channel forecasts and capacity so the system stays rate-aware and avoids interference. Multi-UAV setups need bandwidth-aware coordination, CP re-optimization, and fair task division. This keeps the good parts of older methods while closing gaps in energy, safety, and communication for UAV WSNs.

3. Methodology

3.1. System Overview

This section outlines the design, architecture, and implementation of the proposed Genetic Programming (GP) and Soft-Actor-Critic hybrid framework specifically for UAV path planning in complex and dynamic environments. This proposed method is composed of the following three stages: (I) an offline GP module which has been implemented primarily to evolve high quality initial waypoints and paths, (II) an online SAC module that will be optimized for continuous control and finally (III) a hybrid integration strategy which makes use of the high-quality GP derived individuals to seed SAC's replay buffer and accelerate convergence while producing shorter, smoother and more efficient paths.

3.2. Environment and Problem Formulation

The UAV operates in a two-dimensional discrete grid world that is represented as a binary occupancy matrix as follows:

$$G \in \{0, 1\}^{M \times N} \tag{1}$$

Where
$G_{ij} = 0$ denotes a free cell and $G_{ij} = 1$ denotes an obstacle.
The UAV's position at time step t is defined as

$$\mathbf{x}_t = (r_t, c_t) \in Z^2 \tag{2}$$

with

$$0 \leq r_t < M, \; 0 \leq c_t < N \tag{3}$$

The start and goal positions are given by

$$x_0 = \mathbf{s}, \; \mathbf{x_t} = \mathbf{g} \tag{4}$$

where T is the episode length.

3.2.1. Action Space

Let the UAV action space A be a set of movement commands, which have been defined as continuous:

$$A \subset \mathbb{R}^2 \tag{5}$$

in which each action defines a velocity vector (Δ_r, Δ_c)
At every time step, an action $at \in A$ updates the state according to:

$$\mathbf{x_{t+1}} = \mathbf{x_t} + a_t \tag{6}$$

subject to collision constraints.

3.2.2. Collision Constraint

A collision occurs if the location of the UAV intersects an obstacle or if the straight-line segment from one waypoint to the next intersects any obstacle cell:

$$\text{Collision}(xt, xt+1) = \begin{cases} 1, \; \exists (i,j) \in \text{Bresenham}(\mathbf{x_t}, \mathbf{x_{t+1}}) \mid G_{i,j} = 1, \\ 0, \; \text{otherwise.} \end{cases} \tag{7}$$

Here, Bresenham $(\cdot)$ returns all grid cells along the straight path segment between $\mathbf{x}_t$ and $\mathbf{x_{t+1}}$.

3.2.3. Objective Function

The main objective is to minimize the path cost:

$$J(\pi) = \alpha \cdot L(\pi) + \beta \cdot C(\pi) - \gamma \cdot S(\pi) \tag{8}$$

where:

- $L(\pi)$: Overall path length provided by

$$L(\pi) = \sum_{t=0}^{t-1} \| X_{t+1} - X_t \| \tag{9}$$

that is the overall Euclidean distance covered in the trajectory $\pi = (x0, x1, \ldots, xT)$.

- $C(\pi)$: Collision Cost Defined as

$$C(\pi) = \sum_{t=0}^{T} \mathbb{I}[x_t \in \mathcal{O}] \tag{10}$$

where $\mathbb{I}[\cdot]$ is the indicator function and $\mathcal{O}$ is the set of cells that contain obstacles. It punishes trajectories that cross through obstacles.

- $S(\pi)$: smooth term that

$$S(\pi) = \sum_{t=1}^{T-1} \cos(\theta_{t+1} - \theta_t) \tag{11}$$

where θ_t represents the UAV heading angle at time t. It advocates smooth turns and does not favour abrupt changes of direction.

Here, the scalar weights α, β, and $\gamma > 0$ are used to balance the contribution of path length, collision avoidance, and smoothness.

3.3. Genetic Programming Module

The Genetic Programming (GP) module is designed to operate as an offline phase to generate path individuals. The paths are in the form of near-optimal waypoint sequences from some initial position to a goal position, avoiding both static and dynamic obstacles. Additionally, each GP individual encodes a variable-length sequence of waypoints to allow fine-tuning to the environment.

$$P = \{w_1, w_2, \ldots, w_k\},\ w_i \in \mathbb{R}^2 \tag{12}$$

where each waypoint is located in obstacle-free space and respects the safety distance d_{safe}.

3.3.1. Population Initialization

The initial population of N_{pop} individuals is generated by uniformly sampling waypoints within free space, ensuring:

$$d(w_i, o_j) \geq d_{safe}, \forall i, j \tag{13}$$

where O_j denotes obstacle *j*.

3.3.2. Fitness Function

Each individual's fitness F is computed as a weighted sum of multi-objective criteria:

$$F(P) = \omega_L L(P) + \omega_A A(P) + \omega_C \text{ Colli}(P) \tag{14}$$

where:

$$L(P) = \sum_{-1}^{k-1} \|w_{i+1} - w_i\| \tag{15}$$

This term represents the total Euclidean distance travelled along the path, computed by summing the straight-line distances between consecutive waypoints.

$$A(P) = \frac{1}{k-2} \sum_{i=2}^{k-1} \theta_i \tag{16}$$

This term is the mean change in heading direction at intermediate waypoints, measuring the average turning angle across the path.

$$Colli(P) = \sum_{t=1}^{k-1} i \tag{17}$$

Is the number of path segments intersecting with the obstacles, for which ⊮ is an indicator function equal to 1 if there is a collision, otherwise it's 0. Here,

Θ_i is the heading change at the i^{th} waypoint. In this method, lower values indicate superior path quality, and as a result, the GP aims to minimize this function. For every individual path, the function evaluates the three performance measures described here, namely: total path length L(P), average turning angle A(P), and finally and collision count Colli(P). Ultimately, GP chooses the solutions with the lowest fitness values, ensuring shorter, smoother, and collision-*free* paths.

3.3.3. Genetic Operators

The GP loop consists of:

1. Selection: Tournament selection with size t_s.
2. Crossover: One-point crossover exchanging waypoint subsequence.
3. Mutation: Gaussian perturbation of waypoint coordinates, with re-sampling if constraints are violated.

The evolutionary algorithm proposed in this paper works in an iterative loop consisting of selection, crossover, and mutation operations. Selection is performed using a tournament selection strategy, in which a tournament size, t_s, of individuals is stochastically chosen from the population that the algorithm is currently exploring. The solution with the lowest fitness is then chosen as a parent for future offspring. This method maintains selection pressure for better quality solutions while also preserving genetic diversity. Crossover occurs as a one-point crossover operator in which a single waypoint in a path (sequence of waypoints) is selected as the crossover point, with the waypoints before this being inherited from one parent and the remainder from the other. This process is effective in that it enables the combination of advantageous sub-paths from different individuals, thus potentially producing superior offspring. Finally, mutation is applied intrinsically through the addition of random way-point coordinate noise, in which each selected waypoint is shifted by a specific value that has been drawn from a normal distribution with zero mean and variance. Additionally, if a waypoint is seen to violate constraints that have been set up, such as entering an obstacle region or leaving the operational bounds, it is then resampled until feasibility is restored. Ultimately, the operators implemented in this work aim to guide the population toward shorter, smoother, and collision-free paths over successive generations.

3.3.4. Evaluation Loop

The Genetic Programming method in this work begins with the random initialization of an initial population of size N_{pop}, whereby each individual is represented by a sequence of waypoints that form a path. Each individual is evaluated via its fitness; the solutions with the lowest fitness values are chosen to be parents, mutation and crossover are applied, and new offspring are generated. This process occurs for either $G_{\max}$ or n number of times to produce the best quality paths. **Algorithm 1** depicts this entire process in great detail.

More specifically, after the generation of a population, each path is evaluated with the function F(P), which quantifies path quality based on length, smoothness, and collision avoidance. Following this, parents are selected through tournament selection, and variation is introduced by the application of mutation and crossover. These

operators then allow for the generation of new solutions that replace the current population to produce the next generation. The process continues iteratively until either the maximum number of generations is reached or no improvement in the best fitness value is observed for a predefined number of consecutive generations. Finally, upon completion, the GP-evolved paths ranked in ascending order of fitness are ready for integration into the SACs replay buffer.

ALGORITHM 1: GENETIC PROGRAMMING (GP) FOR UAV PATH PLANNING

Input: Population size N, max generations G_{max}, stall limit n_{stall}, tournament size t_s, crossover rate p_c, mutation rate p_m, mutation scale σ , bounds B , obstacles O , weights (w_L, w_A, w_C).

Output: Top N_{elite} paths ranked by ascending fitness F(P).

Initialize: Random population P_0 of feasible waypoint points; evaluate F(P)

For each $g \leftarrow 0$, stall $\leftarrow 0$, $F_{best} \leftarrow \min F(P)$

While $g < G_{max}\ and\ stall < n_{stall}\ do$

$g \leftarrow g + 1,\ P_g \leftarrow \emptyset$ **while** $|P_g| < N$ **do**

Selection: $p1 \leftarrow$ TOURNAMENTSELECT($P_g - 1, t_s$), p2$\leftarrow$ TOURNAMENTSELECT ($P_g - 1,\ t_s$)

Crossover: if rand$< p_c$ then $(c_1, c_2)\leftarrow$ ONEPOINTCROSSOVER ($p_1,\ p_2$) else copy parents

Mutation: if rand $< p_m$ apply Gaussian perturbation to waypoint coord, repair if outside B or in O

Evaluate $F(c_1),\ F(c_2)$ and append to P_g (truncate if needed)

$F_{curr} \leftarrow \min_{P \in P_g} F(P)$ **if** $F_{curr} < F_{best}$ **then**

$F_{best} \leftarrow F_{curr},\ stall \leftarrow 0$

else

$stall \leftarrow stall + 1$

EvaluateFitness ($P = (w_1, \ldots, w_k)$) $L \leftarrow \sum_{i=1}^{k-1} ||w_{i+1} - w_i||$; $A \leftarrow \frac{1}{k-2}\sum_{i=2}^{k-1} \theta_i$;

$Colli \leftarrow \sum_{i=1}^{k-1} 1\{(w_i, w_{i+1}) \cap O \neq \emptyset\}$ **return** $\omega_L L + \omega_A A + \omega_C Colli$

3.4. Soft Actor–Critic Module

The Soft-Actor-Critic module (SAC), in contrast to the GP module, is an online learning phase that is designed to optimize continuous UAV control for real-time path refinement and adaptation to more dynamic environments. In this phase, the UAV navigates to and from the defined initial and goal positions, but it now operates within a continuous state-action space, which allows fine-grained control. The proposed design also enables SAC to now not only follow the structured trajectories that have been seeded from the GP-derived solutions, but also adapt to

environmental changes, including moving obstacles or altered mission parameters, which also reinforce both strategic and reactive path planning methods.

3.4.1. State and Action Spaces

State space for the SAC module is built to encapsulate all relevant spatial, kinematic, and environmental data required for effective UAV navigation. The state vector s_t, at each time-step t, contains the current Cartesian location (x_t, y_t) and heading θ_t, velocity v_t , the Euclidean distance to the goal d_{goal} , and flags for collision proximity to obstacles. Additional features, relative bearing to the goal, and most recently enacted control input, are added to provide temporal information

The action space is continuous and two-dimensional, defined as

$$a_t = (\Delta v_t, \Delta \theta_t) \tag{18}$$

where Δv_t represents the change in translational speed and $\Delta\theta_t$ the change in heading angle. Actions are constrained such that

$$v_{min} \leq v_t \leq v_{max}, |\Delta\theta_t| \leq \theta_{max} \tag{19}$$

guaranteeing compliance with UAV kinematic constraints. With this description, SAC can produce smooth and dynamically plausible motion commands, facilitating fine-grained manoeuvring in the continuous control space while still being consistent with the waypoint sequences resulting from the GP, employed for seeding the replay buffer.

3.4.2. Reward Shaping

SAC's reward function is designed, by construction, to balance goal-driven progress, efficient paths, smooth motion, and safety. The agent, at each time step *t*, is given a shaping reward correlated with the decrease in Euclidean distance to the goal, which incentivizes persistent progress proceeding in the correct direction:

$$r_{progress} = \alpha\,(d_{t-1} - d_t) \tag{20}$$

where d_t is the current distance to the goal and $\alpha > 0$ regulates the reward for moving forward. When the goal is reached, a final success reward R_{goal} is received to reward successful task accomplishment:

$$r_{goal} = R_{goal} \cdot \mathbb{I}_{goal} \tag{21}$$

Conversely, if a collision occurs, a large negative penalty is applied:

$$r_{collision} = -R_{collision} \cdot \mathbb{I}_{collision} \tag{22}$$

This term avoids unsafe paths by strongly discouraging collisions against obstacles. For smooth paths, a penalty term for angles is introduced:

$$r_{smooth} = -\beta\ |\Delta\theta_t| \tag{23}$$

where $\beta > 0$ penalizes hard turns and sudden changes in heading. The total reward is then given by:

$$r_t = r_{progress} + r_{goal} + r_{collision} + r_{smooth} \tag{24}$$

These parameters, namely α, β, R_{goal}, and $R_{collision}$, are found via empirical tuning to be a balance between exploration and exploitation. As the SAC module is provided with high-quality trajectories from the GP module, the reward function is constructed to preserve the strengths of these seeded paths while enabling SAC to be adaptable to dynamic environmental variations, e.g., moving obstacles or changes to mission constraints. The shaping guarantees that the policy learned is not only collision-free and goal-oriented but also smooth and dynamically viable.

3.4.3. Policy Learning

The Soft Actor-Critic (SAC) algorithm optimizes a stochastic policy by maximizing both the expected return and the entropy of the policy, encouraging exploration while seeking high-value actions. The entropy-regularized policy objective is defined as:

$$J_\pi = \sum_{t=0}^{\infty} E_{(s_t,a_t)\sim\mathbb{D}}\left[Q\,(s_t, a_t) - \alpha \log \pi\,(a_t|s_t)\right] \tag{25}$$

Here, $Q\,(s_t, a_t)$ is the state-action value estimate. The expression $-\alpha\, log\, \pi(a_t \mid s_t)$ is the entropy regularization term, where the temperature parameter α balances the trade-off between maximizing the return and maintaining policy entropy. High entropy implies more exploration, while low entropy results in the policy being more deterministic. SAC utilizes two Q-functions Q_{ϕ_1} and Q_{ϕ_2} in order to deal with overestimation bias possible in value-based techniques. Each of the Q-functions is updated by minimizing the Mmean-SquaredBellman error:

$$J_Q(\phi_i) = E_{((s_t,a_t)\sim D)}\left[\frac{1}{2}\left(Q_{(\phi_i)}(s_t,a_t) - y_t\right)^2\right],\ i \in \tag{26}$$

The target value y_t is calculated as:

$$y_t = r_t + \gamma E_{a_{t+1}\sim\pi}\left[\min_{i=1,2}\overline{Q_{\phi_i}}(s_{t+1},a_{t+1}) - \alpha\log\pi(a_{t+1}|s_{t+1})\right] \tag{27}$$

Here, $\overline{Q_{\phi_i}}$ is the target network or a slow-moving copy of the Q-functions. The min operator prevents overestimation by returning the lesser of the two Q-values. $r_t$is the environment's immediate reward, and γ is the discount factor governing the weight of future returns. The policy π_θ is updated in such a way as to maximize the expectation of the Q-function with the extra entropy term:

$$J_\pi(\theta) = E_{s_t\sim\mathbb{D},a_t\sim\pi_\theta}\left[\alpha\log\pi_\theta(a_t|s_t) - Q_{\phi_1}(s_t,a_t)\right] \tag{28}$$

This update promotes policies that select actions with high Q-values while retaining sufficient randomness for exploration. Instead of fixing the temperature α, SAC learns it automatically to maintain a desired target entropy $\mathcal{H}_{target}$ using the objective:

$$J(\alpha) = E_{a_t\sim\pi_t}\left[-\alpha\left(\log\pi_t(a_t|s_t) + \mathcal{H}_{\text{target}}\right)\right] \tag{29}$$

By optimizing this loss, the algorithm dynamically adjusts α so the policy does not become overly deterministic too soon or remain excessively random.

3.5. Hybrid GP+SAC Integration

The hybrid combination of Genetic Programming (GP) with Soft Actor-Critic (SAC) is created to merge GP's symbolic optimization strengths with SAC's continuous control learning. It is aimed to provide SAC with a warm start by initializing it with high-quality, human-interpretable strategies created with GP, yet still allowing exploration to facilitate adapting to changing environments. First, GP is used in the role of the offline high-level planner. With a population of candidate navigation programs, GP adapts solutions by performing genetic operators like selection, crossover, and mutation, and optimizing for measures like goal reachability, avoidance of obstacles, and kinematic feasibility. GP's output is a group of waypoint sequences $\{w_1, w_2, \dots, w_N\}$ or control rules for safe navigation in the environment.

3.5.1. Replay Buffer Seeding

The top-N highest GP-evolved paths, sampled in proportion to their fitness for reachability of the goal, avoidance of obstacles, and kinematic feasibility, are transformed into complete transition tuples of the form(s_t, a_t, r_t, s_{t+1}). Each path is simulated with a low-level tracking controller to determine the actual control inputs $a_t$necessary itomove the UAV from the current state s_t to the next in the sequence of waypoints, with the environment returning the next-time-step reward r_t and next-state transition s_{t+1}. These high-quality transitions are then preloaded into SAC's replay buffer $\mathcal{D}$ prior to training initiation. In this way, the initial updates of SAC are biased in favour of the safe and successful paths, drastically reducing the probability of early-stage policy breakdown and accelerating the efficiency of the samples in the exploration phase.

3.5.2. Training Loop

The learning procedure commences with initialization, for which the replay buffer of SAC $\mathcal{D}$ is pre-seeded with transitions of the top-ranked GP-generated trajectories. The pre-seeding of such samples gives the agent initial experience of safe and successful navigation policies, thereby decreasing the necessity for suboptimal random exploration during the initial training phases. Provide a concise and precise description of the experimental results, their interpretation, as well as the experimental conclusions that can be drawn. Once the environment has been initialized, the interaction with the environment occurs in time steps. At each t, the agent selects an action at $a_t \sim \pi_\theta(\cdot \mid s_t)$ drawn from the stochastic policy and acts in the environment, which returns the reward r_t and next

state s_{t+1}. The observed transitions (s_t, a_t, r_t, s_{t+1}) are stored in $\mathcal{D}$,alongside the GP-seeded samples. Mini-batches are then drawn from this buffer to update the twin Q-networks Q_{ϕ_1}, Q_{ϕ_2} and the policy π_θ.

As you can see in the following algorithm, this loop also involves tuning the temperature parameter α in order to achieve the target entropy.

Finally, an exploration–exploitation balance is enforced in early training by keeping a ratio ρ of GP-seeded to newly collected samples when forming mini-batches. This ratio is gradually decreased, allowing SAC to transition from relying on GP-derived behaviors to learning entirely from its own experience. **Algorithm 2** represents the Hybrid GP + SAC training framework.

	Algorithm 2: hybrid GP+SAC TRAINING framework
1	**Input:** Number of elite GP solutions N_{elite}, GP fitness criteria, SAC, parameters θ, ϕ_1, ϕ_2, α, initial speed ratio ρ
2	**Output:** Trained SAC policy π_θ
3	**Phase1 –GP Solution Generation**
4	Run Genetic Programming to evolve candidate navigation policies.
5	Evaluate policies on fitness: goal reachability, obstacle avoidance, kinematic feasibility
6	Select Top N_{elite} solutions
7	**Phase 2: Replay Buffer Seeding**
8	Convert selected GP solutions into waypoint sequences $\{w_1 \dots w_N\}$
9	Simulate waypoint tracking to generate transitions (s_t, a_t, r_t, s_{t+1})
10	Pre-load transitions into SAC replay buffer D
11	**Phase 3 – Hybrid Training Loop**
12	while training not converged do
13	Observe current state s_t
14	Select action $a_t \sim \pi_\theta(\|s_t\|)$ and execute in environment
15	Receive reward r_t and next state s_{t+1}
16	Store $(s_t, a_t r_t, s_{t+1})$ in D
17	Sample mini-batch from D with fraction ρ GP-seeded samples
18	Update $Q_{\phi 1}$, $Q_{\phi 2}$ via Bellman error minimisation
19	Update policy π_θ via SAP entropy- regularized objective
20	Adjust temperature α towards entropy
21	Decay ρ to shift learning toward self generated experience

3.6. Experimental Parameters and Settings

3.6.1. GP Parameters

Table 1 displays the parameters and respective values that were selected for the algorithm. These settings were carefully chosen to balance exploration, exploitation, and computational efficiency.

Table 1. GP Parameter Settings.

Parameters	**Values**
Population Size	240
Generations (Max)	1
Crossover Rate (PC)	0.65
Mutation Rate (PM)	0.6
Selection Method	Tournament Selection
Tournament Size	5
Elitism (Top individuals)	10
Stall Limit	20
Waypoint Length (Min, Max)	6, 18
Mutation Sigma	1.2
Mutation Probability per Gene	0.4
Waypoint Insert Probability	0.25
Waypoint Delete Probability	0.25
Fitness Weights (Ω_L, Ω_A, Ω_C)	(1.0, 0.5, 50.0)

The GP module in this work was configured with carefully chosen parameters to balance the exploration of diverse way-points with the exploitation of promising paths. A population size of 240 was selected to ensure sufficient genetic diversity while also keeping computational overhead at a minimum, thus enabling a broader search space for the solution to be found across generations. Furthermore, the algorithm was run for a maximum of 120 generations with a stall limit implemented to enable early stopping if no improvement is found across 20 generations, thus preventing wasted computation. A higher crossover rate of 0.65 was used to encourage the combination of useful information from different individuals while introducing genetic diversity. A mutation rate of 0.6 ensured adequate variation and the possibility of escaping local minima. Tournament selection with a size of 5 was also applied as

the parent selection mechanism, not only introducing selection pressure but also maintaining diversity within the population. Waypoint path length was also constrained to a limit of 6–18 to prevent long and overly short paths. Collectively, these parameters were justified by the requirement to produce paths that are high-quality, short, and thus energy efficient.

3.6.2. SAC Parameters

Table 2 shows how the Soft Actor-Critic (SAC) component was parameterized to capture task-specific UAV navigational requirements as well as theory-motivated continuous control fundamentals.

Table 2. Environment and Reward Parameter Settings.

Parameters	Values
Progress Scaling (α)	1.0
Smoothness Penalty (β)	0.05
Success Reward (R_{goal})	150.0
Collision Penalty ($R_{collision}$)	−250.0
FA-DQN Goal Reward (Baseline-paper)	10.0
FA-DQN Collision Penalty (Baseline-paper)	−10.0
Progress Coefficient ($\kappa_{progress}$)	1.0
Step Penalty (step penalty)	0.0
Velocity Bounds ($v_{\min}, v_{\max}$)	0.0, 3.5
Heading-Change Bound ($\Delta\psi_{\max}$)	25^0
Maximum Steps (max-steps)	4 × Manhattan distance

The reward function was carefully specified to balance progress, smoothness, and safety while reflecting realistic UAV motion constraints. The progress term was normalised with a value of α=1.0, thus ensuring that the progress toward the goal directly reflects the Euclidean distance reduction without so much as overpowering other reward components. Moreover, in order to discourage abrupt and sudden turns while also allowing for sufficient exploration, a small smoothness component of β = 0.05 was applied. Terminal rewards were strongly weighted with R_{goal} = 150 to reinforce goal achievement and $R_{collision} = -250$ to then emphasize safety-first behavior since collisions usually represent mission-ending failures in UAV navigation. For a fairer comparison, the baseline FA-DQN, we also defined a 'paper-mode' wherein a goal achievement results in a reward of +10 while a collision results in a -10. Progress was also shaped with $K_{progress}$ = 1.0 and a step penalty of 1.0. It is worth noting that the absence of a step penalty prevents discouraging exploration in the early stages of training, particularly since GP-seeding already accelerates convergence. UAV motion was also dynamically constrained with velocity bounds that we defined as $v_{\min} = 0$ and $v_{\max} = 3.5$, where the lower bound allows hovering and the upper reflects practical maneuverability limits, thus ensuring stable yet efficient flight. In addition to this, a heading change constraint value of $\Delta\Psi_{max}$= 25^0 per step was imposed to enforce smooth and physically feasible turning rates consistent with UAV kinematics. Finally, the episode horizon was then also truncated at four times the Manhattan distance between the start and goal to prevent excessively long episodes while still allowing sufficient exploration. Collectively, these settings were chosen to balance exploration, stability, and safety.

4. Results

In order to rigorously test and evaluate the effectiveness of the proposed GP-seeded Soft-Actor-Critic framework, a series of simulation experiments was conducted across 3 different benchmark grid environments [29]. The experimental evaluation across the three designed environments confirms that GP+SAC converges substantially faster than FA-DQN while maintaining comparable path efficiency. In the scalability test on Map 3, FA-DQN achieved an average path length of 28.38 units, while GP+SAC produced a closely aligned average of 30.55 units, demonstrating that the rapid convergence of GP+SAC does not compromise the quality of generated paths. Nevertheless, the instability observed in reward trajectories, particularly in the more constrained Map 2 with narrow corridors and higher obstacle densities, indicates that performance may be sensitive to highly cluttered environments. From a practical standpoint, while the accelerated learning and efficient path generation are promising, the current framework has yet to be validated against the uncertainties of real-world UAV operations, such as sensor noise, actuation delays, and environmental disturbances. Addressing these factors through robustness enhancements, stability mechanisms, and hardware-in-the-loop testing will be essential to ensure that the advantages demonstrated in

simulation can reliably transfer to physical UAV platforms.

Another important observation is that the efficiency of GP+SAC in producing viable paths across different map structures highlights its adaptability to diverse planning scenarios. However, achieving consistent performance in real-world missions requires additional considerations beyond those captured in simulation. For example, UAVs operating in outdoor environments must contend with dynamic obstacles, variable weather conditions, and limitations in onboard computational capacity. While the hybrid design of GP+SAC provides a strong basis for fast convergence and efficient exploration, future improvements should incorporate mechanisms for energy-aware planning, adaptive policy stabilization, and real-time environment perception. These enhancements will not only improve resilience under uncertainty but also ensure that the framework remains scalable when extended to continuous three-dimensional navigation and cooperative multi-UAV operations.

4.1. Environments Setup

In this study, the UAV navigates within a 2D grid space where each cell can either be free or occupied by an obstacle (a black 1×1 square). To evaluate the proposed GP+SAC approach under diverse and representative conditions, three experimental maps were constructed, corresponding to **Figures 1–3**. In **Figure 1**, obstacles were strategically arranged to form narrow corridors, bottlenecks, and detours, thereby simulating cluttered urban-like environments where UAVs must carefully navigate through tight passages to ensure both efficiency and safety. **Figure 2** depicts a different scenario in which obstacles are concentrated along the bottom and central regions of the grid, effectively blocking the most direct route from the start to the goal. This setup is representative of semi-structured terrains such as suburban or partially restricted airspaces, where UAVs often need to adaptively reroute to avoid obstacles while still maintaining path efficiency. Finally, **Figure 3** presents the most challenging environment with obstacles placed in a checkerboard-like pattern, introducing repeated blockages across rows and columns. This pattern mimics highly dynamic or adversarial environments in which UAVs face recurring obstructions, requiring robust decision-making and adaptability under persistent constraints. Collectively, these three maps form a diverse benchmark suite that captures varying levels of navigation complexity, ensuring that the evaluation of the proposed method is not limited to a single obstacle configuration but instead reflects the range of realistic challenges encountered in real-world UAV path-planning applications.

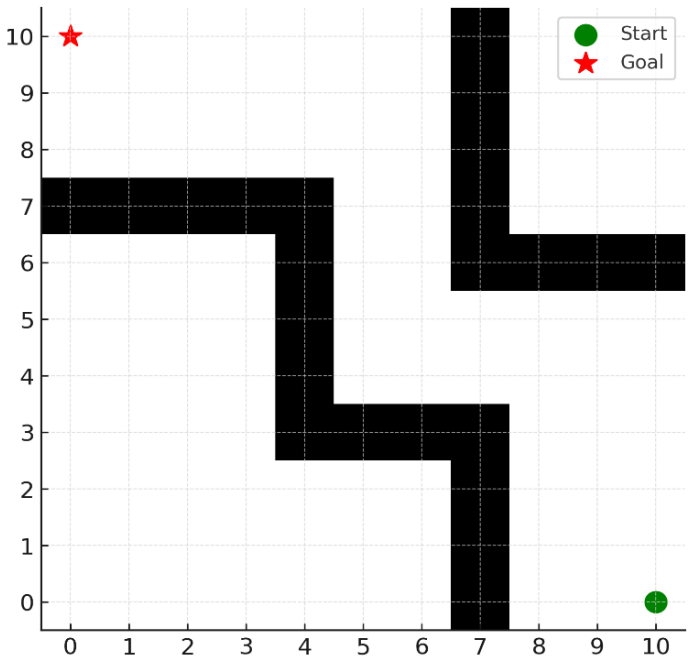

Figure 1. Moderately Constrained Search Space.

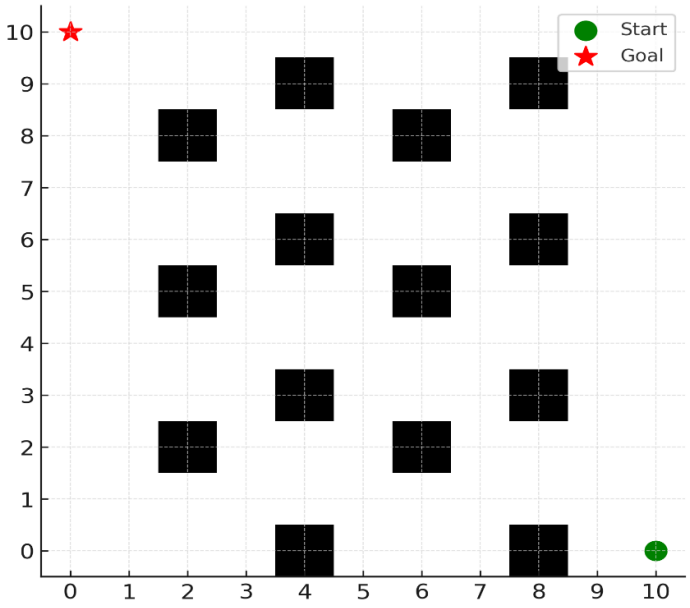

Figure 2. Higher Obstacle Density and Narrow Corridors.

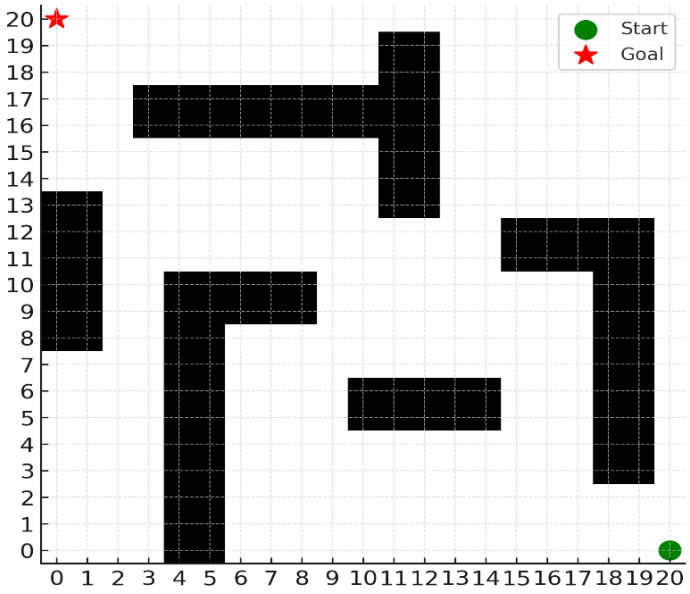

Figure 3. Multiple Horizontal Barriers with Few Narrow Openings.

Figure 3, on the other hand, being larger in size, has multiple horizontal barriers with only a few narrow openings, simulating a highly constrained environment that requires the UAV to search for viable corridors to reach the goal.

The UAV begins at a designated start cell, (0, 0) in this case, and must reach the goal cell while avoiding all obstacles that have been placed around the grid. At every time step, the UAV selects an action that it will execute from its action space, which then determines its movement. A collision in this work is defined to occur if the UAV either lands on an obstacle cell or if its trajectory line (line segment connecting two consecutive coordinates) intersects with an obstacle cell. Through this setup, the environment provides a challenging yet structured testbed for evaluating UAV path planning strategies under varying obstacle densities and configurations.

4.2. Training

A key finding of this work was that the proposed method was able to achieve faster convergence as compared to the baseline FA-DQN framework. It is worth mentioning that in the case of our proposed method, convergence took place earlier, with the agent able to learn viable and acceptable policies in a few episodes, as seen in **Figure 4,** which shows the training and reward structure of the agent in the first map environment.

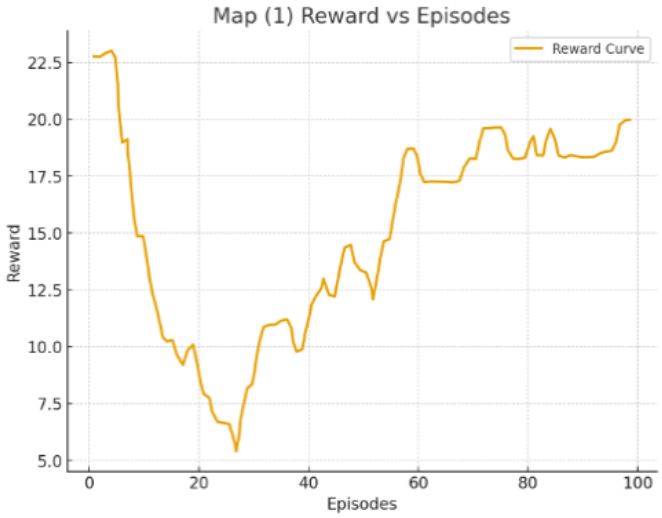

Figure 4. Reward vs Episodes Moving average (GP-SAC).

This is attributable to the demonstration seeding mechanism, where GP provided high-quality initial trajectories that accelerated the early learning phase of SAC. Additionally, when examining the reward profiles as shown in **Figure 5**, it was also noted that the proposed approach was able to achieve higher returns relative to FA-DQN, indicating more efficient exploitation of the reward structure; however, it is also evident that the learning process was unstable throughout, as reflected in the frequent fluctuations of episode rewards. This suggests that although the method is able to achieve higher rewards and earlier convergence, additional mechanisms such as adaptive exploration or regularization may need to be used in the future with the goal of stabilizing performance across training runs.

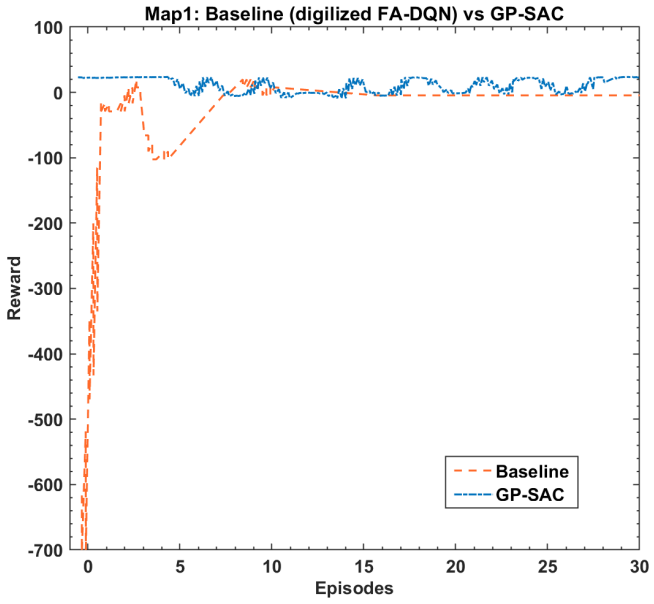

Figure 5. Reward vs Episodes (GP-SAC and FA-DQN).

4.3. Path Efficiency

Another important metric that was utilized to test the efficiency and performance of our proposed method is the average path length, specifically in the third map, as done in the paper [30], which is meant to reflect the UAV's trajectory. Within the third map, the FA-DQN approach achieves a path length of about 28.38 units, with our method achieving one of 30.55 as seen in **Figure 6** below.

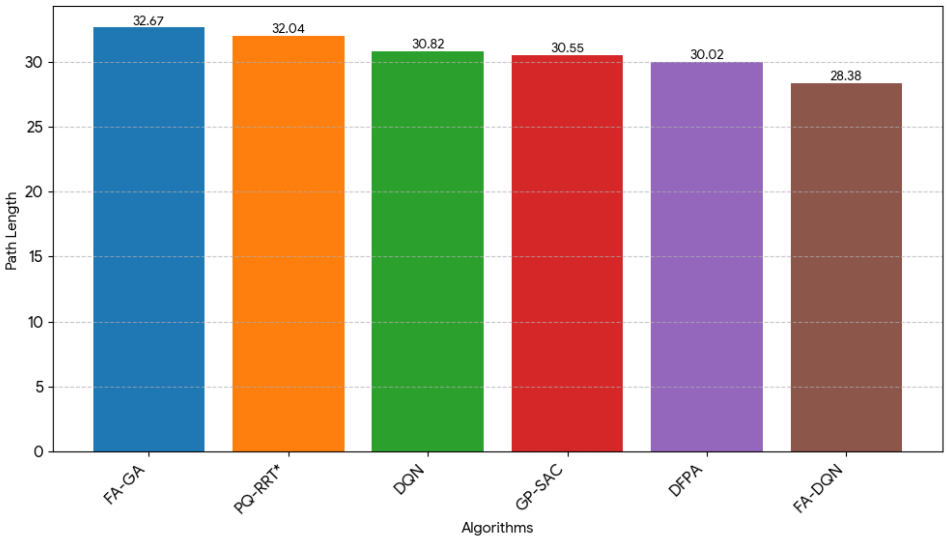

Figure 6. Path length of different algorithms.

It demonstrates the ability of this approach to generate paths that are similar in terms of efficiency to those produced by the FA-DQN despite differences in algorithmic design. Crucially, this indicates that the accelerated convergence of GP-seeded SAC does not come at the cost of path optimality; instead, we see that our method balances speed of convergence with near-optimal paths of comparable lengths. In addition to FA-DQN, the results in **Figure 6** also include the Deterministic Fixed Policy Algorithm (DFPA) as another comparative baseline. While GP+SAC consistently demonstrated superior convergence speed relative to both FA-DQN and DFPA, its path efficiency did not surpass DFPA in certain scenarios. Specifically, DFPA produced slightly shorter paths under stable grid conditions, benefiting from its deterministic structure and lack of exploration variance. However, this advantage comes at the cost of flexibility, as DFPA lacks the adaptive learning capability required for generalization to diverse or dynamic environments. By contrast, GP+SAC, although yielding paths of comparable but not superior efficiency, combines rapid convergence with the capacity to adapt policies across varying map complexities. This balance suggests that DFPA can provide strong performance in fixed, well-structured environments, but GP+SAC is better suited for real-world UAV navigation tasks where adaptability and fast policy learning are critical.

5. Discussion

Planning for efficient and robust UAVs in unknown terrains has constituted one of the high-priority areas for research, which has been inspired mainly by the limitations of the traditional Deep Reinforcement Learning (DRL) approaches, which give rise to sluggish convergence and laborious explorations of trial and error. The hybrid technique of employing heuristic-based methods for facilitating DRL convergence at a faster rate has constituted one of the potential solutions, as our GP+SAC technique and later work, for example, the Firefly Algorithm-augmented Deep Q-Network (FADQN), have indicated.

GP+SAC uses Genetic Programming to provide us with a high-quality "warm-start" policy, effectively skipping the exploratory first phase of the SAC algorithm. The next test results numerically verify this benefit. First, taking a look at convergence dynamics, the GP-seeded SAC agent converged earlier than the FA-DQN baseline, learning plausible policies very quickly within a few episodes. This behaviour is transparently clear from the training reward trace plots, as our method not only converged much more quickly but also achieved higher episodic returns than the

FA-DQN. Nevertheless, it is also interesting that the training was unstable and the trajectories of the reward grew very much from one episode to the next. This implies that while GP seeding favours early learning and enhanced exploitation of the reward, some other mechanism may be required for stabilizing long-term training. In terms of path efficiency, our approach demonstrated a similar level of performance as that of FA-DQN. In Map 3, for which scalability was studied, FA-DQN produced a mean path of length 28.38 units, and GP+SAC achieved a very close average of 30.55 units. This result demonstrates that fast convergence of GP-seeded SAC is not at the cost of the optimality of the path. Instead, the model can balance fast policy learning and efficient generation of paths that are of a similar length as heuristic-guided DRL approaches. Overall, these findings emphasize both the advantages and the limitations of the technique: GP+SAC excels at fast policy convergence and effective generation of trajectories, yet suffers from instability in more constrained environments like Map 2.

Although the proposed GP+SAC framework achieves faster convergence and demonstrates competitive path efficiency when compared with FA-DQN, certain limitations highlight areas for further refinement. One of the primary issues observed was the instability of training dynamics, where episodic rewards exhibited significant fluctuations across episodes. This instability indicates that while GP seeding provides strong initial policies, the underlying SAC component may still require mechanisms such as adaptive learning rates, prioritized experience replay, or regularization techniques to stabilize long-term policy learning. Another limitation lies in the reliance on grid-based simulation environments, which, while effective for controlled experimentation, do not fully capture the complexities of real-world UAV navigation, such as sensor noise, GPS inaccuracies, wind disturbances, and time-varying obstacle dynamics. Moreover, the computational requirements of running hybrid DRL models may pose challenges for deployment on UAV hardware with limited on-board resources, making algorithmic efficiency a critical area of improvement. Future research should therefore explore lightweight network architectures, knowledge distillation, or edge-assisted computation to enhance practical feasibility. Similarly, while the framework was validated for static obstacle-rich scenarios, extending its adaptability to dynamic and adversarial environments remains an important goal. Integrating energy-aware planning modules would also allow the model to balance optimal navigation with UAV endurance constraints, which is critical for real-world operations. Finally, to bridge the gap between simulation and practice, hardware-in-the-loop testing and field trials on actual UAV platforms are necessary to evaluate robustness, scalability, and safety under uncertain and dynamic conditions. These improvements would enable the framework to evolve from a simulation-proven concept into a reliable solution for autonomous UAV navigation in real-world missions.

The critical advantage of the proposed GP+SAC framework lies in its rapid convergence, which is not merely a numerical improvement but a decisive factor for real-world UAV path planning applications. In reinforcement learning, convergence speed directly translates to reduced training time and computational cost, both of which are highly significant when dealing with resource-constrained UAV platforms or time-sensitive mission planning. Unlike FA-DQN, which requires extensive exploration to achieve stable policies, GP+SAC leverages genetic programming to provide a high-quality warm-start, thereby bypassing the costly trial-and-error phase of conventional DRL. This accelerated policy acquisition enables UAVs to learn feasible navigation strategies within a limited number of episodes, which is especially crucial in scenarios where the environment is unknown, rapidly changing, or mission deadlines are strict. Moreover, faster convergence enhances the practicality of deploying DRL-based methods in iterative re-training cycles, such as when UAVs must adapt online to new terrains, dynamic obstacles, or evolving mission goals. Therefore, the importance of rapid convergence extends beyond simulation efficiency; it directly impacts scalability, adaptability, and feasibility of autonomous UAV navigation in operational settings.

6. Conclusions

This work demonstrates that the proposed UAV navigation method can reliably and effectively generate viable paths in grid-based, obstacle-rich environments. The mathematical formulation clearly encompasses obstacle avoidance, efficiency of path, and achievement of the goal as components of one single optimization problem in order to enable robust decision-making in uncertain environments. The simulation results indicate that the system has high success rates and shorter path lengths compared to standard heuristic and uninformed search strategies. The framework's adaptiveness to the type of grid change and obstacle densities entails high extensional ability to continuous 3D guidance, real-time mission planning, and multi-UAV cooperative scenarios.

In addition, the transferability of the proposed method to real UAV hardware platforms is promising, as the

learned policies can be integrated with on-board navigation and sensing systems to support real-time decision-making. However, practical deployment would require addressing issues such as robustness to sensor noise, computational limitations of UAV hardware, and stability of control under uncertain environmental conditions.

The future work would be focused on the integration of dynamic obstacle management, energy-efficient path planning, and hardware-in-the-loop verification in order to close the gap to real-world deployment from simulation.

Author Contributions

Conceptualization, M.U.M., P.N. and O.M.; methodology, T.S. and H.V.; software, F.T.; validation, M.U.M., H.V. and O.M.; formal analysis, H.V.; investigation, K.J.M.; resources, K.J.M.; data curation, T.S.; writing—original draft preparation, P.N.; writing—review and editing, M.U.M. and P.N.; visualization, O.M.; supervision, H.V., M.U.M. and T.S.; project administration, M.U.M. All authors have read and agreed to the published version of the manuscript.

Funding

This work received no external funding.

Institutional Review Board Statement

Not applicable.

Informed Consent Statement

Not applicable.

Data Availability Statement

The data that support the findings of this study are available from the corresponding author upon reasonable request.

Conflicts of Interest

The authors declare no conflict of interest.

References

1. Wei, Z.; Zhu, M.; Zhang, N.; et al. UAV-Assisted Data Collection for Internet of Things: A Survey. *IEEE Internet Things J.* **2022**, *9*, 15460–15483.
2. Liu, L.; Wang, A.; Sun, G.; et al. Multi-Objective Optimization for Data Collection in UAV-Assisted Agricultural IoT. *IEEE Trans. Veh. Technol.* **2024**, *7*, 6488–6503.
3. Hsueh, H.-Y.; Toma, A.-L.; Jaafar, H.A.; et al. Systematic Comparison of Path Planning Algorithms Using PathBench. *Adv. Robot.* **2022**, *36*, 566–581.
4. Sun, W.; Huan, J.; Wang, Q.; et al. FADQN: A Heuristic Reinforcement Learning Mechanism for UAV Path Planning in Unknown Environment. *IEEE Access* **2025**, *13*, 131909–131921.
5. Zhang, J.; Xian, Y.; Zhu, X.; et al. A Hybrid Deep Learning Model for UAV Path Planning in Dynamic Environments. *IEEE Access* **2025**, *13*, 67459–67475.
6. Mohammed, Y.I.; Hassan, R.; Hasan, M.K.; et al. Revolutionizing FANETs With Reinforcement Learning: Optimized Data Forwarding and Real-Time Adaptability. *IEEE Open J. Commun. Soc.* **2025**, *6*, 4295–4310.
7. Medani, K.; Gherbi, C.; Mabed, K.; et al. Energy-Efficient Q-Learning-Based Path Planning for UAV-Aided Data Collection in Agricultural WSNs. *Internet Things* **2025**, *33*, 101698.
8. Bashir, N.; Boudjit, S.; Dauphin, G.; et al. An Obstacle Avoidance Approach for UAV Path Planning. *Simul. Model. Pract. Theory* **2023**, *129*, 102815.
9. Mushtaq, M.U.; Venter, H.; Singh, A.; et al. Advances in Energy Harvesting for Sustainable Wireless Sensor Networks: Challenges and Opportunities. *Hardware* **2025**, *3*, 1.
10. Umer, M.M.; Venter, H.; Muhammad, O.; et al. Cognitive Strategies for UAV Trajectory Optimization: Ensuring Safety and Energy Efficiency in Real-World Scenarios. *Ain Shams Eng. J.* **2025**, *16*, 103301.

11. Zhang, Q.; Guo, Q.; Jiang, H.; et al. EMD Empowered Neural Network for Predicting Spatio-Temporal Non-Stationary Channel in UAV Communications. *Appl. Intell.* **2025**, *55*, 285.
12. Yang, K.; Liu, L. An Improved Deep Reinforcement Learning Algorithm for Path Planning in Unmanned Driving. *IEEE Access* **2024**, *12*, 67935–67944.
13. Wang, Z.; Ng, S.X.; Mohammed, E.-H. Deep Reinforcement Learning Assisted UAV Path Planning Relying on Cumulative Reward Mode and Region Segmentation. *IEEE Open J. Veh. Technol.* **2024**, *5*, 737–751.
14. Sharma, G.; Jain, S.; Sharma, R.S. Path Planning for Fully Autonomous UAVs—A Taxonomic Review and Future Perspectives. *IEEE Access* **2025**, *13*, 13356–13379.
15. Muhammad, O.; Jiang, H.; Bilal, M.; et al. Optimizing Power Allocation for URLLC-D2D in 5G Networks With Rician Fading Channel. *PeerJ Comput. Sci.* **2025**, *11*, e2712.
16. Zhang, Q.; Jiang, H.; Guo, Q.; et al. UAV-Assisted Wireless Communication Network Capacity Analysis and Deployment Decision. In *Communications, Signal Processing, and Systems. CSPS 2020. Lecture Notes in Electrical Engineering*; Liang, Q., Wang, W., Liu, X., et al., Eds.; Springer: Cham, Switzerland, 2020; 654, pp. 1185–1195.
17. Oladeji-Atanda, G. *A Multi-Greedy Forwarding Approach in Geographic Packet Routing*; Botswana International University of Science and Technology: Palapye, Botswana, 2023.
18. Parekh, R.; Honavar, V. Learning DFA from Simple Examples. *Mach. Learn.* **2001**, *44*, 9–35.
19. Karvinen, T. Configuration Management of Distributed Systems over Unreliable and Hostile Networks. Ph.D. Thesis, University of Westminster, London, UK, 15 May 2023.
20. Li, Y.; Li, S.; Zhang, Y.; et al. Dynamic Route Planning for a USV-UAV Multi-Robot System in the Rendezvous Task With Obstacles. *J. Intell. Robot. Syst.* **2023**, *107*, 52.
21. Kaljaca, D.; Vroegindeweij, B.; Van Henten, E. Coverage Trajectory Planning for a Bush Trimming Robot Arm. *J. Field Robot.* **2020**, *37*, 283–308.
22. Fu, H.; Li, Z.; Zhang, W.; et al. Research on Path Planning of Agricultural UAV Based on Improved Deep Reinforcement Learning. *Agronomy* **2024**, *14*, 2669.
23. Debnath, D.; Vanegas, F.; Boiteau, S.; et al. An Integrated Geometric Obstacle Avoidance and Genetic Algorithm TSP Model for UAV Path Planning. *Drones* **2024**, *8*, 302.
24. Sonmez, C.; Ozgovde, A.; Ersoy, C. EdgeCloudSim: An Environment for Performance Evaluation of Edge Computing Systems. *Trans. Emerg. Telecommun. Technol.* **2018**, *29*, e3493.
25. Javed, S.; Hassan, A.; Ahmad, R.; et al. State-of-the-Art and Future Research Challenges in UAV Swarms. *IEEE Internet Things J.* **2024**, *11*, 19023–19045.
26. Kadu, A.; Reddy, K.V.; Gawande, U. Innovative AgTech: A Predictive Machine Learning-Driven Precision Farming Solution for Enhancing Agricultural Productivity in Wardha. In Proceedings of the 2nd DMIHER International Conference on Artificial Intelligence in Healthcare, Education and Industry (IDICAIEI), Wardha, India, 29–30 November 2024; pp. 1–6.
27. Shao, L.; Qian, L.; Wu, M.; et al. Integrated Clustering and Routing Design and Triangle Path Optimization for UAV-Assisted Wireless Sensor Networks. *China Commun.* **2024**, *21*, 178–192.
28. Amodu, O.A.; Jarray, C.; Mahmood, R.A.R.; et al. Deep Reinforcement Learning for AoI Minimization in UAV-Aided Data Collection for WSN and IoT: A Survey. *IEEE Access* **2024**, *12*, 108000–108040.
29. Nwachukwu, S.E.; Folly, K.A.; Awodele, K.O. A Comparative Study Between Soft Actor-Critic (SAC) and Deep Deterministic Policy Gradient (DDPG) Algorithms for Solar PV MPPT Control Under Partial Shading Conditions. *IEEE Access* **2025**, *13*, 71738–71754.
30. Cheng, M.-M.; Fan, D.-P. Structure-Measure: A New Way to Evaluate Foreground Maps. *Int. J. Comput. Vis.* **2021**, *129*, 2622–2638.