electronics

MDPI

Article

An Evolutionary Game-Theoretic Approach to Triple-Strategy Coordination in RRT*-Based Path Planning

Lin Qi, Yongping Hao *, Liyuan Yang and Meixuan Li

School of Equipment Engineering, Shenyang Ligong University, Shenyang 110159, China; 2311610327@stu.sylu.edu.cn (L.Q.); 2100600007@stu.sylu.edu.cn (L.Y.); 2200600015@stu.sylu.edu.cn (M.L.)
* Correspondence: hyp@sylu.edu.cn

Abstract: To address the limitations of the RRT-series algorithm and its variants, which have demonstrated a lack of dynamic adaptability in various environments, poor path quality, slow convergence rates, and a tendency to become trapped in local optima, this paper aims to propose a dynamic multi-strategy path-planning algorithm based on EG-DRRT* (Evolutionary Game-Theoretic Dynamic RRT*). The integration of evolutionary game theory allows the algorithm to use the concept of dynamically updating replicators to develop the payoff function for a fusion of multi-strategies. This enables a dynamic adjustment of the usage ratios of RRT*, Dijkstra, and Goal Bias algorithms. This approach directs the search process toward converging on the goal point, ultimately achieving an ESS (Evolutionarily Stable Strategy) equilibrium. The experimental results reveal that EG-DRRT* successfully establishes a dynamic balance between global exploration and local optimization across various environments, demonstrating remarkable adaptability and robustness. Additionally, EG-DRRT* shows substantial advantages compared to existing algorithms.

Keywords: path planning; EG-DRRT*; evolutionary game theory; ESS

Academic Editor: Mahmut Reyhanoglu

Received: 16 March 2025
Revised: 30 March 2025
Accepted: 2 April 2025
Published: 3 April 2025

Citation: Qi, L.; Hao, Y.; Yang, L.; Li, M. An Evolutionary Game-Theoretic Approach to Triple-Strategy Coordination in RRT*-Based Path Planning. *Electronics* **2025**, *14*, 1453. https://doi.org/10.3390/electronics14071453

1. Introduction

The increasing use of UAVs and autonomous driving technology is driving significant advancements in path-planning technology. As a fundamental aspect of autonomous navigation in unmanned systems, effective path planning directly impacts the success rate of missions and overall operational efficiency [1,2]. Modern path planning aims to overcome the limitations of traditional algorithms. It not only addresses the fundamental challenges of graph search and obstacle avoidance but also considers the adaptability to dynamic environments. Furthermore, it takes into account the dynamics and kinematics constraints, as well as various optimization requirements [3,4]. The RRT family of algorithms, which rely on probabilistic sampling methods, plays a significant role in exploring high-dimensional spaces today. Additionally, graph-search-planning algorithms, such as A* and Dijkstra, are becoming increasingly important in structured environments [5–7]. Path-planning technology is making significant advancements, enabling robots to not only perform their primary task of autonomous navigation but also improve their ability to adapt to different environments. This progress is crucial for the future development of an intelligent society and will play a vital role in facilitating collaborative operations among various intelligent systems [8–10].

Kuffner and LaValle first proposed the RRT algorithm [11]. The algorithm rapidly explores the high-dimensional search space by random sampling and incrementally constructing trees. It has the advantages of simple implementation and adaptability and is

widely used in robot path planning. However, the RRT algorithm was originally designed to find a feasible path quickly, does not guarantee the optimality of the path, and exhibits the disadvantage of low expansion efficiency in narrow-channel environments. Karaman and Frazzoli proposed the RRT* algorithm [12]. It is an improved version of the traditional RRT algorithm. RRT* achieves asymptotic convergence performance to the globally optimal solution and theoretically overcomes the path suboptimality problem of RRT by continuously optimizing the path cost during the expansion process. However, the cost of RRT* is that it significantly increases the computation time and complexity of the algorithm, especially in high-dimensional spaces or complex environments, where its performance is limited by the number of nodes and computational resources, and its real-time performance is difficult to meet the demand of dynamic planning. The RRT-Connect algorithm, proposed by Kuffner and LaValle [13], enhances path-planning efficiency through bi-directional search. However, the paths it generates often lack smoothness, leading to a decline in path quality. Seeking to improve upon this aspect, Marcucci T et al. introduced a path-planning algorithm based on an improved PRM [14], which enhances path smoothness through a post-optimization step. However, this optimization process adds extra computational overhead. The reinforcement learning approach combined with path planning, as proposed by Wang et al. [15], has demonstrated significant adaptability in complex environments. However, the training process for this method relies heavily on a substantial amount of labeled data, making it challenging to generalize to unfamiliar environments. A common issue faced by existing algorithms is generating high-quality paths in complicated settings while maintaining computational efficiency. The Adapt-RRT algorithm, introduced by Choudhury et al. [16], seeks to improve adaptation to complex environments by optimizing the sampling region. Nevertheless, its performance remains suboptimal in areas with dense obstacles. In contrast, Dang et al. developed a path-planning framework known as SIL-RRT* [17], which integrates deep learning with path optimization. This approach has made notable advancements in navigating complex dynamic environments, yet its practical application is hindered by the lengthy training time required for the model. Recently, Y and HE proposed a two-layer optimized A* algorithm that utilizes a dynamic window approach for UAV path planning in complex environments [18], incorporating specific rules for clipping neighboring nodes. While this method shows effectiveness in such intricate settings, further investigation is needed to determine its adaptability across diverse environments. As a result, enhancing the adaptability of path-planning algorithms in complex environments remains a vital area for future research.

An analysis of the aforementioned classical algorithms reveals that, despite significant progress in path planning, they exhibit several key limitations: the absence of strict guarantees for path optimality, inadequate adaptability in complex environments, and substantial computational demands. These issues highlight a clear direction for improvement in the EG-DRRT* algorithm proposed in this paper. Specifically, the aim is to enhance the efficiency and adaptability of path planning by integrating the global optimality of Dijkstra's algorithm with the asymptotic optimality of the RRT* algorithm, alongside the dynamic adjustment of the policy.

The rest of this paper is organized as follows: Section 2 presents an overview of the RRT algorithm and Dijkstra's algorithm, along with a description of the problem being addressed. Section 3 offers a detailed explanation of the EG-DRRT* algorithm. In Section 4, we compare our algorithm with other existing algorithms and present the results of our simulation experiments. Finally, Section 5 summarizes the advantages of the EG-DRRT* algorithm and proposes directions for future research.

2. Related Research Background

2.1. Analysis of Classical RRT Algorithm*

The main steps in the path expansion process of the RRT* algorithm include:

Firstly, as in Figure 1a, the existing path is extended to the current node x_{near} from the start node x_{Start}. Based on this, a search radius $step\_size$ is defined around x_{near} to generate a new path node x_{new}, and all nodes within this radius are considered candidate parent nodes. From the candidate parent nodes, the one with the shortest distance from x_{Start} and the lowest cost is selected as the new parent. The new node x_{new} is connected to the chosen parent node at this stage, creating a new path. During the expansion process, the algorithm optimizes the overall structure of the path by selecting an appropriate parent node connected to x_{new}. Specifically, x_1 expands further in the search space by connecting to x_{new} as a temporary node. Based on this, the expansion continues, and x_2 is added to the path and reconnected with x_{new}. Path optimization is further implemented in step (2). The node x_2 becomes part of the new path, updating the connections to minimize total cost by re-evaluating all parent nodes, as detailed in Algorithm 1.

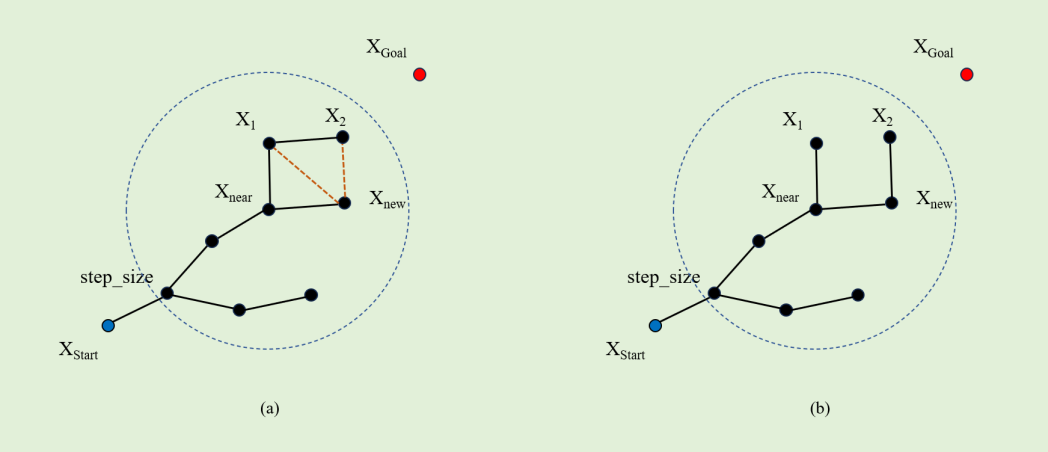


Figure 1. A schematic diagram of the RRT* algorithm: (**a**) initial wiring setup; (**b**) rewire.

Algorithm 1 RRT*

Input: Start node x_{start}, goal node x_{goal}, maximum iterations max_iter;
Output: Optimal path;

```
Initialize an empty tree T with root at x_start;
for i = 1 to max_iter do
    x_rand ← Randomly sample a point;
    x_near ← Randomly sample a point in T;
    x_new ← Move towards x_rand from x_near;
    if x_new is collision-free then
        Add x_new to T;
        Optimize parent of x_new within its neighbors;
        Rewire neighbors of x_new to move the path;
    end if
end for
return the best path from x_start to x_goal;
```

The RRT* algorithm utilizes random sampling for planning, but this approach may result in many invalid nodes and increase computational costs [19].

2.2. Analysis of Classical Dijkstra's Algorithm

Dijkstra's algorithm is a classical method for finding the shortest path from a single source node in a graph. It progressively explores neighboring nodes based on a greedy approach. This algorithm is widely used in various path-planning problems due to its effectiveness in searching for the shortest paths [20–22]. Dijkstra's algorithm considers the edge weights of the graph as the costs of the paths, typically defining these weights as the Euclidean distance between two nodes. For two points $x = (x_1, x_2, \cdots, x_n)$ and $y = (y_1, y_2, \cdots, y_n)$ in n-dimensional space, the Euclidean distance can be expressed as follows:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{1}$$

Dijkstra's algorithm maintains a predecessor node for each node in the graph. It records the shortest distance to be added to the priority queue. As nodes are processed, Dijkstra compares the distances of each node with its neighboring nodes. If a shorter path to a neighboring node is found, the predecessor node is updated, and the new distance is reinserted into the priority queue. This process continues until all nodes have been completed, as detailed in Algorithm 2. Dijkstra's algorithm consistently seeks the shortest path using a fixed greedy approach. However, it lacks the flexibility to incorporate other search strategies. In high-dimensional spaces or large-scale graphs, this global search characteristic results in high computational complexity, leading to limited adaptability in complex environments.

Algorithm 2 Dijkstra

Input: Graph G, start node x_{start};
Output: Shortest path distances from x_{start};
Initialize distances: dist$[x_{start}] = 0$, others = $+\infty$;
Add x_{start} to priority queue Q;
while Q is not empty **do**
    u $\leftarrow$ Extract node with the smallest distance from Q;
    for each neighbor v of u **do**
        if dist$[u]$ + cost(u, v) < dist$[v]$: **then**
            Update dist$[v] \leftarrow$ dist$[u]$ + cost(u, v);
            Update priority of v to u;
            Add v to Q
        end if
    end for
end while
return distances;

3. The Proposed Algorithm

As environmental complexity and the diversity of task requirements increase, traditional path-planning algorithms often struggle to handle various environmental features effectively. To better adapt to these diverse scenarios, we propose a dynamic multi-strategy path-planning algorithm called EG-DRRT*. This algorithm leverages the complementary strengths of RRT*, Dijkstra, and Goal Bias by dynamically adjusting the proportion of each strategy used during the search process. In this chapter, we provide a detailed description of the core design of the EG-DRRT* algorithm. First, Section 3.1 will analyze the design principles and implementation details of the strategy evolution mechanism. Following that, Section 3.2 will elaborate on the specific implementation of integrating strategy evolution with path planning.

3.1. Mechanisms for Evolving Strategies

Evolutionary game theory originated in biology and was initially applied in economic theory to analyze the dynamic evolutionary processes of various strategies in competitive and cooperative environments. The goal is to optimize system performance by adjusting strategy distribution based on the payoff function [23]. Evolutionary game theory offers a flexible and effective approach to solving path-planning problems through dynamic adjustments, adaptive optimization, and management of group strategy distribution [24]. Its unique features not only address the limitations of traditional game methods but also provide a theoretical foundation for tackling complex optimization challenges that involve multi-strategy collaboration. Evolutionary game theory manages the adjustment of strategy weights by utilizing dynamic equations, allowing for continuous optimization of strategy proportions. In contrast, traditional adaptive weight adjustment relies on historical performance to modify weights, which can lead to a bias towards a single strategy and may result in local optimization issues. Overall, evolutionary game theory offers superior adaptability and stability compared to traditional methods. In this algorithm, the three strategies—RRT*, Dijkstra, and Goal Bias—each have their advantages and disadvantages. By incorporating evolutionary game theory, we calculate the priority of each strategy at different stages based on a payoff function. This allows us to adjust the proportion of each strategy adaptively during the algorithm's iterative process, achieving a dynamic balance of strategies as the search progresses. Each strategy (RRT*, Dijkstra, and Goal Bias) is considered an independent player in the game model, and its payoff is determined by performance in the current search phase. To assess the performance of the current search phase, set the environment attribute vector as follows:

$$S(t) = [s_1(t), s_2(t), s_3(t)] \tag{2}$$

In this context, $s_1(t)$ represents the density of obstacles, which is positively correlated to its value. $s_2(t)$ refers to the known degree of the map; the higher the value, the more comprehensively the map is understood. $s_3(t)$ indicates the proximity to the target, meaning that a larger value signifies that the search is closer to the target point. The dynamic gain function is formulated based on the environmental attribute vector.

$$R_1(t) = (1 - \frac{t}{T}) \cdot \frac{1}{1 + s_1(t)} + \lambda_1 \cdot s_3(t) \tag{3}$$

$$R_2(t) = \frac{t}{T} \cdot s_2(t) + \lambda_2(1 - s_1(t)) \tag{4}$$

$$R_3(t) = \frac{1}{1 + e^{-\gamma(\frac{t}{T} - \theta)}} \cdot (1 - s_3(t)) + \mu(t)Q_i \tag{5}$$

In this context, $T = max\_iter$ represents the maximum number of iterations, while $R_1(t)$, $R_2(t)$, and $R_3(t)$ indicate the gains achieved by the three strategies: RRT*, Dijkstra, and Goal Bias, as the number of iterations varies. The parameters $\lambda_1, \lambda_2 > 0$ are tuning parameters that reflect the impact of proximity to the goal and the density of obstacles on the gains, respectively. A larger value of λ_1 will significantly increase the gain of the RRT* algorithm when the goal is closer. Similarly, increasing λ_2 will lead to a greater gain for Dijkstra's algorithm. To balance these two gains, both λ_1 and λ_2 are set to 1 in this experiment.

To quickly adapt to the optimal strategy for achieving the goal, mutation gains are introduced in this evolutionary game concept, along with the mutation probability.

$$\mu(t) = \min\{1, \alpha(\Delta d - d(t))\} \tag{6}$$

The parameter $\alpha > 0$ regulates the sensitivity of mutation. A higher value of α increases the probability of mutation, which accelerates convergence. Conversely, a smaller α slows down the mutation process. To balance these two effects, α is set to 0.05 in the experiment. $d(t)$ denotes the current distance to the target point. Δd is the predetermined step size. When Δd exceeds $d(t)$, a mutation occurs to quickly reach the target point, thereby conserving time and resources. The distribution of the target strategy after this mutation is as follows:

$$Q_i = \begin{cases} 1, & i = 3 \\ 0, & i = 1, 2 \end{cases} \tag{7}$$

The overall complexity of the weight update formula is $O(N)$, with N representing the number of strategies involved in the game. This indicates that the impact of strategy updates on overall computational efficiency is linear. Consequently, the overall computational complexity of the algorithm is primarily determined by the path-planning algorithm. This algorithm employs three independent strategies: RRT*, Dijkstra, and Goal Bias. Previous studies have shown that these strategies demonstrate good scalability in three-dimensional space. Additionally, the computational complexity of updating weights is minimal compared to the overall algorithm, allowing it to run efficiently even in scenarios with high real-time requirements. To maximize the effectiveness of the strategies at any given moment, this algorithm also introduces a competitive penalty factor, denoted as β. It modifies the extent of the return adjustments based on t. Higher values increase competition among strategies, while lower values weaken their impact on strategy selection. This factor enables the concept of negative feedback in the weight-updating process, penalizing the underperforming current strategy by reducing its weight. This approach encourages the system to transition toward a plan that demonstrates better performance. Let $\beta(t)$ represent the competitive penalty factor at the current moment; its dynamic adjustment is as follows:

$$\beta(t) = \beta_{min} + (\beta_{max} - \beta_{min}) \cdot \frac{t}{T} \tag{8}$$

The maximum competitive penalty factor, $\beta_{max} = 0.5$, is pre-set, while the minimum value is $\beta_{min} = 0.1$. Initially, the competition penalty factor is low, which allows for the retention of a greater number of strategies and enhances exploitability. As the process advances, the competition penalty factor is gradually increased to encourage a focus on better-performing strategies. Towards the end of the process, the competitive penalty factor approaches its maximum value, indicating that the offsetting strategy is no longer a factor in the decision making. As the number of iterations, denoted as t, increases, the competitive penalty factor gradually rises from its minimum value β_{min} to its maximum value β_{max}. This trend applies to the overall phase adjustment of the algorithm and aligns with the evolutionary mechanism of the strategy.

Using the concept of dynamic updating of replicators, the returns for each strategy are adjusted based on their respective weights. The formula for the dynamic adjustment update in the evolutionary game is as follows:

$$\widetilde{\omega}_i(t+1) = \omega_i(t)[1 + \beta(t) \cdot (R_i(t) - \bar{R}(t))] \tag{9}$$

where $\omega_i(t)$ represents the current strategy weight of RRT*, Dijkstra, and Goal Bias at time t, and $\bar{R}(t)$ denotes the average of the strategy returns.

$$\bar{R}(t) = \sum_{j=1}^{3} \omega_j(t) R_j(t) \tag{10}$$

3.2. Strategy Implementation and Route Planning

The weights for each of the strategies calculated earlier are normalized and updated.

$$\omega_i(t+1) = (1-\mu(t)) \cdot \frac{\widetilde{\omega}_i(t+1)}{\sum_{j=1}^{3} \omega_i(j+1)} \tag{11}$$

The normalized weights serve as the probability for selecting the final path in the planning process.

$$P_i(t+1) = \omega_i(t+1) \tag{12}$$

The adjustment of strategy weights is determined by the dynamic equation for replication:

$$\frac{dP_i(t)}{dt} = P_i(t)(R_i(t) - \bar{R}(t)), \quad i \in \{1,2,3\} \tag{13}$$

The rate of weight change is shown to be proportional to the deviation of strategy returns. Based on the probabilistic model described above, a current execution strategy is chosen during each path search by generating a random number $r \in [0,1]$. The rules for strategy selection are as follows:

$$Strategy(t+1) = \begin{cases} \text{RRT*}, & \text{if } r < P_1(t+1) \\ \text{Dijkstra}, & \text{if } P_1(t+1) \le r < P_1(t+1) + P_2(t+1) \\ \text{Goal Bias}, & \text{if } r \ge P_1(t+1) + P_2(t+1) \end{cases} \tag{14}$$

The sampling point x_{rand} is generated based on the following probability distribution:

$$P(x_{rand}) = \begin{cases} P_{goal}, & \text{if } x_{rand} = x_{goal} \\ 1 - P_{goal}, & otherwise \end{cases} \tag{15}$$

When P_{goal} represents the predefined target-point bias probability, employing the target bias strategy can efficiently and quickly correspond to the target point, as illustrated in Figure 2.

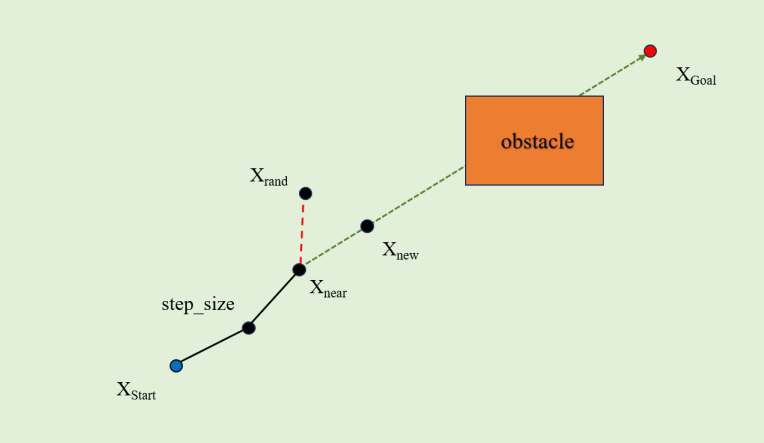


Figure 2. A diagram of the target bias strategy.

The tuning strategy for the evolutionary game is outlined as follows: At the beginning of the exploration phase, the RRT* strategy takes precedence, facilitating rapid global expansion to explore the unknown environment and ensure comprehensive path coverage. During this phase, the weights assigned to Dijkstra and Goal Bias are kept low. As the search process progresses, the influence of Dijkstra's algorithm gradually increases to improve the

quality of the path. This enhancement is achieved by optimizing the connections between critical points on the path, utilizing the algorithm's precise capability to find the shortest route. Toward the end of the exploration, the weight assigned to the Goal Bias strategy is significantly increased, which helps the search process converge rapidly toward the target point. This method not only shortens the path length but also reduces the computational time. The system can gradually converge to a final adaptive steady state, where the strategy combinations become balanced and an efficient path is established. From the perspective of system theory, the iteration of this algorithm treats the behavior of group strategies as a dynamic system. During this process, each strategy's behavior and its relationship with the group are depicted by the principle of time irreversibility. This approach leads to the establishment of a final strategy that achieves the target goal without interference from other strategies. Compared to the Nash equilibrium, this method demonstrates absolute robustness and aligns with the ESS equilibrium, as illustrated in the accompanying Figure 3.

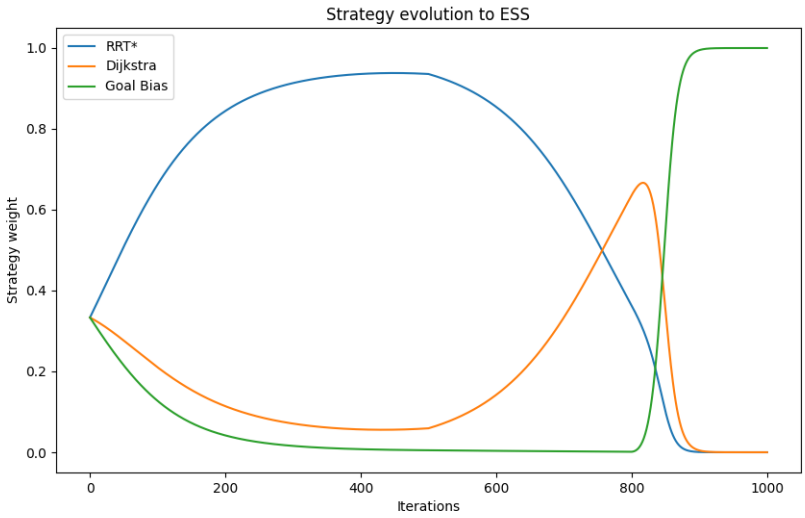


Figure 3. Diagram of Evolutionarily Stable Strategy.

4. Experimental Results and Discussion

To evaluate the performance of the EG-DRRT* algorithm, we compared it with the RRT, RRT*, and RRT-Connect algorithms in various environments, including simple, complex, and 3D settings. To adhere to the principle of controlling variables, we maintained consistent parameter settings across all comparison experiments. During the pre-experimental phase, we tested various parameter combinations and found that their effects on the optimization results were minimal, with all configurations converging to the ESS equilibrium. As a result, we selected a set of parameters that effectively balanced computational efficiency and algorithm performance. Due to the randomness inherent in RRT-series algorithms, we averaged the experimental data over 1000 iterations. The experiments were conducted using Python 3.12 on a Windows 11 operating system, with hardware specifications of a 2.60 GHz Intel (R) Core (TM) i5-13600KF CPU (Intel Corporation, Santa Clara, CA, USA) and 32 GB of RAM.

Figure 4 compares the success rates of four algorithms in simple, complex, and 3D environments. EG-DRRT* achieves high success rates across these different environments, particularly in simple and 3D scenarios. In contrast, the traditional RRT, RRT*, and RRT-Connect algorithms significantly decrease success rates in high-complexity situations, indicating that they are less robust when dealing with complex environments. Meanwhile, EG-DRRT* demonstrates a stronger ability to adapt.

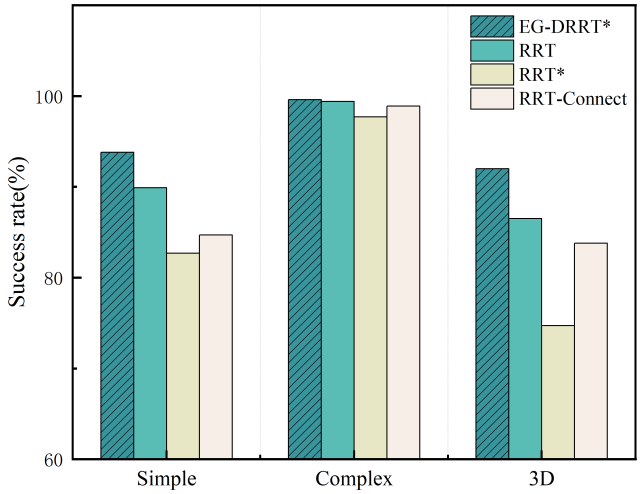


Figure 4. The effectiveness of the four algorithms in various environments.

4.1. Simple Environment

In a simple 100 × 100 environment, the starting point is at (5, 95) and the goal point is at (95, 5). While the RRT family of algorithms typically requires many iterations to find a clear path and is effective for path planning in complex, unstructured environments, it can generate a significant number of unnecessary and redundant branches in simpler paths. This issue can lead to increased time and resource consumption.

In the simple environment presented in Table 1 and Figure 5, EG-DRRT* demonstrates a significant advantage in computation time compared to RRT, RRT*, and RRT-Connect, indicating that it can provide quick responses. Additionally, the path length is reduced by 9.8% compared to RRT and by 8.4% compared to RRT-Connect. EG-DRRT* also has the lowest number of iterations among all the algorithms. Although the path generated by RRT* is 3.2% shorter than that of EG-DRRT*, the time cost for RRT* is considerably higher. Overall, EG-DRRT* achieves a better balance between efficiency and path length.

Table 1. The experimental data on four algorithms tested in a simple environment.

Algorithm Category	Time/s	Length/m	Iterations
EG-DRRT*	0.12	155.81	745.3
RRT	0.29	172.76	1212.0
RRT*	1.53	150.83	1369.5
RRT-Connect	0.63	166.53	1246.7

4.2. Complex Environment

In the 100 × 100 complex environment, the starting point is located at (5, 95), while the goal point is at (95, 5). This environment presents more obstacles, fewer feasible paths, and a denser layout compared to the simpler environment. Our algorithm demonstrates a level of randomness comparable to the other three algorithms in the initial stages. However, it reaches the goal point more quickly in the later stages, especially when the complexity of obstacles is lower. This efficiency reduces both the time needed and computational redundancy, as illustrated in Figure 6.

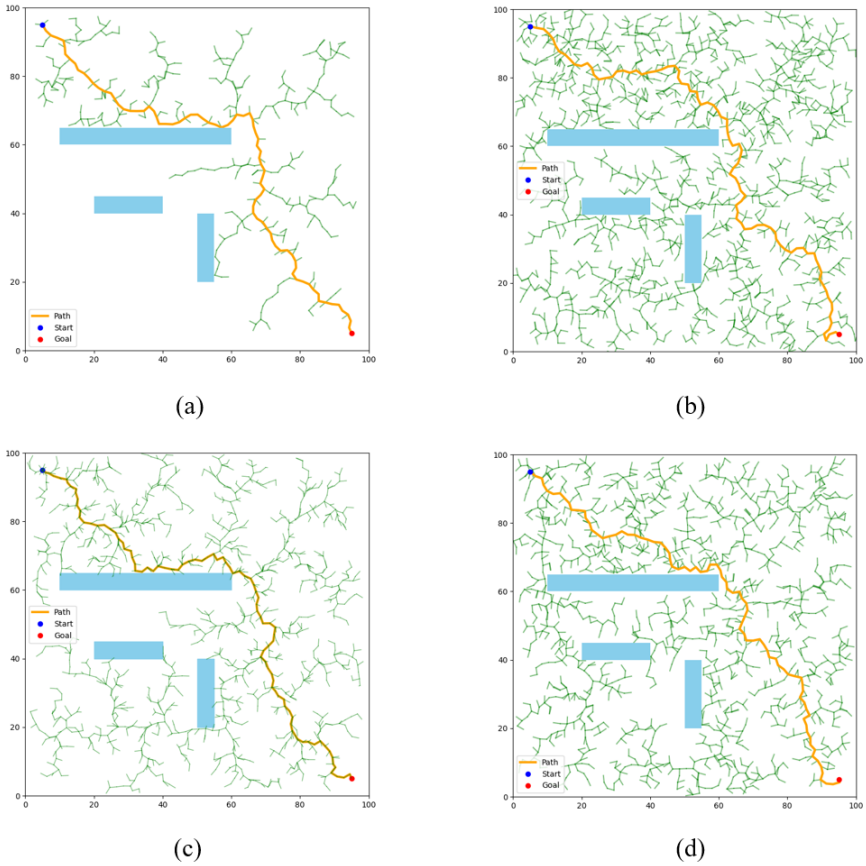


Figure 5. Path diagrams of four algorithms: (**a**) EG-DRRT*, (**b**) RRT, (**c**) RRT*, and (**d**) RRT-Connect in a simple environment.

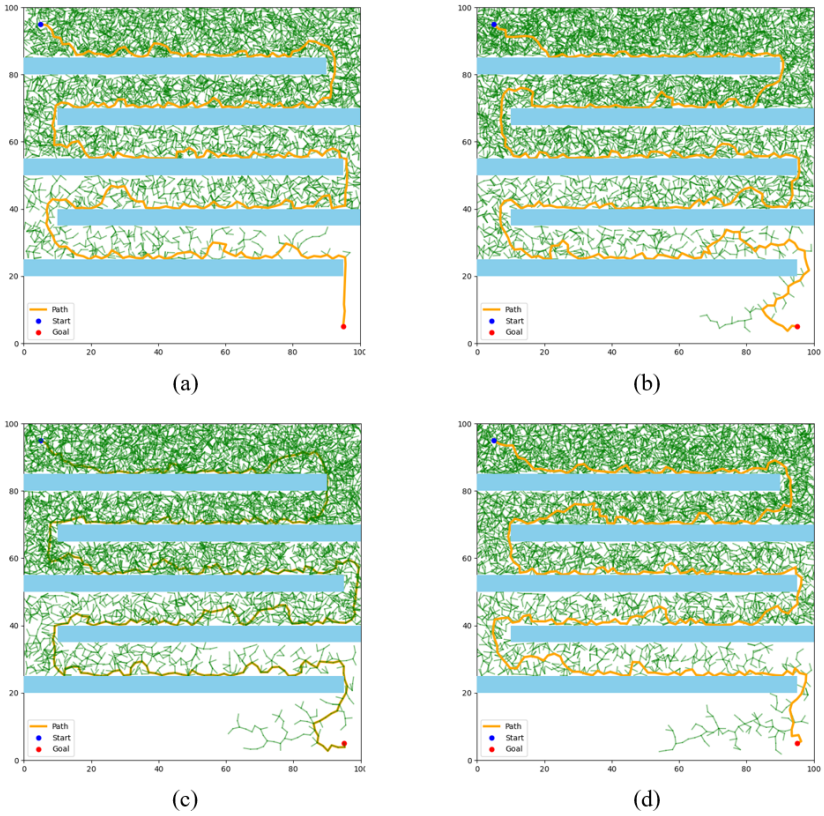


Figure 6. Path diagrams of four algorithms: (**a**) EG-DRRT*, (**b**) RRT, (**c**) RRT*, and (**d**) RRT-Connect in a complex environment.

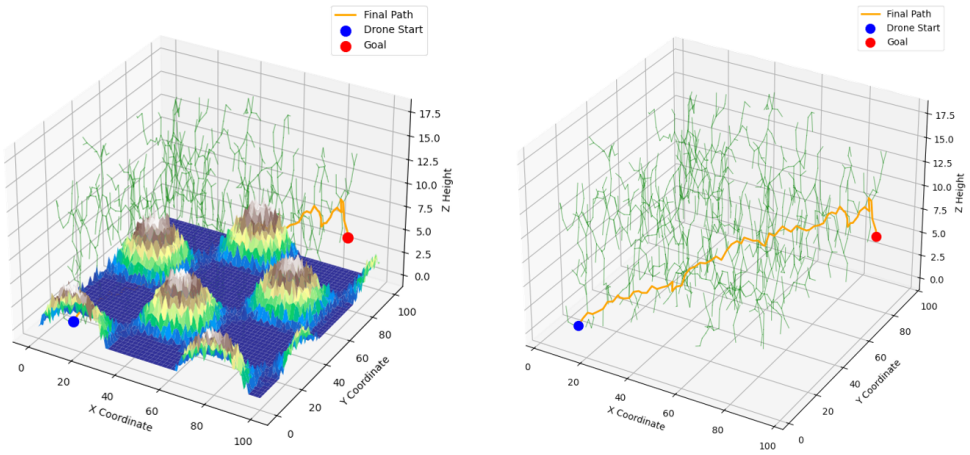
As shown in Table 2, the computation time of EG-DRRT* is reduced by 74.1% compared to RRT*. Additionally, the path length is shortened by 2.45% compared to RRT, 0.41% compared to RRT*, and 1.19% compared to RRT-Connect. Even when employing heuristic search and bi-directional expansion, EG-DRRT* remains superior. The number of iterations is reduced by 3.72% compared to RRT* and by 5.99% compared to RRT-Connect, although this reduction is slightly greater than that of RRT. Overall, the time efficiency and quality of paths produced by EG-DRRT* demonstrate its superiority in complex environments.

Table 2. The experimental data on four algorithms tested in a complex environment.

Algorithm Category	Time/s	Length/m	Iterations
EG-DRRT*	19.38	596.13	17,764.8
RRT	19.86	611.09	17,742.7
RRT*	74.82	598.59	18,450.3
RRT-Connect	35.79	603.33	18,896.2

4.3. Three-Dimensional Environment

In a 100 × 100 × 20 3D environment, the starting point is at (10, 10, 0), and the target point is at (90, 90, 5). There are column obstacles with heights of 3, 4, and 5 positioned at (50, 50), (30, 40), and (70, 70), respectively. Figure 7 illustrates the path-planning results of the EG-DRRT* algorithm in this 3D environment. The path successfully navigates through the irregularly distributed obstacle regions and reaches the target point, demonstrating the effectiveness of the algorithm in a 3D setting.



(a) (b)

Figure 7. The EG-DRRT* path in a 3D environment: (**a**) displayed obstacles; (**b**) concealed obstacles.

The random tree generated by EG-DRRT* is sparser, as shown in Figure 8. This sparsity leads to a reduction in redundant nodes, lowers search complexity, and enhances the efficiency of the expansion strategy. According to Table 3, the time taken by EG-DRRT* is reduced by 23.1% compared to RRT, 58.9% compared to RRT*, and 31.2% compared to RRT-Connect. Additionally, the path length is shortened by 6.9%, 3.2%, and 1.5%, respectively. This indicates that EG-DRRT* maintains the fastest search speed and the shortest path length in 3D path planning. While the number of iterations is slightly higher than that of RRT, it is still lower by 20.7% and 19.5% when compared to RRT* and RRT-Connect, respectively. This demonstrates EG-DRRT*'s effectiveness in controlling the number of node extensions in a 3D environment.

Table 3. The experimental data on four algorithms tested in a 3D environment.

Algorithm Category	Time/s	Length/m	Iterations
EG-DRRT*	6.42	157.78	4217.6
RRT	8.35	169.52	3727.7
RRT*	15.63	163.06	5317.8
RRT-Connect	9.33	160.25	5236.8

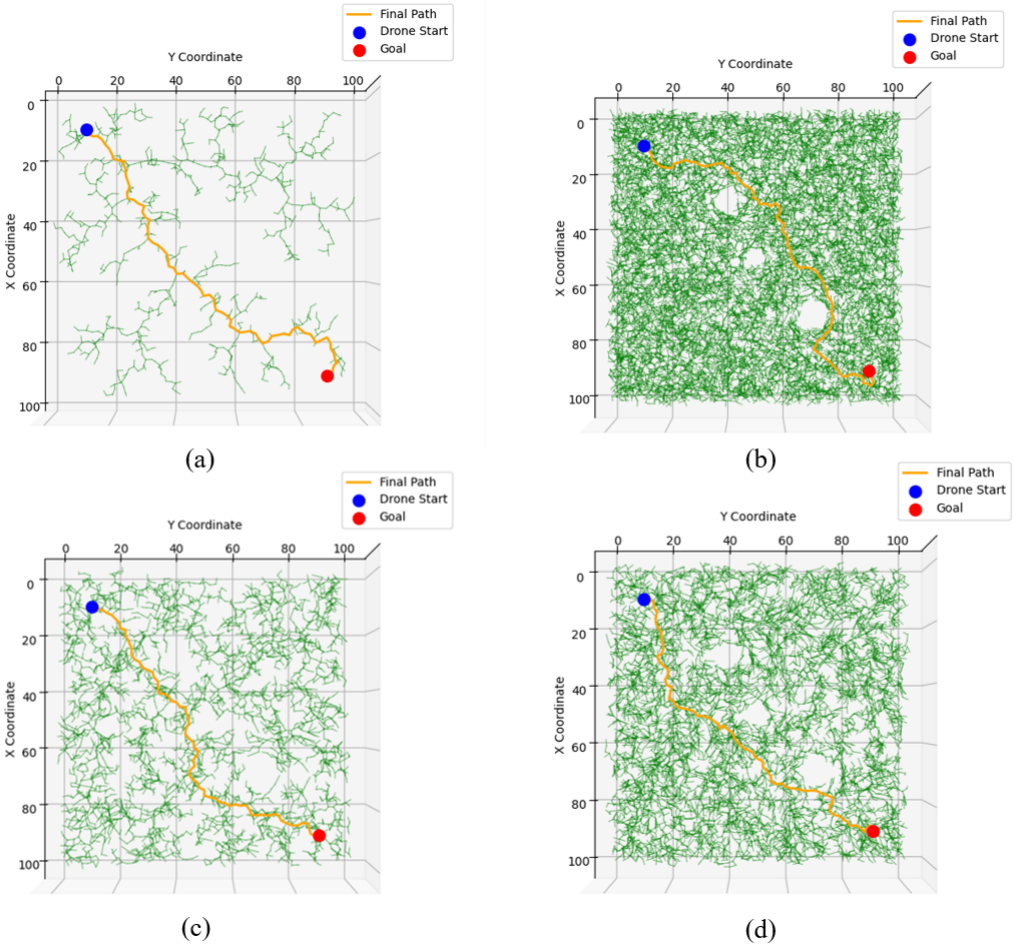


Figure 8. The top view of the four algorithmic paths: (**a**) EG-DRRT*, (**b**) RRT, (**c**) RRT*, and (**d**) RRT-Connect in a 3D environment.

5. Conclusions

Our proposed EG-DRRT* algorithm performs simulation experiments in three different environments by integrating evolutionary game theory, which emphasizes dynamically changing adaptations and strategies. The algorithm shows significant improvements in search time for simple environments, achieving an average path-length reduction of 5.0%. In complex environments, it reports an impressive average search-time reduction of 39.9% and a path-length reduction of 1.35%. For 3D environments, the search time is reduced by an average of 37.7%, while the average path length decreases by 3.9%. Overall, the expected results in terms of path dimension, complexity, time, and path length have been successfully met, effectively enhancing traditional RRT-series algorithms that typically exhibit poor dynamic adaptability and insufficient robustness in various environments.

While the algorithm has made notable strides in improving path-planning efficiency and quality, the fetch-random-point-type algorithm remains influenced by its initial parameter settings. Future studies will include experimental validation on the PX4 open-source

UAV platform, along with the ROS operating system, building on the current research conducted within a simulation environment. Future theoretical research will concentrate on further optimizing the evolutionary game theory model, developing a more refined strategy for calculating payoff functions, and exploring the creation of path-replanning mechanisms in dynamic obstacle environments. These efforts aim to augment the versatility and practical application of the EG-DRRT* algorithm for unmanned systems.

Author Contributions: Conceptualization, L.Q. and Y.H.; methodology, L.Q. and L.Y.; literature survey, L.Q., L.Y. and M.L.; data curation, L.Y. and M.L.; formal analysis, L.Q. and Y.H.; writing—original draft preparation, L.Q.; writing—review and editing, L.Q. and Y.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

Acknowledgments: The authors wish to thank the anonymous reviewers for their constructive comments that helped improve the scholarly quality of the paper.

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

Abbreviations

The following abbreviations are used in this manuscript:

EG-DRRT*	Evolutionary Game-Theoretic Dynamic RRT*
RRT	Rapidly Exploring Random Tree
RRT*	Rapidly Exploring Random Tree Star
UAVs	Unmanned Aerial Vehicles
ESS	Evolutionarily Stable Strategy

References

1. Aggarwal, S.; Kumar, N. Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges. *Comput. Commun.* **2020**, *149*, 270–299. [CrossRef]
2. Sanchez-Ibanez, J.R.; Pérez-del Pulgar, C.J.; García-Cerezo, A. Path planning for autonomous mobile robots: A review. *Sensors* **2021**, *21*, 7898. [CrossRef]
3. Yin, C.; Xiao, Z.; Cao, X.; Xi, X.; Yang, P.; Wu, D. Offline and online search: UAV multiobjective path planning under dynamic urban environment. *IEEE Internet Things J.* **2017**, *5*, 546–558. [CrossRef]
4. Lin, Y.; Saripalli, S. Sampling-based path planning for UAV collision avoidance. *IEEE Trans. Intell. Transp. Syst.* **2017**, *18*, 3179–3192. [CrossRef]
5. LaValle, S.M. *Planning Algorithms*; Cambridge University Press: Cambridge, UK, 2006.
6. 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]
7. Arslan, O.; Tsiotras, P. Use of relaxation methods in sampling-based algorithms for optimal motion planning. In Proceedings of the 2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, 6–10 May 2013; pp. 2421–2428. [CrossRef]
8. Arulkumaran, K.; Deisenroth, M.P.; Brundage, M.; Bharath, A.A. Deep reinforcement learning: A brief survey. *IEEE Signal Process. Mag.* **2017**, *34*, 26–38. [CrossRef]
9. Jones, M.; Djahel, S.; Welsh, K. Path-planning for unmanned aerial vehicles with environment complexity considerations: A survey. *ACM Comput. Surv.* **2023**, *55*, 1–39. [CrossRef]
10. Choset, H.; Lynch, K.M.; Hutchinson, S.; Kantor, G.A.; Burgard, W. *Principles of Robot Motion: Theory, Algorithms, and Implementations*; MIT Press: Cambridge, MA, USA, 2005.
11. LaValle, S. *Rapidly-Exploring Random Trees: A New Tool for Path Planning; Research Report*; TR 98-11; Computer Science Department, Iowa State University: Ames, IA, USA, 1998.

12. Karaman, S.; Frazzoli, E. Sampling-based algorithms for optimal motion planning. *Int. J. Robot. Res.* **2011**, *30*, 846–894. [CrossRef]
13. Kuffner, J.J.; LaValle, S.M. RRT-connect: An efficient approach to single-query path planning. In Proceedings of the 2000 ICRA, Millennium Conference, IEEE International Conference on Robotics and Automation, San Francisco, CA, USA, 24–28 April 2000; Symposia Proceedings (Cat. No. 00CH37065); IEEE: Piscataway, NJ, USA, 2000; Volume 2, pp. 995–1001. [CrossRef]
14. Marcucci, T.; Petersen, M.; von Wrangel, D.; Tedrake, R. Motion planning around obstacles with convex optimization. *Sci. Robot.* **2023**, *8*, eadf7843. [CrossRef] [PubMed]
15. Wang, B.; Liu, Z.; Li, Q.; Prorok, A. Mobile robot path planning in dynamic environments through globally guided reinforcement learning. *IEEE Robot. Autom. Lett.* **2020**, *5*, 6932–6939. [CrossRef]
16. Choudhury, S.; Scherer, S.; Singh, S. RRT*-AR: Sampling-based alternate routes planning with applications to autonomous emergency landing of a helicopter. In Proceedings of the 2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, 6–10 May 2013; pp. 3947–3952. [CrossRef]
17. Dang, X.; Edelkamp, S. SIL-RRT*: Learning Sampling Distribution through Self Imitation Learning. *arXiv* **2024**. [CrossRef]
18. He, Y.; Hou, T.; Wang, M. A new method for unmanned aerial vehicle path planning in complex environments. *Sci. Rep.* **2024**, *14*, 9257. [CrossRef] [PubMed]
19. Elbanhawi, M.; Simic, M. Sampling-based robot motion planning: A review. *IEEE Access* **2014**, *2*, 56–77. [CrossRef]
20. Dhulkefl, E.; Durdu, A.; Terzioğlu, H. Dijkstra algorithm using UAV path planning. *Konya J. Eng. Sci.* **2020**, *8*, 92–105. [CrossRef]
21. Soltani, A.R.; Tawfik, H.; Goulermas, J.Y.; Fernando, T. Path planning in construction sites: Performance evaluation of the Dijkstra, A*, and GA search algorithms. *Adv. Eng. Inform.* **2002**, *16*, 291–303. [CrossRef]
22. Zeng, W.; Church, R.L. Finding shortest paths on real road networks: The case for A. *Int. J. Geogr. Inf. Sci.* **2009**, *23*, 531–543. [CrossRef]
23. Smith, J.M.; Price, G.R. The logic of animal conflict. *Nature* **1973**, *246*, 15–18. [CrossRef]
24. Sandholm, W.H. *Population Games and Evolutionary Dynamics*; MIT Press: Cambridge, MA, USA, 2010.