Journal of Soft Computing and Computer Applications

Manuscript 1009

Improved Rapidly-Exploring Random Tree using Firefly Algorithm for Robot Path Planning

Dena Kadhim Muhsen

Firas Abdulrazzaq Raheem

Yuhanis Yusof

Ahmed T. Sadiq

Faiz Al Alawy

JOURNAL OF SOFT COMPUTING AND COMPUTER APPLICATIONS 2024;1:1009

ORIGINAL STUDY

Improved Rapidly-Exploring Random Tree using Firefly Algorithm for Robot Path Planning

Dena Kadhim Muhsen [a,*], **Firas Abdulrazzaq Raheem** [b], **Yuhanis Yusof** [c], **Ahmed T. Sadiq** [d], **Faiz Al Alawy** [e]

[a] University of Technology – Iraq, Department of Computer Science, Al-Sina'a St., Al-Wehda District, 10066 Baghdad, Iraq
[b] University of Technology – Iraq, Department of Control and Systems Engineering, Al-Sina'a St., Al-Wehda District, 10066 Baghdad, Iraq
[c] Universiti Utara Malaysia, School of Computing, Sintok, Kedah, 06010, Malaysia
[d] University of Technology Iraq, Department Computer Science, 10066 Baghdad, Iraq
[e] Kent State University, College of Engineering, Ohio, 44240 U.S.A.

ABSTRACT

In robotics, efficient path planning makes robots work independently and move through changing environments over time. This study combines the Rapidly-exploring Random Tree (RRT) architecture with the Firefly Algorithm (FA) to make robot's path-planning better. The proposed ERRT-FA, which stands for "Enhanced RRT with Firefly Algorithm", generates better routes using Firefly social habits. Plan routes using Firefly social habits can effectively aid in exploring configuration space. The role of the FA is to enhance the RRT algorithm by providing an optimized exploration of the search space, ultimately leading to optimizing the path found by the RRT algorithm and better paths in complex environments. The basic idea of the FA is to refine the resulting path by the RRT algorithm through optimizing the positions of Fireflies based on their intensity. Various tests show that ERRT-FA works better than the RRT algorithm in many robotic situations. It indicates a significant reduction in computation time, exploration efficiency, and route length, with statistical analysis showing a mean decrease. Such a result denotes that the proposed ERRT-FA is an alternative solution for optimizing ERRT-FA as a perfect path plan.

Keywords: Robot path planning, Rapidly exploring random tree, Firefly algorithm, Robustness

Received 4 May 2024; accepted 4 August 2024.
Available online 31 December 2024

* Corresponding author.
E-mail addresses: dena.k.muhsen@uotechnology.edu.iq (D. K. Muhsen), firas.a.raheem@uotechnology.edu.iq (F. A. Raheem), yuhanis@uum.edu.my (Y. Yusof), ahmed.t.sadiq@uotechnology.edu.iq (A. T. Sadiq), falalaw@kent.edu (F. Al Alawy).

https://doi.org/10.70403/3008-1084.1009

1. Introduction

Robots have the ability to navigate through dynamic and complex environments by strategically devising their routes. For this to be successful, all possible and best ways must be carefully considered, taking into account time, distance, and safety. Rapidly-exploring Random Tree (RRT) algorithms are famous for how quickly and correctly they can move through configuration spaces and develop appropriate paths [1]. Even though they have many benefits, earlier versions of RRT algorithms have issues in complicated situations changing variables, limited paths, and obstacles. Most stochastic roadmaps for the RRT method examine the configuration space using a progressive tree structure [2]. As the tree grows closer to an arbitrary configuration point, a study is being carried out. It is possible that unpredictability will cause the convergence time to slow down and the steps' quality to worsen in scenarios where time and accuracy are very important [3]. This study solves these problems with a new approach by integrating the RRT algorithm with the Firefly Algorithm (FA), a metaheuristic optimization method based on the flickering behavior of fireflies. The FA was proposed by [4], who got the idea from the bioluminescence that fireflies show. The fluctuating patterns exhibited by fireflies are linked to the attractiveness of their mates. The FA performs exceptionally well with robots and other optimization problems due to its adaptability [5]. It encourages teamwork among Fireflies to find optimal solutions. The suggested method fixes the problems with the basic RRT algorithm and makes the most of the findings about the RRT framework and the FA. An increase in the firefly's intensity results in a more precise response. The potential integration of the RRT and FA can enhance the efficiency, robustness, and adaptability of path planning. It enables the establishment of a foundation for enhanced autonomous guidance by integrating the most advantageous aspects of both algorithms. This would increase the apparatus's speed and precision during navigation in challenging conditions. The current study aims to enhance the RRT algorithm through its integration with the FA. The objective is to improve the velocity and precision of trajectory determination for point-mass robot model robotic systems functioning in various surroundings.

The contributions of this study can be summarized as follows:

1. A new proposed algorithm: A new enhanced RRT algorithm based on the meta-heuristic FA has been proposed and named Enhanced RRT with Firefly Algorithm (ERRT-FA).
2. Reduced computational effort: This is accomplished through overcoming many nodes, path length, and more time-consuming issues found in RR. Here, ERRT-FA generates better path planning than the basic RRT algorithm.
3. Evaluation of ERRT-FA: The purpose of this study is to evaluate the efficacy and effectiveness of robotic system path planning by integrating numerical analyses and simulations.

This study is organized as follows: Section 2 presents a detailed discussion of studies involving the research that has been done on robotic versions of the FA and RRT algorithms. Section 3 illustrates how the theoretically-driven RRT. Section 5 explains the basic idea of FA. Section 5 goes into more detail about the possible benefits of the proposed ERRT-FA improved RRT-Firefly algorithm while details of the evaluation are included in Section 6. Lastly, Section 7 concludes the main findings of the study.

2. Theoretical background

The focus on improving FA-based mobile robot path planning has been shown recently in [6]. During iteration, the authors found a discrepancy between Fireflies and barriers.

The authors solved this with a Firefly algorithm that self-adjusts population size. They corrected with nonlinear functions and evaluated collision degree to determine population size. Fireflies were added or eliminated to prioritize workable alternatives. The authors included a coefficient to control the distance between unfeasible and workable paths. Results showed better solution stability, faster convergence, and shorter running time than the Firefly algorithm with a secure population size.

An adaptive dynamic FA was proposed for mobile robot path planning in [7]. The authors overcame the FA inaccuracy and slow convergence to local optima since route planning is crucial for mobile robot studies. They carefully optimized adaptive parameters to construct an adaptive FA for path planning. Both theoretical and experimental validation showed that the optimized strategy significantly increased the mobile robot's processing and reaction speed. Their investigation revealed that the suggested algorithm for mobile robot path planning was the most efficient and effective.

Path planning algorithms that help self-driving cars were proposed in [8] and named Unmanned Aerial Vehicles (UAVs). The mobile robots find safe and quick ways without causing any accidents. This study shows how important it is to select methods that work with the system's constraints and geometry, which makes it useful for beginners. This study observes many different algorithms and provides valuable comparisons and use case examples to assist individuals in selecting the appropriate algorithms. This survey discusses path planning methods in detail, making them reliable for robotics engineers and embedded system developers.

In a recent study [9], the authors illustrated how to use a robotic limb to navigate problem nodes by adding a gravitational potential function and using B-spline curves to smooth out the trajectory. This study improved the RRT algorithm that was first created. Other algorithms are better than RRT in both two-dimensional and three-dimensional tests. The new RRT algorithm works better than the previous version of the algorithm one, as shown by execution times of 0.39 seconds and average path lengths of 147.63 millimeters. This study proves that the updated algorithm can easily find the best paths for robotic arms. This study demonstrates the criticality of devising innovative approaches to enhance algorithms, thereby improving the performance of computer arms across diverse scenarios.

Recently, FA step-kinematic has been evaluated in [10] as a potentially practical method for controlling and planning mixed motion. The study consists of kinetic equations and stepping ahead Firefly. The authors use robotic navigation to handle crowded areas. When obstacles are detected, the system instantly switches between equation-based mobility and Firefly stepping. This will lead to self-driving ways that are safer and work better. Path length improved by 5.79 %, 0.37 %, and 9.22 %, showing that it works better than Ant Colony Optimization (ACO)-kinematic. In this way, the method becomes useful and more valuable.

In [11], the authors came up with a new way of kinematics in planar robots. The extended Jacobian and FA are used to make it work. The PUMA 560 robotic arm is used as an example. This study uses inverse kinematics to show how defect tolerance can be raised in places where maintenance is expected. The Firefly and Jacobian algorithms are combined to make this possible. This study shows how fabrication processes could be improved through operational flexibility and advanced robotic design options. Increasing understanding of how to improve the performance and durability of planar bots enables their application in a broader range of environments.

The authors in [12] explained the inverse kinematics problem of a 7-Degree of Freedom (7-DOF) redundant robot manipulator using the Firefly method, a swarm optimization methodology. The FA was tested on a redundant robotic arm with unsatisfactory inverse kinematic solutions using traditional methods. It was compared to artificial bee colony and particle swarm optimization in speed and accuracy. The FA minimized joint

angle identification error; forward kinematic calculation determined the end effector's workspace location, and the manipulator position equations were derived. The FA performed complex robotic manipulation problems efficiently and accurately.

This study combined the FA with the RRT for several reasons. The FA's simplicity and flexibility allow it to handle multimodal optimization problems. It is ideal for increasing local optimization in path planning algorithms like RRT because Fireflies generate optimal solutions. Even though the Genetic Algorithm (GA) [13], ACO [14], and Particle Swarm Optimization (PSO) [15] have been widely deployed in the literature, each of these alternatives has its downsides. Hence, making the FA the better optimization algorithm. GA's computational overhead is not necessarily proportionate to its strength because it sometimes requires intricate mutation and crossover procedures. In discrete contexts like RRT-based path planning, PSO may not be as effective as in continuous search spaces. ACO is useful in some pathfinding applications, despite its intensive computations and the need for substantial parameter modifications.

FA is a versatile, easy-to-use, and low-computing approach. When paired with RRT, the enhancement in exploration and exploitation lead to an improve in path planning. The FA may dynamically change firefly brightness and appeal to improve computing efficiency and path optimality by fine-tuning RRT paths. FA being hybridized with RRT emphasizes the goal of producing a powerful and efficient path-planning algorithm for complex real-time environments.

3. Rapidly exploring random tree path planning

Many point-mass robot models use the RRT algorithm to plan their routes because it quickly and accurately builds paths and explores configuration space. This is the robot's first configuration, which is where its RRT technique story starts [16]. The process then creates a tree structure by growing the value over and over to any point in the configuration space. The algorithm steps are as follows:

Algorithm 1. RRT

1. Set up the tree T from scratch using the default settings q_init
2. K = number of samples in random
3. For k = 1 to K Do
   4. Within the configuration space, generate a random configuration q_rand
   5. Find the nearest node q_near in the tree T to q_rand using the Euclidean distance formula shown in Eq. (1):

$$d(q_{near'} q_{rand}) = \sqrt{\sum_{k=1}^{K} (q_{near'_k} q_{rand_k})^2} \tag{1}$$

   6. Cover the tree from q_near towards q_rand with a step size δ, generating a new node q_new
   7. If the path from q_near to q_new is collision-free then enhance q_new to the tree T
8. End for
9. Return the tree T

Due to the use of random sampling, the basic RRT method can quickly cover a large area and provide useful paths. Routes might not work as well in very small places or with many obstacles. For areas with many dimensions, it may take a long time for the algorithm to find the best results because it works with a high degree of randomness [17, 18].

Fig. 1. Firefly in nature [20].

4. Firefly algorithm

Metaheuristic optimization, inspired by how Firefly wings flap, as shown in Fig. 1, was used in the first versions of the FA. Firefly social ties are being modeled to solve efficiency problems. The technique shows possible results for optimization problems involving Fireflies. The quality of the results is demonstrated by how bright or intense the Fireflies are [19]. The main concepts about the method are its appeal and randomness. Because of stochasticity, Fireflies can explore the solution space by moving around randomly during the trial. Every time the algorithm runs, Fireflies move based on a random factor and the draw of other Fireflies in the area. Along with looking for more busy partners, Fireflies also cause random changes in motion. When an end condition is met, a process is stopped and started again. It could be the number of attempts or the quality of the result. FA is characterized by several positive qualities, including simplicity, effectiveness, and durability. Setting up and handling problems with discrete, combinatorial, and continuous optimization is easy [20, 21]. The method can handle a wide range of issue aspects and levels of complexity, and it is very responsive to changes in parameter values. A known tool for improving the FA is based on how Fireflies interact with each other as shown in Pseudo-code 1. The goal of this tool is to make it easier to explore the solution area and find the best option [4, 22]. The light intensity is calculated in Eq. (2), where I_0 is the light

Pseudo-code 1. FA.

```
Objective function f(x), x = (x_1, ..., x_d)^T
Generate initial population of Fireflies x_i (i = 1, 2, ..., n)
Light intensity Ii at xi is determined by f(x_i)
Define light absorption coefficient
while (t < Max Generation)
for i = 1: n (all n Fireflies)
    for j = 1: i (all n Fireflies)
        if (I_j > I_i), Move Firefly i towards j in d-dimension by Eq. (5).
        End if
        Attractiveness varies with distance r via exp[−γr^2]
        Evaluate new solutions and update light intensity
    end for j
end for i
Rank the Fireflies and find the current best
end while
Postprocess results and visualization
```

intensity at the beginning:

$$I_r = I_0 e^{-\gamma r_{ij}^2} \tag{2}$$

The brightness (attraction) of Fireflies is calculated in Eq. (3)

$$\beta_i = \beta_0 e^{-\gamma r_{ij}^2} \tag{3}$$

where r_{ij} is the Euclidian distance between two Fireflies x_i and x_j, and β_0 signifies the initial attraction factor, which is typically 1. Parameter γ is the fixed value of the light absorption coefficient, which is generally also 1. For 2 Fireflies x_i and x_j randomly chosen in the identical search space, the distance r_{ij} among them is considered as Eq. (4):

$$r_{ij} = \sqrt{\sum_{s=1}^{n} (x_i - x_j)^2} \tag{4}$$

where n expresses the dimension of the problem, and s denotes the s-th dimension of the position vector of the Firefly. Throughout the search, Firefly x_i is involved with the brighter Firefly x_i and moves towards x_i. The location of Firefly is calculated according to Eq. (5):

$$x_i(t+1) = x_i(t) + \beta_i(x_j(t) - x_i(t)) + \alpha.\ (rand - 0.5) \tag{5}$$

where $x_i(t)$ and $x_j(t)$ represent the positions of Fireflies i and j at time t, respectively, the term $\beta_i(x_i(t) - x_i(t))$ models the attraction of Firefly_i to Firefly_j, which is based on the relative brightness and the distance between them. The coefficient α controls the randomness of the movement and (rand- 0.5) is a random number drawn from a uniform distribution. t signifies the number of iterations of the algorithm, α is the step size parameter, which is typically a random number with the value of [0,1].

5. An enhanced rapidly-exploring random tree

Point-mass robot models plan their routes using many algorithms to navigate complicated environments, avoid obstacles, and finish already-set jobs. They often use the RRT method to plan paths because it can quickly search configuration spaces and identify feasible paths. However, the environment can get hard and change quickly, so the basic RRT method might have issues like poor path quality and slow convergence.

The RRT method incorporates the FA to increase robotic system path planning efficiency and dependability, as shown in Pseudo-code 2. This integration involves the following steps:

1. **Initialization**
   Start and goal nodes should be set up in the supplied environment to initialize the RRT. Firefly spots are randomly set in the search space. A diagram shows a map with the start and goal nodes marked, along with several Fireflies (points) scattered randomly within the map.
2. **Random Node Generation**
   The algorithm randomly generates nodes in the search space. These nodes serve as potential waypoints for constructing the path from the start to the goal. A map with a few randomly placed nodes (potential waypoints) was highlighted.

Pseudo-code 2. ERRT- FA.

Input:
q_{start} starting point
q_{goal} target point
l step length
C all frontier point locations in all obstacles (known)
R number of samples in random
Output:
S resulting path
Initialization:
T *Null* tree<node, edge>
F List of fireflies
Begin
T ← **Add** root<qstart>
For r = 1 **To** R **Do**
Create r-th sample in random
q_{rand} node ← position of the r-th sample in random
q_{near} node ← position of the closest node in T from q_{rand} node
If not within (q_{near} node, q_{rand} node, l) **Then**
q_{new} node ← juncture point between line segment linking q_{rand} node and q_{near} n node, and circle which radius is l centered at q_{near} node
Else q_{new} node ← q_{rand} node
If not strapped (q_{new} node, q_{near} node, C) **Then**
T ← **Add** node<q_{new} node> & edge<q_{new} node, q_{near} node>
End if
End if
If within (q_{new} node, q_{goal}, l) **Then**
T ← **Add** node<q_{goal}> & edge<q_{new} node, q_{goal}>
P ← path from last added node{q_{goal}} to root node{q_{start}} in T
End if
If [length of S] > [length of P], **Then** S ← P
Remove node<q_{goal}> & edge<q_{new} node, q_{goal}> from T
End if
Find the optimize solution (S) by call FA (Pseudo code 1)
If is within (solution for firefly, qgoal, l) Then
T ← Add node <solution for firefly> & edge <solution for firefly, qgoal>
P ← path from last added node <qgoal> to root node <qstart> in T
End if
If length of S > length of P, Then S ← P
Remove node <solution for firefly> & edge <solution for firefly, qgoal> from T
Return S
End if
End for
End

3. **Nearest Neighbour Search**
   The algorithm identifies the nearest existing node in the RRT for each newly generated node. This step is crucial for extending the tree towards the new node. A zoomed-in section of the map shows the freshly generated node and the nearest existing node connected by a dashed line, indicating the search process.
4. **Tree Expansion**
   The tree is expanded by connecting the nearest node to the new node if the path between them is collision-free. This iterative method continues until the tree is close to the goal node. The map shows the growing tree structure, with branches extending from the start node towards the goal, highlighting collision-free paths.

5. **Firefly Attraction Mechanism**
   The FA is employed to optimize the path. Fireflies are attracted to brighter (better) solutions, guiding the path optimization process. The brightness of a Firefly is resolute by the quality of the solution (path) it represents. A sequence of images shows Fireflies moving towards brighter Fireflies, with the brightness being indicated by varying sizes or colours of the Fireflies.
6. **Path Optimization**
   The positions of the Fireflies are updated iteratively based on the attraction mechanism, resulting in an optimized path that minimizes distance and avoids obstacles. A comparison diagram shows the initial unoptimized path and the final optimized path, highlighting the improvements in path quality.
7. **Convergence Check**
   The algorithm checks if the tree has reached the goal node. The process terminates if the goal is reached or a satisfactory path is found. Otherwise, it continues with further iterations.

The FA is called under conditions that ensure the computational effort leads to substantial improvements. This approach avoids redundant calculations and ensures that computational resources are utilized efficiently. ERRT-FA leverages the strengths of FA while mitigating its computational cost through strategic implementation and optimization techniques. The research demonstrates that ERRT-FA offers improved performance in practical scenarios, showing that the benefits outweigh the theoretical computational costs. This strategic implementation of ERRT-FA reassures the audience about its effectiveness.

6. Result and discussion

Regarding the point-mass robot model, Table 1 shows how the RRT and the ERRT-FA stack up against each other. Each method indicates the path, the time it takes to calculate, and the tree nodes. ERRT-FA always finds a way that is faster than an RRT route. When the FA is used, it finds faster and more direct routes between the starting and the finishing points. This helps the RRT framework to plan optimal paths. In contrast to the RRT method, the ERRT-FA method is able to find the best paths more quickly due to its shorter average path length. Small hallways and barriers that are hard to get around require critical attention. ERRT-FA often makes trees with fewer nodes than RRT. For this reason, fewer nodes are needed to look into the configuration space and find suitable ways. ERRT-FA finds the best lines with fewer iterations and less computer work, showing that it is a better exploration method. Because fewer points need to be observed, the vast majority of the time ERRT-FA does calculations faster than RRT. This shows that ERRT-FA may be able to keep a higher number of nodes and a longer path while using less computing power. This reduction is significant for ERRT-FA as it is necessary to decrease computation time for real-time apps that need to plan paths quickly. It improves the planning systems' performance by optimizing their actions and paths.

The differences between ERRT-FA and RRT are shown in Fig. 2. The path starts and ends between point (2,5) and point (6,9). The path ERRT-FA makes is shorter and straighter than that of the RRT. The configuration space ERRT-FA has searched is shown by a tree with fewer nodes. Given that it takes less time to compute, the ERRT-FA method seems to work better than the RRT technique. Fig. 3 shows that the RRT and ERRT-FA methods are different using the starting point (8,2) and ending point (2,6) values. The ERRT-FA method has a shorter path and fewer tree branches than the RRT method; because it takes less time to compute than the RRT method, the ERRT-FA algorithm could speed up path planning.

Table 1. Comparison between RRT and ERRT-FA.

	RRT			ERRT-FA		
Starting and goal points	Path length	Number of nodes	Time	Path length	Number of nodes	Time
(2, 5)	101	1333	1.48	50	357	0.11
(6, 9)	87	901	0.7	44	346	0.1
	97	1515	1.96	41	238	0.4
	106	1358	1.54	41	488	0.3
	97	1213	1.2	44	325	0.1
Average	98	1264	1.37	44	351	0.2
(8, 2)	Path length	Number of nodes	Time	Path length	Number of nodes	Time
(2, 6)	95	1644	2.13	50	370	0.1
	92	969	0.8	43	375	0.1
	100	746	0.6	54	301	0.3
	99	798	0.5	43	384	0.2
	104	762	0.5	51	180	0.03
Average	98	984	0.9	48	322	0.1
(6, 4)	Path length	Number of nodes	Time	Path length	Number of nodes	Time
(7, 7)	56	942	0.7	43	589	0.2
	81	832	0.7	28	190	0.04
	67	1316	1.3	28	130	0.02
	64	752	0.5	27	301	0.08
	82	1399	1.6	30	394	0.1
Average	70	1048	0.96	31	321	0.08
(8, 3)	Path length	Number of nodes	Time	Path length	Number of nodes	Time
(4, 9)	113	747	0.5	55	352	0.1
	115	1213	1.2	50	445	0.1
	102	1805	2.75	68	459	0.1
	102	1690	2.31	64	629	0.3
	92	1778	2.57	69	470	0.1
Average	105	1447	1.86	61	471	0.1
(9, 1)	Path length	Number of nodes	Time	Path length	Number of nodes	Time
(2, 9)	144	1085	1.03	75	943	0.7
	144	1079	0.97	70	481	0.2
	176	1139	1.09	68	464	0.1
	136	1257	1.29	68	485	0.2
	145	1486	1.81	66	351	0.1
Average	149	1209	1.23	69	545	0.3

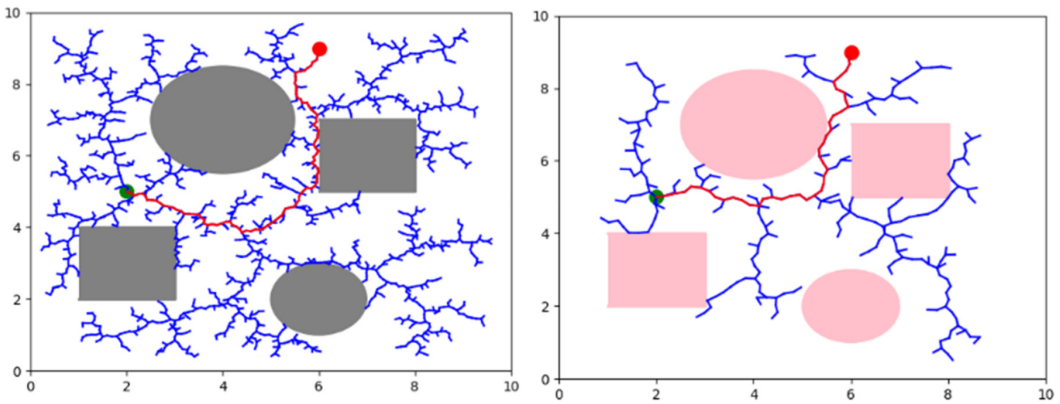

Fig. 2. The RRT algorithm vs. ERRT-FA algorithm for points (2,5) and (6,9).

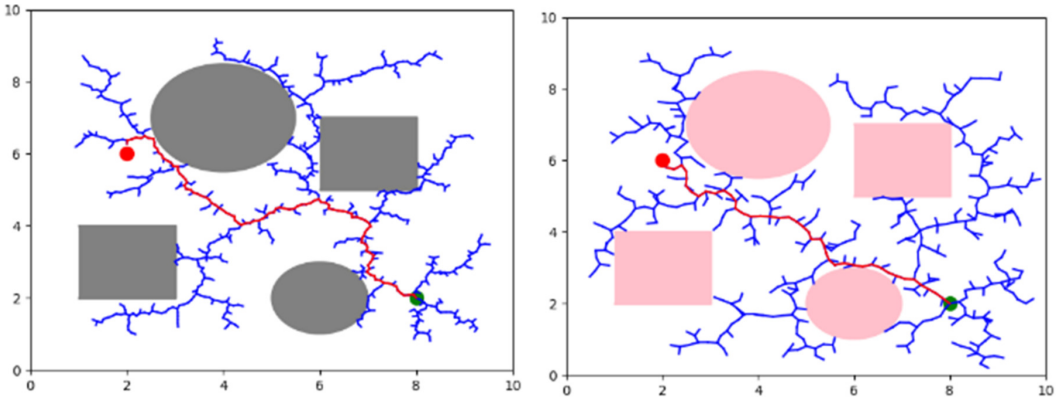

Fig. 3. The RRT algorithm vs. ERRT-FA algorithm for points (8, 2) and (2, 6).

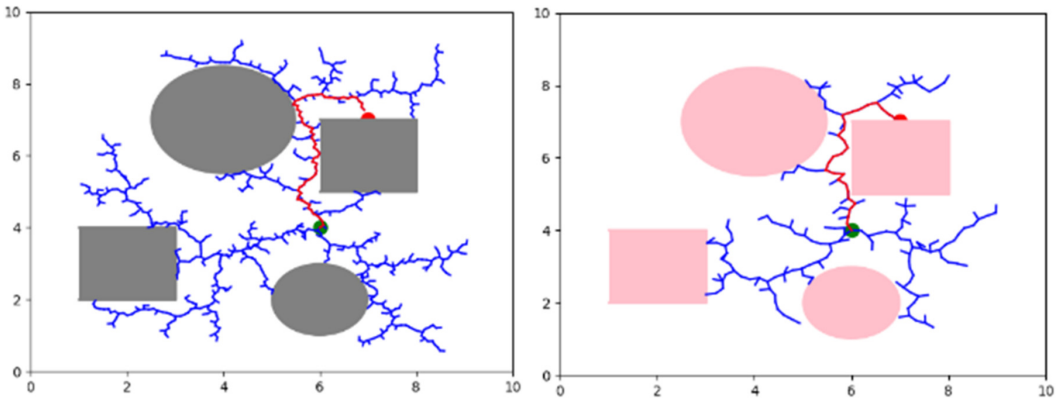

Fig. 4. The RRT algorithm vs. ERRT-FA algorithm for points (6, 4) and (7, 7).

In Fig. 4, the RRT method and the ERRT-FA algorithm can be tested at the starting point (6,4) and ending point (7,7). If the ERRT-FA method is compared with the RRT algorithm, it is shown that the ERRT-FA algorithm makes shorter and more direct routes. ERRT-FA tries to make the best use of setup space by reducing the number of nodes in the tree. The ERRT-FA algorithm performs calculations much faster than the RRT algorithm.

Fig. 5 shows how ERRT-FA and RRT are different. The line on the picture goes from point (8,3) to point (4,9). Regarding path length and tree nodes, ERRT-FA is better than RRT. Less time is spent on ERRT-FA computations, which speeds up path planning. In Fig. 6, both the starting point (9,1) and ending point (2,9), it is possible to compare the RRT method to the ERRT-FA algorithm. The graphs show that ERRT-FA has a shorter path and uses fewer tree nodes than RRT. Creating ERRT-FA took less time than other algorithms, so it is a better path-planning algorithm.

The ERRT-FA was evaluated in several conditions, including tight passages and obstacles. Figs. 2, 4 and 5, where the narrow passage between two obstacles, the ERRT-FA's confined-space navigation in an efficient way. The ERRT-FA proved to be reliable in tight locations with complex barrier layouts. Incorporating the RRT framework, the FA improves navigation and path optimization in complex environments.

The results of this study show that adding the FA to the RRT structure can make it work much better. Comprehensive simulations showed that the ERRT-FA algorithm, an

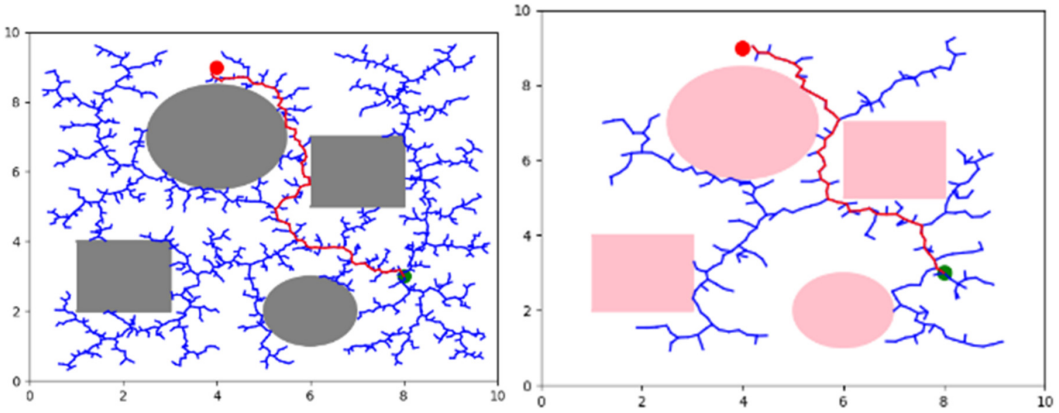

Fig. 5. The RRT algorithm vs. ERRT-FA algorithm for points (8, 3) and (4, 9).

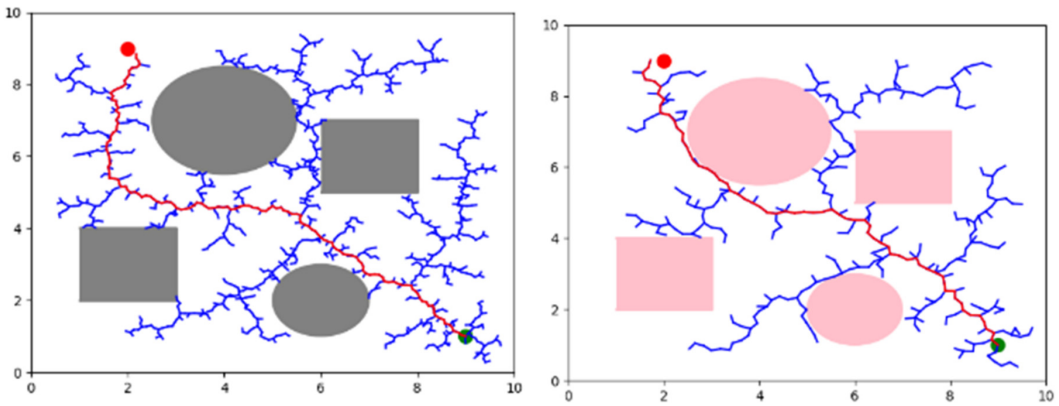

Fig. 6. The RRT algorithm vs. ERRT-FA algorithm for points (9, 1) and (2, 9).

improved version of the standard RRT algorithm, worked better in a wide range of robotic usages. In many cases, ERRT-FA found more direct and effective paths while using less computing power at different starting and ending places. This is evident as processing time and tree nodes are reduced because ERRT-FA is more efficient and effective; it has promising potential for robotic uses, especially in complicated environments that change quickly and need accurate path planning. By making these changes, situations like these would get better. ERRT-FA can find the best routes by using the appeal and randomness of Firefly movements. Exploration variety is kept up so that convergence does not happen too soon. The simulation results (Figs. 2 to 6) illustrate that the paths generated by the ERRT-FA exhibit some zigzag patterns. While these may initially suggest suboptimality, it is crucially essential to recognize the context in which “better” path planning is evaluated.

The ERRT-FA has been developed to streamline real-time path building and flexibility. Despite their disadvantages, zigzag routes benefit real-time applications because they allow rapid re-planning and obstacle avoidance. These routes prioritize avoiding obstacles and maintaining path length to ensure accurate navigation without the burden of processing. Local optimization reduces FA zigzagging compared to RRT. More direct routes would save time and energy, but ERRT-FA’s robustness in dynamically changing settings allows it to preserve viable paths, improving path planning.

Future development will improve route quality without hurting ERRT-FA's real-time performance, which may require post-processing to smooth pathways. When the environment changes, ERRT-FA can instantly adjust its search using the FA. It will last longer and perform better. Scalability difficulties in complex contexts may be examined. The FA simplifies robot path calculation in the RRT framework. Multipurpose tools may be useful, beneficial, adaptable, and efficient.

The RRT and ERRT-FA depend on the random nodes, and the distance between these nodes at each step is calculated, not the Euclidean distance between the start and endpoints.

Given the substantial disparity, what generated these results must be determined. Instead of computing the Euclidean distance from start to finish, methods like ERRT-FA and RRT create random nodes and iteratively calculate distances. This variation in approach shows how the algorithm relies on iterative path refining and node selection. The stochasticity of ERRT-FA explains the vast disparity between computed and tabular values. Due to node selection and local path alterations, this approach can report substantial path length changes. Given this disparity, ERRT-FA's accuracy in depicting optimal path solutions in the given scenario shows the intricate interaction between algorithmic design, node selection strategies, and path quality.

The study found that the ERRT-FA less rapidly explored random trees and used fewer nodes than RRT. The reduction in nodes directly affects path planning computation efficiency and performance, making it a crucial component that must be explained.

The RRT and FA reduce ERRT-FA's node count. The classic RRT approach incrementally builds a search tree from a starting point to a goal point, generating many nodes to analyze the entire configuration space. This method is computationally demanding due to the large number of nodes, yet it is excellent for complex investigations.

In contrast, ERRT-FA integrates FA's optimization into RRT. Similar to Firefly attraction, the FA converges on the best replies. FA is used by the ERRT-FA algorithm expansion to direct tree expansion to direct tree expansion intelligently. To clarify, FA expands trees by selecting more promising nodes, reducing the need for exploratory nodes. Targeting specific nodes ensures that the tree uses fewer nodes to get there faster without compromising path quality. Thus, reducing the number of nodes leads to a reduction in the computational cost. The computing costs of path planning algorithms depend on the number of nodes generated and analyzed. The method evaluates and stores each node, representing a configuration space state. By reducing node count, ERRT-FA reduces computational overhead from node formation, assessment, and storage.

FA optimization boosts efficiency. The FA can avoid repeated expansions by picking nodes with promising pathways using an attraction mechanism. This improves path planning and computational resource use. ERRT-FA's node reduction benefits are shown by comparing it to RRT. The traditional RRT uses random sampling to study various configurations. This random sampling can produce many nodes in complex or high-dimensional spaces, but it helps find plausible pathways. Random node selection adds processing overhead and duplicate expansions. FA combines RRT's exploration with FA's optimization. The FA's attraction mechanism favors nearby or optimum nodes. This tailored development reduces repetition therefore the tree grows well in complex environments. Because it uses fewer nodes to generate the same or better path quality, ERRT-FA has lower computational costs than RRT.

The study supports these theoretical benefits empirically. The simulation results show that ERRT-FA uses fewer nodes than the standard RRT while maintaining or improving path quality. This reduces the number of nodes, the processing time and resources, proving the efficiency benefits of FA hybridization. Lower nodes do not impair the algorithm's

resilience. ERRT-FA can adapt to changing conditions. Even with FA's optimization, the paths are practical and efficient, meeting safe and effective robot navigation standards. The FA intelligent node selection and optimization approaches minimize ERRT-FA's node count compared to standard RRT. This boosts computational efficiency since fewer nodes mean lower computing costs.

This study used the RRT algorithm and does not use RRT* because of its simplicity and high-dimensional configuration space exploration. The main goal was to increase RRT's path quality and computing efficiency with the FA. RRT* has asymptotic optimality, ensuring that the path converges to an optimal solution as computing resources increase. RRT was used to balance computational overhead and path development speed. RRT* may be too computationally intensive in environments with various obstacles for real-time applications that need quick decisions. The FA has been used to fix inefficient paths and other RRT issues without increasing RRT* computing demand. The ERRT-FA creates optimal paths faster than RRT alone and on par with more computationally intensive algorithms like RRT*, outperforming past hybrid studies. Because it balances efficiency and optimality, ERRT-FA is suitable for real-time path-planning applications with limited computational resources.

7. Conclusions and future work

This study tackled the integration of the FA into the RRT framework to improve the path planning for robots. The ERRT-FA outperforms RRT in many robotic settings. Even with limited computational capabilities, ERRT-FA delivers faster and more direct paths during high flux and pressure. Its production and efficiency are indicated. ERRT-FA improves by adapting its search method to changing environmental and mission conditions. This is achieved using the FA flexibility and robustness. Effective and reliable path planning is crucial for robotic applications, and ERRT-FA shows its usefulness. Knowing how ERRT-FA works in real life would help it perform better in simulations. RRT architecture with the FA improves artificial path planning systems. Thus, robots will be more efficient, effective, and versatile everywhere.

Future studies could investigate the efficacy and scalability of the ERRT-FA algorithm in handling big, complicated situations. More studies could also be done to fine-tune the parameters of the FA so that it works best on various robot platforms. Further investigation on how ERRT-FA can be combined with other metaheuristic algorithms could focus on improving path planning in terms of efficiency and resilience.

Acknowledgment

The authors thank the Department of Computer Science, University of Technology—Iraq, for administering this study.

Author contributions

Dena Kadhim Muhsen: Conceptualization, Methodology, Formal analysis, programming, Investigation, Writing – review and editing. Firas Abdulrazzaq Raheem: Conceptualization, Formal analysis, Data curation, Supervision, review and editing. Yuhanis Yusof: review and editing. Ahmed T. Sadiq: Conceptualization, Formal analysis, Data curation, Supervision, review and editing. Faiz Al Alawy: review and editing.

Conflict of interest

The authors declare no conflict of interest.

Data availability statement

Data sharing is not applicable to this article as no datasets were generated or analyzed during the current study.

References

1. F. Kiani *et al.*, "Adapted-RRT: novel hybrid method to solve three-dimensional path planning problem using sampling and metaheuristic-based algorithms," *Neural Computing and Applications*, vol. 33, no. 22, pp. 15569–15599, 2021, doi: 10.1007/s00521-021-06179-0.
2. L. Hong *et al.*, "Two-layer path planner for AUVs based on the improved AAF-RRT algorithm," *Journal of Marine Science and Application*, vol. 21, no. 1, pp. 102–115, 2022, doi: 10.1007/s11804-022-00258-x.
3. D. K. Muhsen *et al.*, "Memorized rapidly exploring random tree optimization (mrrto): an enhanced algorithm for robot path planning," *Cybernetics and Information Technologies*, vol. 24, Issue no. 1, pp. 190–204, 2024, doi: 10.2478/cait-2024-0011.
4. X. S. YANG, *Nature-Inspired Metaheuristic Algorithms*, 2nd ed. UK: Luniver Press, 2010.
5. Y. Wang *et al.*, "Research on path planning algorithm of intelligent inspection robot," *Journal of Physics: Conference Series*, vol. 2181, no. 1, 2022, Art. no. p. 012006, doi: 10.1088/1742-6596/2181/1/012006.
6. F. Li *et al.*, "A firefly algorithm with self-adaptive population size for global path planning of mobile robot," *IEEE Access*, vol. 8, pp. 168951–168964, 2020, doi: 10.1109/ACCESS.2020.3023999.
7. G. Xu *et al.*, "A new path planning method of mobile robot based on adaptive dynamic firefly algorithm," *Modern Physics Letters B*, vol. 34, no. 29, 2020, Art. no. 2050322, doi: 10.1142/S0217984920503224.
8. K. Karur *et al.*, "A survey of path planning algorithms for mobile robots," *Vehicles*, vol. 3, no. 3, pp. 448–468, 2021, doi: 10.3390/vehicles3030027.
9. L. Guo, "Robotic arms passing through obstacle nodes: using an improved intelligent algorithm," *International Journal of Mechatronics and Applied Mechanics*, no. 13, pp. 76–80, 2023.
10. K. Chaudhary *et al.*, "Robot motion control using stepping ahead firefly algorithm and kinematic equations," *IEEE Access*, vol. 12, pp. 43078–43088, 2024, doi: 10.1109/ACCESS.2024.3380004.
11. T. R. Prathab *et al.*, "A method of extended jacobian and firefly algorithm for the kinematic analysis of planar robots," *International Journal of Robotics and Automation*, vol. 6, no. 2, pp. 141–150, 2017, doi: 10.11591/ijra.v6i2.pp141-150.
12. S. Dereli and R. Köker, "Calculation of the inverse kinematics solution of the 7-DOF redundant robot manipulator by the firefly algorithm and statistical analysis of the results in terms of speed and accuracy," *Inverse Problems in Science and Engineering*, vol. 28, no. 5, pp. 601–613, 2020, doi: 10.1080/17415977.2019.1602124.
13. T. Alam *et al.*, "Genetic algorithm: reviews, implementations, and applications," *International Journal of Engineering Pedagogy*, vol. 10, no. 6, pp. 57–77, 2020, doi: 10.3991/ijep.v10i6.14567.
14. X. Dai *et al.*, "Mobile robot path planning based on ant colony algorithm with A* heuristic method," *Frontiers in Neurorobotics*, vol. 13, no. 15, pp. 1–9, 2019, doi: 10.3389/fnbot.2019.00015.
15. E. Krell *et al.*, "Collision-free autonomous robot navigation in unknown environments utilizing PSO for path planning," *Journal of Artificial Intelligence and Soft Computing Research*, vol. 9, no. 4, pp. 267–282, 2019, doi: 10.2478/jaiscr-2019-0008.
16. D. K. Muhsen *et al.*, "Improved rapidly exploring random tree using salp swarm algorithm," *Journal of Intelligent Systems*, vol. 33, no. 1, 2024, Art. no. 20230219, doi: 10.1515/jisys-2023-0219.
17. D. K. Muhsen *et al.*, "A Survey on swarm robotics for area coverage problem," *Algorithms*, vol. *17*, no. 1, Art. no. 3, 2024, doi: 10.3390/a17010003.
18. A. K. Sadhu *et al.*, "Synergism of firefly algorithm and Q-learning for robot arm path planning," *Swarm and Evolutionary Computation*, vol. 43, pp. 50–68, 2018, doi: 10.1016/j.swevo.2018.03.014.
19. L. Jun *et al.*, "A survey on firefly algorithms," *Neurocomputing*, vol. 500, pp. 662–678, 2022, doi: 10.1016/j.neucom.2022.05.100.

20. R. Nand *et al.*, "Preference-based stepping ahead firefly algorithm for solving real-world uncapacitated examination timetabling problem," *IEEE Access*, vol. 12, pp. 24685–24699, 2024, doi: 10.1109/ACCESS.2024.3365734.
21. M. M. Gangadharan and A. Salgaonkar, "Ant colony optimization and firefly algorithms for robotic motion planning in dynamic environments," *Engineering Reports*, vol. 2, no. 3, Art. no. e12132, 2020, doi: 10.1002/eng2.12132.
22. B. K. Patle *et al.*, "Path planning in uncertain environment by using firefly algorithm," *Defence Technology*, vol. 14, no. 6, pp. 691–701, 2018, doi: 10.1016/j.dt.2018.06.004.