Plan like Ants - Efficient path planning using RRT* and Ant Colony Optimization

Abhijay Singh
Maryland Applied Graduate Engineering
University of Maryland
College Park
abhijay@umd.edu

Tharun Puthanveettil
Maryland Applied Graduate Engineering
University of Maryland
College Park
tvpian@umd.edu

***Abstract*—Planning in contexts with many impediments has been successfully accomplished using rapidly exploring random trees (RRTs). These techniques undertake a uniform sampling of the state space, which is necessary to ensure the algorithm's completeness, but does not always result in the most beneficial outcome. Past research works show that including heuristics to improve the sampling strategies has in turn improved the overall performance of the algorithm. In this work, arbitrary heuristics are introduced that modify the sampling strategy based on the user requirements. This conceptualization is predicated on "learning from experience." In particular, a utility function is introduced that accounts for the samples' contribution to tree construction; which enables sampling at areas with higher utility. The idea is actualized using the Ant Colony Optimization algorithm incorporated with a well-crafted utility function that ensured an optimal tradeoff between the exploration and exploitation of the state space.**

The results of the implementation were evaluated on multiple test cases with different start and goal nodes. In addition, the effects of tuning the hyper-parameters of the Ant Colony Optimization(ACO) algorithm on the results were analyzed. The proposed optimization pipeline is seen to significantly improve the performance of the Rapidly-exploring Random Trees*(RRT*) path planning algorithm.

***Index Terms*—path-planning, sampling-based path planning, rapidly exploring random trees, ant colony optimization**

I. Introduction

In several disciplines, including robotics, autonomous vehicles, and computer graphics, where effective and optimal traversal across complex environments is required, path planning plays a key role. A basic issue in these disciplines is the capacity to identify a practical and ideal path from a starting point to a goal position while avoiding barriers. Path planning is important because it can help to allocate resources more efficiently, increase productivity, and ensure safe passage.

Algorithms for path planning that use sampling have proven to be efficient in addressing the challenges posed by high-dimensional and dynamic settings. These algorithms sample points in the environment, connect them and create a roadmap to create a graph-based representation of the state space. Rapidly exploring Random Trees (RRTs), one of these methods, has grown in popularity as a result of its propensity to handle high-dimensional spaces and quickly traverse the state space in search of viable paths.

https://github.com/tvpian/ACO-RRT-Star

While sampling-based path planning techniques have the potential to produce positive outcomes, they have difficulty maximizing path quality, particularly in terms of length and optimality. The investigation of cutting-edge methods to improve the effectiveness of current sampling-based procedures is motivated by this constraint. An effective method to increase the effectiveness and optimality of path-planning algorithms is Ant Colony Optimization (ACO), which takes its inspiration from the foraging activity of ants.

To direct the exploration of the search space, ACO introduces a pheromone-based communication system among artificial ants. ACO tries to identify the most advantageous paths based on the ants' acquired knowledge by laying and updating pheromone trails on the graph representation. In terms of path quality, convergence rates, and robustness to challenging environments, the integration of ACO with sampling-based path planning algorithms has yielded encouraging results.

This study looks into the advantages of introducing ACO methods into current sampling-based path-planning algorithms. We seek to enhance the effectiveness and optimality of pathfinding in dynamic and high-dimensional situations by utilizing the exploratory nature of ACO. The usefulness and superiority of the suggested strategy will be proven through thorough experimentation and comparison with cutting-edge sampling-based path-planning techniques.

In conclusion, sampling-based algorithms have emerged as practical methods to solve the challenges of path planning, a crucial activity with many applications. This research intends to improve the quality of pathfinding, resulting in more effective and optimal traversal through complicated settings by integrating the capability of Ant Colony Optimization into existing sampling-based methodologies. The findings of this study will help to expand path-planning methodologies, enabling better autonomy and decision-making across a variety of areas.

A. Background

Several sampling-based path-planning algorithms have been put out in recent years. These works all outperform the original

RRT* method by tweaking and improving a few of the subroutines that make up the algorithm. The referenced algorithms, on the other hand, are created expressly to address the optimal shortest path-planning problem under certain constraints. The chosen work focuses on developing a framework that will enable the addition of heuristics to the original RRT and RRT* algorithms by adjusting the sampling distribution, which is learned online as the state space is sampled. Having such heuristics defined has two benefits:

- It enables the incorporation of certain new information to effectively answer the path-planning problem.
- It could enhance the performance of any of the aforementioned works when utilized in conjunction with them. We will specifically demonstrate in this study that our approach beats the cutting-edge RRT and RRT* algorithms when paired with shortest-path heuristics.

II. Literature Review

Sampling-based path-planning solutions due to their excellent performance in high-dimensional domains are largely consumed for multiple robotic applications. Rapidly exploring random trees (RRTs) [1] and probabilistic roadmaps (PRMs) [2] are two good examples of these techniques. A variant of the RRT algorithm known as RRT* [3] is also known to reach asymptotic optimality with respect to a certain cost function.

Unlike traditional methods, the biological mechanisms of animals have been optimized through millions of years of natural evolution, and hence their behaviors can perform many intelligent tasks accurately and robustly. Therefore, it has been observed that the use of such algorithms in tandem with standard path-planning techniques could significantly improve performance in terms of efficiency [4]

Ant Colony Optimization is found to be superior to many other standards of Bio-inspired algorithms like Monte- Carlo methods and other swarm optimization techniques, especially for determining optimal sampling distributions for continuous domains [5]. Ant colony optimization has also been used, by Mohamad et al., [6] in the context of PRMs. The goal of Mohamad et al was to reduce the number of intermediate configurations from an initial to a goal position. That work serves as a model for integrating the ACO into a sampling-based path planner even if it has a different goal. We may create a scalable method suitable for real-time path-planning applications by learning the sample distribution and defining a novel utility function.

III. Methodology

A. Path planning

This conceptualization is predicated on "learning from experience." In particular, a utility function is developed that considers the contribution of the samples to the tree building; sampling then becomes more frequent in areas of higher utility. The concept is implemented by incorporating an ant colony optimization concept into the RRT/RRT* algorithm and creating a novel utility function that enables trading off state space exploration vs exploitation.

1) Rapidly-exploring Random Trees (RRT)::* It is a path-planning algorithm commonly used in robotics and motion planning. It is an extension of the original RRT algorithm, which optimizes the path found by RRT by iteratively rewiring the tree. RRT* is based on the following algorithm,

- Initialize the tree T with a single node at the starting configuration q_start.
- For $k = 1$ to K, repeat:
  a. Randomly sample a configuration q_rand from the configuration space.
  b. Find the node in the tree $T\_nearest$ to q_rand.
  c. Extend T from $T\_nearest$ to q_rand to create a new node q_new.
  d. For all nodes q_near in the neighborhood of q_new, rewire T to minimize the cost-to-come to q_new.

The cost-to-come to a node is the sum of the costs of all edges in the tree from the starting node to that node. The cost-to-go from a node to the goal is the estimated cost of the cheapest path from that node to the goal.

The RRT* algorithm uses two cost functions, the cost-to-come g and the cost-to-go h, to find the optimal path. The total cost of a path is the sum of g and h, i.e., $f = g + h$. The cost-to-go function h is typically estimated using heuristics based on the distance between the node and the goal.

The rewire step in RRT* involves checking whether rewiring a node q_near to the new node q_new would result in a lower cost-to-come. If so, the parent of q_near is updated to q_new, and the costs of all nodes in the subtree rooted at q_near are updated accordingly.

The RRT* algorithm guarantees asymptotic optimality, meaning that as the number of nodes in the tree approaches infinity, the algorithm will converge to the optimal solution with probability 1.

2) Ant Colony Optimization (ACO): Ant Colony Optimization (ACO) is a metaheuristic algorithm inspired by the foraging behavior of ants. It is commonly used for solving combinatorial optimization problems such as the Traveling Salesman Problem (TSP).

The ACO algorithm works by simulating the behavior of ants in finding the shortest path between their nest and a food source. The algorithm uses a probabilistic approach to guide the search towards the most promising solutions.

The algorithm can be summarized as follows,

- Initialize a population of ants at a starting node.
- For each ant, repeat until it reaches the goal:
  a. Choose the next node to visit based on a probability distribution.
  b. Update the pheromone trail on the edge between the current and next node.
- Update the pheromone trail globally based on the quality of the solutions found.
- Repeat steps 2-3 until a stopping criterion is met.

The probability of choosing the next node to visit is based on a combination of the pheromone trail on the edges and the distance between the nodes. The pheromone trail represents the cumulative experience of the ants in the population and is updated by each ant based on the quality of the solutions found.

The probability of choosing the next node j from the current node i by ant k is given by,

$$p_{ij}^k = \frac{\tau_{ij}(t)^\alpha \eta_{ij}^\beta}{\sum_{l \in J_i^k} \tau_{il}(t)^\alpha \eta_{il}^\beta}$$

where, $\tau_{ij}(t)$ is the amount of pheromone on the edge between nodes i and j at time t η_{ij} is a heuristic value that represents the desirability of moving from node i to node j α and β are parameters that control the relative importance of the pheromone trail and the heuristic value, respectively. J_i^k is the set of nodes that can be visited from node i by ant k. The pheromone trail is updated globally by depositing pheromones on the edges based on the quality of the solutions found. The amount of pheromone deposited on the edge between nodes i and j by the ant k is given by,

$$\Delta\tau_{ij}^k = \begin{cases} Q/L_k & \text{, if ant k uses edge (i,j) in its solution} \\ 0 & \text{, otherwise} \end{cases}$$

where, Q is a constant that represents the amount of pheromone deposited L_k is the length of the solution found by ant k The pheromone trail is also subject to evaporation over time, which prevents the algorithm from getting stuck in local optima.

ACO has been shown to be effective for a wide range of combinatorial optimization problems and has been used in various applications such as routing, scheduling, and network design.

3) ACO - RRT:* There are five stages in the algorithm. Initially, the ants that will produce subsequent samples are set up. Second, a sample is taken from the distribution that the present ants have described. Then, using the original RRT/RRT* technique, the designed tree is updated. Then, based on how much the sample could advance the existing course, its utility is determined. Using the existing solution and searching the state space for a different, better solution are the two parts of this. The ants are further updated and further re-sampled in accordance with the new distribution using that utility.

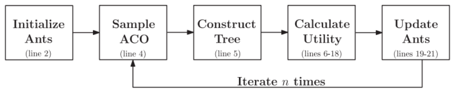

Fig. 1. Block Diagram for ACO-RRT*

Algorithm 1 ACO/RRT* Algorithm

```
0: $\nu \leftarrow x_A$; $\varepsilon \leftarrow \emptyset$; $\mathbf{T} \leftarrow \emptyset$; $\mathcal{T}_{A,B} \leftarrow \emptyset$; $c_{path} \leftarrow \infty$
0: $\mathbf{T} \leftarrow$ InitializeAnts($\mathbf{T}$, $k$)
0: for $i = 1, \dots, n$ do
0:   $x_{rand}, l \leftarrow$ SampleACO($\mathbf{T}$)
0:   $x_{new}, \mathcal{G}(\nu, \varepsilon) \leftarrow$ ConstructTree$(x_{rand}, \mathcal{G})$
0:   $U_{explore} \leftarrow U_{explore}(x_{new}, \mathcal{G})$
0:   if $c_{path} \neq \infty$ then
0:     if Cost$(x_new, \mathcal{G})$ + Cost$(\mathcal{T}(x_{new}, x_B)) < c_{path}$ then
0:       $\alpha \leftarrow \hat{\alpha}$
0:       if Cost$(x_B, \mathcal{G}) < c_{path}$ then
0:         $U_{exploit} \leftarrow U_{exploit}(x_{new}, \mathcal{G})$
0:       else
0:         $U_{exploit} \leftarrow \hat{U}_{exploit}(x_{new}, \mathcal{G})$
0:       end if
0:     else
0:       $\alpha \leftarrow \emptyset$; $U_{explore} \leftarrow \emptyset$; $U_{exploit} \leftarrow \emptyset$
0:     end if
0:   else
0:     $\alpha \leftarrow \hat{\alpha}$
0:     $U_{exploit} \leftarrow \hat{U}_{exploit}(x_{new}, \mathcal{G})$
0:   end if
0:   $\mathcal{T}_{A,B} \leftarrow$ FindBestPath$(x_A, x_B, \mathcal{G})$
0:   $\mathcal{P} \leftarrow \{U_{explore}, U_{exploit}, x_{new}, x_A, x_B, \alpha, l, \mathcal{T}_{A,B}(\mathcal{G}), c_{path}\}$
0:   $\mathbf{T} \leftarrow$ UpdateAnts$(\mathbf{T}, \mathcal{G}, \mathcal{P})$
0:   if $\mathcal{T}_{A,B} \neq \emptyset$ then
0:     $c_{path} \leftarrow$ Cost$(x_B, \mathcal{G})$
0:   end if
0: end for
0: return $\mathcal{T}_{A,B}(\mathcal{G})$ =0
```

IV. Results

Table I contains a comparison between ACO-RRT* and RRT* for three different cases based on the cost of the optimal path, the total algorithm compute time and the number of explored nodes. Figs. 2-7 show the results for the three cases for both algorithms.

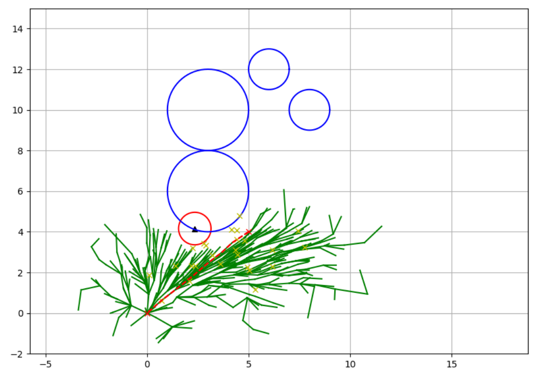

Fig. 2. Simulation result for ACO-RRT* - Case 1

TABLE I
COMPARISON OF ACO-RRT* WITH RRT* ON THE BASIS OF COST, TIME OF EXECUTION, AND NUMBER OF NODES EXPLORED.

Algorithm	**Cost**			**Time**			**No. of nodes**		
	Case 1	**Case 2**	**Case 3**	**Case 1**	**Case 2**	**Case 3**	**Case 1**	**Case 2**	**Case 3**
RRT*	7.37	13.46	10.05	184.64	162.53	186.73	506	478	484
ACO-RRT*	5.44	12.19	7.44	204.81	145.31	230.66	482	282	536

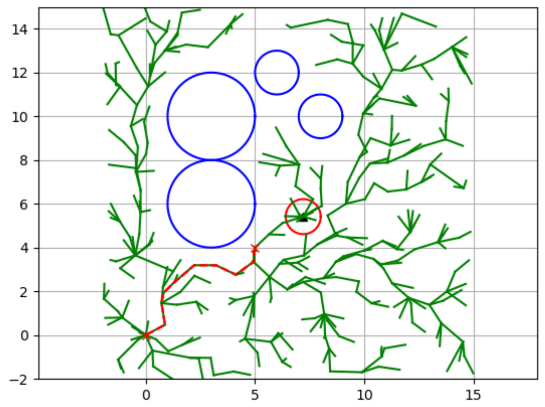
Fig. 3. Simulation result for RRT* - Case 1

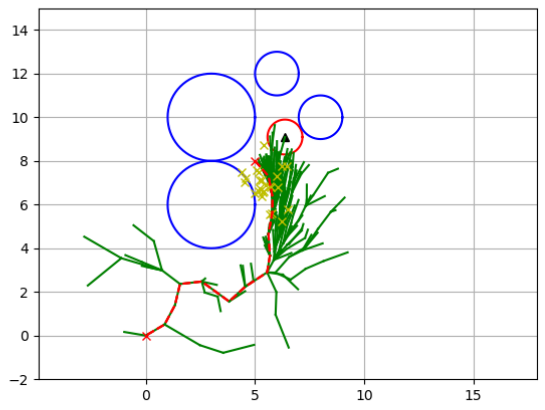
Fig. 4. Simulation result for ACO-RRT* - Case 2

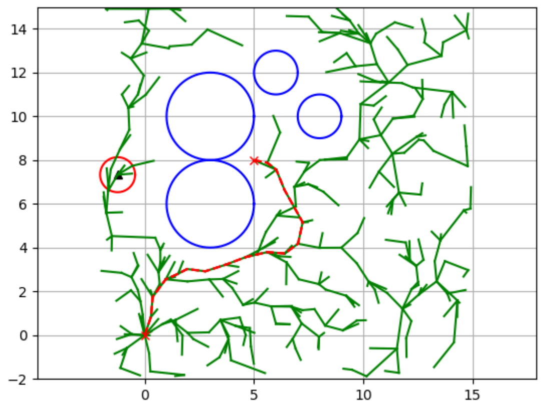
Fig. 5. Simulation result for RRT* - Case 2

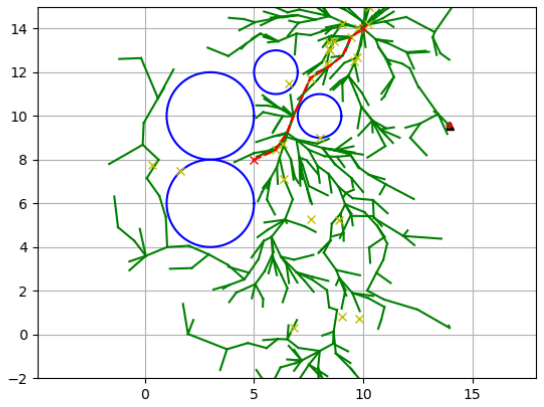
Fig. 6. Simulation result for ACO-RRT* - Case 3

TABLE II
EFFECT OF RADIUS OF CONNECTION ON THE COST, TIME OF EXECUTION, AND NO. OF NODES

Radius	**Cost**	**Time**	**No. of Nodes**
0.1	10.99	150.55	191
0.5	10.75	103.65	134
1	10.89	105.11	201

The effect of the radius of connection on the cost, time of execution, and the number of nodes has been tabulated in Table II. Fig. 8-10 shows the graphical representation of the same.

The effect of the evaporation factor on the cost, time of execution, and the number of nodes has been tabulated in Table

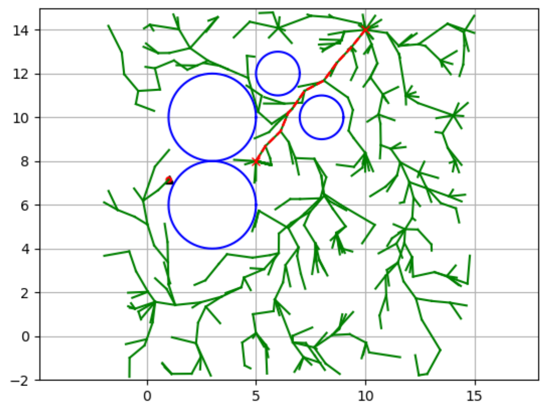
Fig. 7. Simulation result for RRT* - Case 3

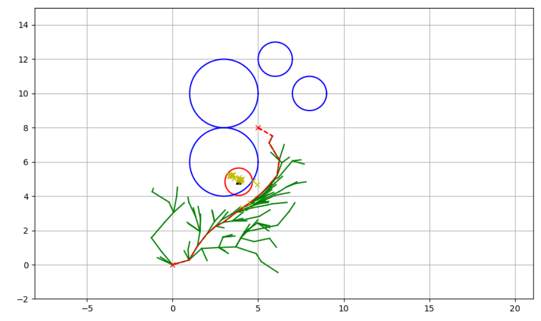
Fig. 8. Effect of radius of connection - R = 0.1

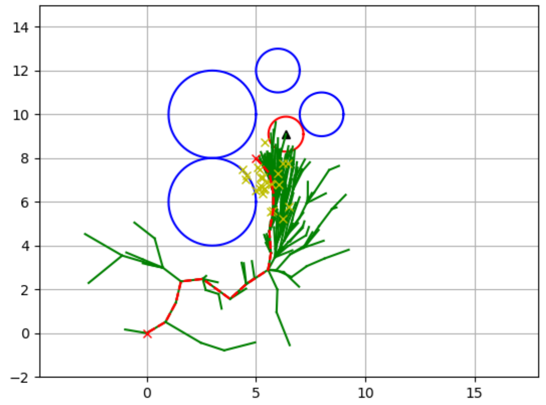
Fig. 9. Effect of radius of connection - R = 0.5

III. Fig. 11-13 show the graphical representation of the same.

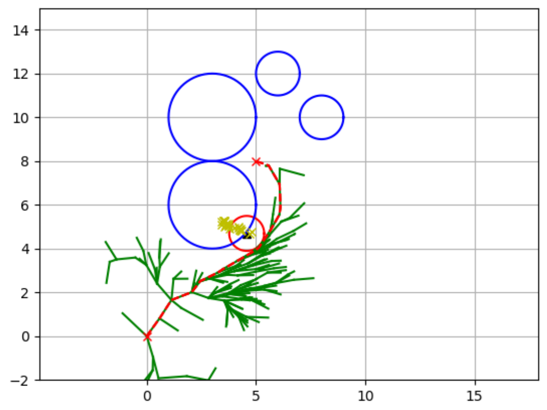
Fig. 10. Effect of radius of connection - R = 1

TABLE III
EFFECT OF EVAPORATION FACTOR ON THE COST, TIME OF EXECUTION, AND NO. OF NODES

Evaporation Factor	**Cost**	**Time (s)**	**No. of Nodes (#)**
0.4	11.26	147.68	301
0.6	12.43	12.44	129
1	10.99	150.55	191

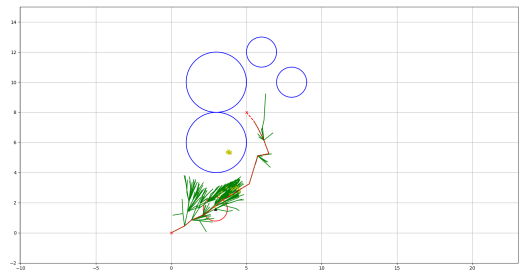
Fig. 11. Effect of evaporation factor on the exploration - η = 0.4

The effect of the number of ants on the cost, time of execution, and the number of nodes has been tabulated in Table IV. Fig. 14-16 show the graphical representation of the same.

TABLE IV
EFFECT OF NUMBER OF ANTS ON THE COST, TIME OF EXECUTION, AND NO. OF NODES

No. of Ants	**Cost**	**Time (s)**	**No. of Nodes (#)**
10	10.6	184.25	432
25	10.99	150.55	191
50	10.36	166.10	340

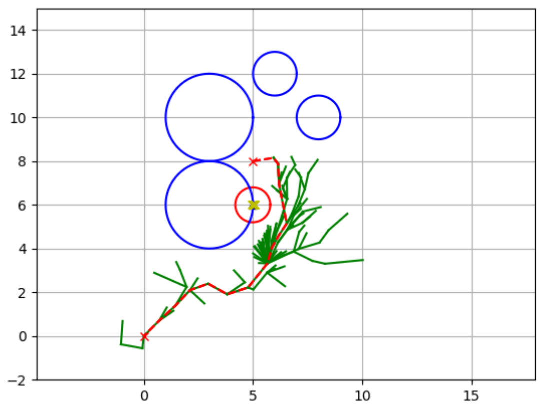
Fig. 12. Effect of evaporation factor on the exploration - η = 0.6

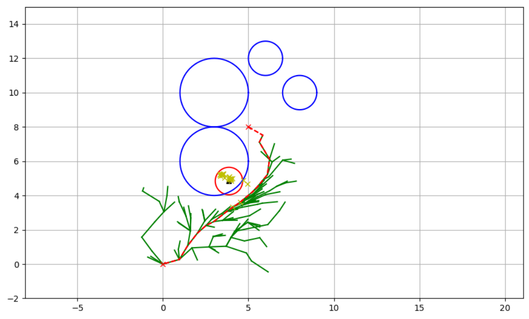
Fig. 13. Effect of evaporation factor on the exploration - η = 1

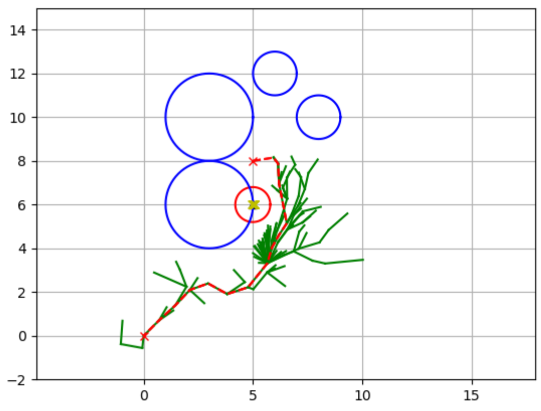
Fig. 14. Effect of number of ants on the exploration - k = 10

V. Experiments

The path generated for case 1 from Table I was used to generate the trajectory for a turtle bot simulated in the Gazebo environment as shown in Fig. 17. The outcome validated the conclusions derived from the above experiments. The video of the experiment can be found here

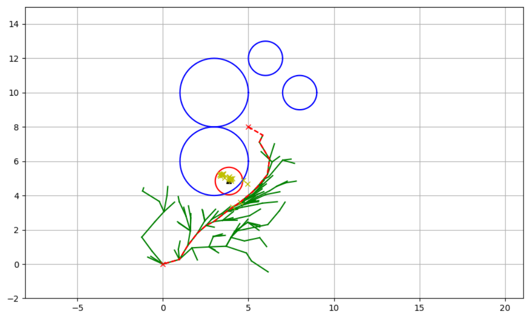
Fig. 15. Effect of number of ants on the exploration - k = 25

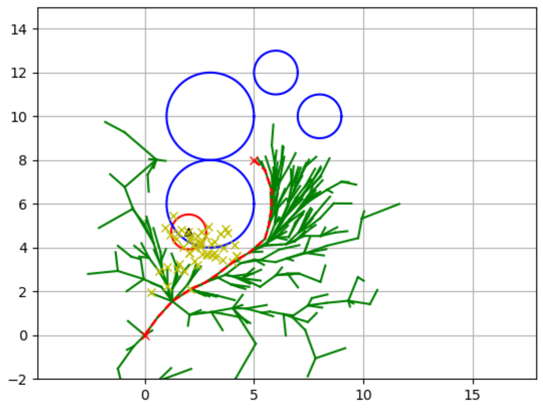
Fig. 16. Effect of number of ants on the exploration - k = 50

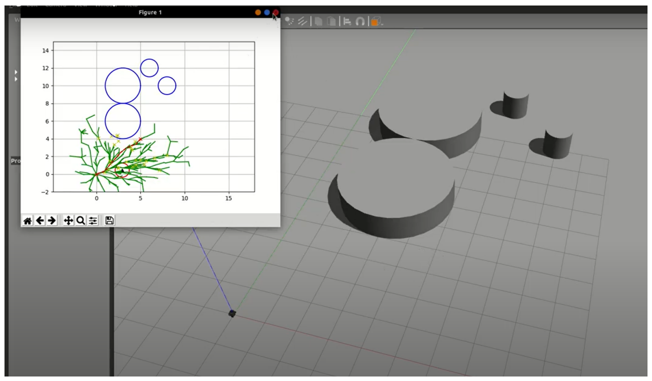
Fig. 17. Gazebo simulation of Turtle bot executing the path generated by ACO-RRT*

VI. Discussion

The farthest distance that two nodes in the state space can be connected by an edge is determined by the connection radius.

The RRT* method constructs a network, and one of its key parameters is the connection radius, which controls how connected the graph is. The method might not be able to generate workable solutions if the connection radius is set too low since the nodes might not be connected, producing incomplete or inefficient solutions. On the other side, if the connection radius is set too broad, the graph could end up being very thick and experience slow convergence and excessive computational costs.

The pace at which the pheromone trails that the ants left behind on the edges of the RRT* graph evaporate over time is controlled by the evaporation factor.

The algorithm's exploration and exploitation stages must be balanced, and the evaporation factor is a key component in that process. The pheromone trails may collect too quickly and the ants may become overly concentrated on one area of the state space if the evaporation factor is set too low, resulting in an early convergence and a less-than-ideal outcome. The pheromone trails could, on the other hand, dissipate too quickly if the evaporation factor is set too high, making it challenging for the ants to locate and follow a useful path in the state space.

The Ant Colony Optimization (ACO) based RRT* algorithm considers the number of ants as a key parameter because it directly influences the algorithm's capacity for exploration and exploitation.

The ACO-based RRT* algorithm increases the search space exploration by using more ants. This is due to the fact that each ant helps to explore the search space by producing new samples in various locations. The algorithm can quickly examine a greater section of the search field with more ants.

A search could become redundant and ineffective if there are too many ants present. In some scenarios, numerous ants may be searching the same regions of the search space, generating extra samples that could slow down the search process. Additionally, deploying an excessive number of ants can raise the algorithm's computing needs.

On the other side, using too few ants can restrict the search space's exploration and lower the likelihood of discovering the best course. In such circumstances, the algorithm might converge too soon and produce less-than-ideal results. As a result, it's critical to determine the ideal number of ants based on the problem's complexity and the available processing resources.

VII. Conclusion

A novel path-planning algorithm (ACO-RRT*) based on quickly exploring random trees (RRTs) has been presented and examined in this work. The sampling technique for the RRT algorithm has been altered so that it takes the current tree into account. In order to do this, a unique utility function is defined and combined with the ant colony optimization process. Exploring the states' space and (i) taking advantage of the current solution are trade-offs in the utility function definition. In three difficult situations, we evaluated the ACO-RRT* algorithm performance to the RRT and RRT* algorithms.

In comparison to the other options, the suggested method is able to find a first path of higher quality (1.4 times). The findings also imply that our algorithm approaches the ideal solution approx. 3:54 times faster than the RRT* algorithm. However, finding the first way takes longer.

VIII. Future Work

Identifying the optimal parameters that give the best possible result in the quickest time using learning strategies. Attempt to extend the objective function to accommodate complex behaviors like autonomous exploration, and dynamic obstacle avoidance. Improve the ACO-RRT* to an anytime path planner to reduce the time taken for finding the first path.

IX. Acknowledgment

We would like to thank Dr. Reza Mofaredi for giving us this opportunity. Also, we greatly appreciate the efforts put in by the Teaching Assistants of the course ENPM661 for reviewing our submission on time and giving us their valuable feedback.

References

[1] Steven M LaValle and James J Kuffner Jr. Randomized kinodynamic planning. *The international journal of robotics research*, 20(5):378–400, 2001. 2

[2] L.E. Kavraki, P. Svestka, J.-C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996. 2

[3] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011. 2

[4] Zhen Zhang, Rui He, and Yang Kuo. A bioinspired path planning approach for mobile robots based on improved sparrow search algorithm. *Advances in Manufacturing*, 10:1–17, 08 2021. 2

[5] Krzysztof Socha and Marco Dorigo. Ant colony optimization for continuous domains. *European journal of operational research*, 185(3):1155–1173, 2008. 2

[6] Marco Morales, Lydia Tapia, Roger Pearce, Samuel Rodriguez, and Nancy M Amato. A machine learning approach for feature-sensitive motion planning. *Algorithmic Foundations of Robotics VI*, 17:361–376, 2005. 2