MOD RRT* Implementation

Bhavana B Rao
UID: 120389092
University of Maryland
College Park, USA
bhavana3@umd.edu

Sri ram prasad Kothapalli
UID: 120165177
University of Maryland
College Park, USA
sriram21@umd.edu

***Abstract*—This study extends the multiobjective dynamic rapidly exploring random tree star (MOD-RRT*) algorithm, originally designed for autonomous robot navigation in dynamically changing environments. MOD-RRT* integrates a modified RRT* for initial path generation with a subsequent shortcutting method to optimize this path. Additionally, it incorporates a dynamic replanning feature when paths become infeasible, selecting optimal nodes swiftly by considering path length and smoothness. Our work not only implements this algorithm but also conducts a comparative analysis against traditional RRT* and dynamic RRT* methods. Through both simulated environments, which demonstrate the algorithm's capability to circumvent unforeseen obstacles effectively, and real-world applications, we assess the performance enhancements offered by MOD-RRT*. Results indicate that MOD-RRT* significantly improves the quality of initial path generation and overall navigation efficiency in dynamic settings, showcasing its practical advantages over existing static and dynamic path planning techniques.**

***Index Terms*—Multi-Objective, RRT*, Dynamic Obstacles, Path Planning**

I. Introduction

In recent years, the field of autonomous mobile robotics has garnered significant attention due to its vast applications in areas such as emergency rescue, autonomous exploration, and warehouse management. Path planning, a core area of robotics research, involves the computation of a collision-free trajectory from a robot's starting point to its destination. This not only necessitates determining a safe path but also optimizing the trajectory for energy and time efficiency. Techniques such as path smoothing with spline curves and considering terrain roughness to reduce energy consumption have been explored to enhance trajectory optimization.

Among the prevalent algorithms in this domain, sampling-based strategies like the Rapidly Exploring Random Tree (RRT) and Probabilistic Roadmap have been effective for tackling complex planning challenges, especially with nonholonomic systems. These methods have primarily focused on speed rather than optimality. However, the standard RRT does not guarantee asymptotic optimality, which led to the development of the Optimal RRT (RRT*), enhancing the algorithm by introducing a rewire step that optimizes the path by adjusting neighboring vertices of newly added nodes.

As environments are often only partially known, real-time path replanning becomes crucial to avoid newly detected obstacles, ensuring safety and continuity of the navigation process. Incremental search algorithms, leveraging the tree structure of the RRT for dynamic replanning, have shown promise over traditional methods. Despite these advancements, the focus has mostly been on optimizing path length, whereas path smoothness is also a critical factor that significantly impacts energy consumption.

To address these challenges, this report introduces a multiobjective dynamic path planning algorithm, termed Multiobjective RRT* (MOD-RRT*), which is designed to efficiently handle the complexities of dynamic environments. The MOD-RRT* consists of two main components: an initial path generation using a modified RRT* to build a state tree based on known maps and a replanning scheme to adjust paths in response to detected obstacles. This algorithm employs a backward expansion strategy to store cost values from tree nodes to the goal, a path optimization method to minimize the effects of random sampling, and a reconnection scheme to maintain path integrity in the face of unexpected obstacles. Additionally, it utilizes multiobjective path replanning based on Pareto efficiency to select the

optimal path during navigation. This comprehensive approach ensures the algorithm maintains both probabilistic completeness and asymptotic near-optimality in dynamic path planning scenarios.

A. Pareto Dominance Theory

Pareto dominance is a fundamental principle used to evaluate efficiency in multi-objective optimization problems. It provides a way to compare different solutions or decision outcomes where multiple conflicting objectives are involved. The concept is named after Vilfredo Pareto, an Italian economist, who introduced it in the context of economic efficiency and income distribution. A solution 'A' is said to Pareto dominate another solution 'B' if 'A' is atleast as good as 'B' in all objectives and better in at least one objective. Mathematically, there are two solutions A and B with objective functions f_1,f_2,....f_n, A Pareto dominates B if:

$$\forall_i \in (1, 2, 3, ..., n), f_i(A) \leq f_i(B) \tag{1}$$

$$\exists j \in (1, 2, 3, ..., n), f_i(A) < f_i(B) \tag{2}$$

In multiobjective optimization, the goal is often to find the Pareto Front rather than a single optimal solution. This approach is particularly important in fields like engineering, economics, and operations research where trade-offs between conflicting objectives must be carefully managed. For instance, in robotic path planning, objectives such as minimum path length, maximum safety, and minimum energy consumption may conflict, and optimizing one may lead to suboptimal results in others.

The use of Pareto Dominance in optimization algorithms enables the generation of a set of possible solutions, allowing decision-makers to understand the trade-offs and make informed choices based on their specific preferences or constraints.

II. Methodology

This section outlines the core algorithms employed in the study: Rapidly Exploring Random Tree Star (RRT*), Dynamic RRT* (DRRT*), and Multiobjective Dynamic RRT* (MOD-RRT*). Each of these algorithms was run in 2 different maps with more obstacles in the second map to report a detailed analysis of 3 algorithms and come to a conclusion on our main objective of showing the effectiveness of MOD RRT* in a dynamic environment setting.

A. RRT* (Rapidly Exploring Random Tree)

RRT* is an advanced version of the Rapidly Exploring Random Tree (RRT), a path planning algorithm known for its efficiency and simplicity in navigating high-dimensional spaces. While RRT quickly finds a feasible path, it does not guarantee optimality. RRT* addresses this by incorporating a reconnection strategy that continually refines the path to optimality.

In RRT*, each new node added to the tree is checked for potential connections not just to its nearest neighbor but also to other close nodes within a defined radius. This radius shrinks as the tree grows, balancing between exploration and optimization. The algorithm evaluates whether connecting through the new node reduces the path cost of its neighbors. If so, the tree is restructured to adopt these more efficient paths, progressively improving the solution towards an optimal one.

B. Dynamic RRT*

Dynamic RRT* extends the RRT* to adaptively respond to changing environments—a necessity in real-world applications where obstacles might move or appear unexpectedly. DRRT* integrates feedback mechanisms to detect changes and modifies the path in real-time. This dynamic adjustment is typically achieved by periodically re-evaluating and updating the path as new environmental information becomes available.

The DRRT* uses a similar approach to RRT* for path optimization but with added mechanisms for path invalidation and regeneration. When a previously valid path becomes obstructed, the algorithm identifies the affected nodes and efficiently searches for alternative paths, minimizing the need to rebuild the entire tree from scratch.

C. MOD-RRT* (Multiobjective Dynamic RRT*)

MOD-RRT*, the focal point of this study, builds upon the strengths of DRRT* by incorporating multiple objectives into the path planning process. Unlike its predecessors, which primarily focus on path length or time efficiency, MOD-RRT* simultaneously optimizes for several criteria, such as path smoothness, energy efficiency, and safety.

MOD-RRT* operates in two phases: initial path generation and dynamic replanning. The initial path is generated using a modified RRT* that not only seeks to minimize distance but also considers other objectives by leveraging a multi-criteria cost function. Once the initial path is established, the algorithm continuously monitors the environment for changes that might render the current path infeasible. If such a situation arises, MOD-RRT* employs a multiobjective replanning strategy. This strategy utilizes a Pareto efficiency framework to quickly evaluate and choose among multiple alternative paths, ensuring that the replanned path adheres to the predefined objectives without substantial compromises.

D. Pseudocode

Algorithm 1 RRT*

```
tree ← create_tree(start)
for iterations ← 1 to max_iterations do
    rand_point ← sample_random_point()
    nearest_node ← find_nearest_node(tree, rand_point)
    new_node ← steer(nearest_node, rand_point)
    if not collides(nearest_node, new_node, obstacles) then
        near_nodes ← find_near_nodes(tree, new_node)
        parent ← choose_parent(nearest_node, new_node, near_nodes)
        new_node.cost ← parent.cost + distance(parent, new_node)
        tree.add_node(new_node, parent)
        tree.rewire_near_nodes(new_node, near_nodes)
return find_path(tree, start, goal)
```

- **create_tree(start)**: Initializes a tree structure with a single node at the starting position.
- **sample_random_point()**: Generates a random point within the configuration space to explore new areas.
- **find_nearest_node(tree, point)**: Identifies the closest existing node in the tree to a given point for connection.
- **steer(nearest_node, rand_point)**: Determines a new node by moving from the nearest node towards a random point to explore the space.
- **not collides(nearest_node, new_node, obstacles)**: Checks if the path between the nearest node and the new node intersects any obstacles.
- **find_near_nodes(tree, new_node)**: Locates nodes in the tree that are within a specified radius of the new node for potential connection.
- **choose_parent(nearest_node, new_node, near_nodes)**: Selects the best parent among the nearby nodes for the new node based on cost.
- **new_node.cost = parent.cost + distance(parent, new_node)**: Computes the cost of the new node as the sum of its parent's cost and the distance between them.
- **tree.rewire_near_nodes(new_node, near_nodes)**: Adjusts the tree connections to optimize the path if a better path is found through the new node.
- **find_path(tree, start, goal)**: Traces back from the goal node to the start node along the tree to determine the optimal path.

Algorithm 2 Dynamic RRT*

```
1.2 while True do
    rand_point ← GenerateRandomPoint()
    nearest_node ← FindNearestNode(nodes, rand_point)
    angle ← CalculateAngle(nearest_node.point, rand_point)
    new_point ← CalculateNewPoint(nearest_node.point, angle)
    if ¬in_obstacle(new_point) and ¬intersects_dynamic_obstacle(nearest_node.point, new_point)
    then
        new_node ← CreateNode(new_point, nearest_node)
        new_node.cost ← nearest_node.cost + distance(nearest_node.point, new_point)
        nodes.append(new_node)
        return nodes, new_node
```

- **GenerateRandomPoint()**: Generates a random point within the configuration space.
- **FindNearestNode(nodes, rand_point)**: Identifies the nearest node in the tree to the randomly generated point.
- **CalculateAngle(nearest_node.point, rand_point)**: Calculates the angle between the nearest node's position and the random point.
- **CalculateNewPoint(nearest_node, angle)**: Determines a new point that extends from the nearest node towards the random point.

- **in_obstacle(new_point)**: Checks if the new point is within an obstacle.
- **CreateNode(new_point, nearest_node)**: Creates a new node with the calculated new point and sets the nearest node as its parent.
- **new_node.cost = nearest_node.cost + distance(nearest_node.point, new_point)**: Sets the cost of the new node.
- **nodes.append(new_node)**: Adds the new node to the tree.

Algorithm 3 MOD-RRT* Algorithm

while *NotReachGoal* **do**
  if *FindObstacle* **then**
    $x_{\text{cur}} \leftarrow$ CurrentPosition $X_{\text{candi}} \leftarrow \emptyset$
    $X_{\text{near}} \leftarrow \text{NeighborNodes}(x_{\text{cur}}, T.V, d)$ **for** x_{near} ***in*** X_{near} **do**
      if *Collision_Free*(x_{cur}, x_{near}) **then**
        $X_{\text{candi}} \leftarrow X_{\text{candi}} \cup x_{\text{near}}$
    $x_{\text{next}} \leftarrow \text{Pareto}(X_{\text{candi}})$
  $x_{\text{cur}} \leftarrow x_{\text{next}}$

- **NotReachGoal**: Continue the algorithm until the goal is reached.
- **FindObstacle**: Check for the presence of obstacles.
- **NeighborNodes(x_{cur}, $T.V$, d)**: Find neighboring nodes of x_{cur} within distance d.
- **Collision_Free(x_{cur}, x_{near})**: Check if there is no collision between x_{cur} and x_{near}.
- **Pareto(X_{candi})**: Select the best candidate node based on Pareto dominance from X_{candi}.
- x_{cur}: The current node being considered.
- x_{next}: The next node to be considered based on the algorithm's logic.

III. Results

Figure 1 shows a pygame screen of (800,600) pixels. We have 3 static obstacles shown in black color. And the RRT* Algorithm finds a path between the start and the end node.

Figure 2 is a representation of the RRT* Algorithm in a different map.

Figure 3 represents the dynamic RRT* Algorithm. The obstacles shown in purple are the dynamic obstacles. The spatial position and shape of the dynamic obstacle is unknown to the algorithm. It finds a path avoiding both static and dynamic obstacles. The red line represents the final path. The orange line shows the smoothened and more feasible path.

Figure 4 represents the dynamic RRT* Algorithm in a different map.

Figure 5 shows the MOD-RRT* Algorithm. Pareto dominance theory ensures that each new node added to the tree is not dominated by any existing node in terms of cost and distance to the goal. This selection method helps maintain a diverse set of non-dominated solutions, improving the algorithm's ability to explore the search space efficiently.

Figure 6 represents the MOD-RRT* Algorithm in a different map.

Comparing the algorithms based on the table 1, we can draw several insights:

Execution Time: MOD-RRT* generally has the shortest execution times compared to RRT* and Dynamic-RRT*. This could be due to MOD-RRT*'s more efficient node selection process, which focuses on maintaining a diverse set of non-dominated solutions. Number of Nodes Explored: MOD-RRT* explores significantly fewer nodes compared to RRT* and Dynamic-RRT*. This suggests that MOD-RRT* is more efficient in its exploration strategy, potentially due to the Pareto dominance theory guiding its node selection. Number of Nodes in Path: MOD-RRT* and RRT* tend to have a similar number of nodes in their paths, while Dynamic-RRT* generally requires more nodes. This could be due to the additional complexity of avoiding dynamic obstacles, which leads to a more convoluted path. Overall Efficiency: MOD-RRT* appears to strike a good balance between exploration efficiency (low number of nodes explored) and path optimality (similar number of nodes in path compared to RRT*). It achieves this by focusing on non-dominated solutions, ensuring a diverse and potentially more optimal set of paths.

The time and space complexity of the MOD-RRT* algorithm can be described as follows:

Time Complexity: **O(n log n)**, where n is the number of iterations.

Space Complexity: **O(n)**, where n is the total number of nodes in the tree.

Algorithm	Execution time	Number of nodes explored	Number of nodes in path
RRT* (Map1)	17.36s	518	27
RRT* (Map1)	17.44s	520	27
Dynamic-RRT* (Map1)	27.65s	547	56
Dynamic-RRT* (Map2)	22.13s	360	60
MOD-RRT* (Map1)	11.36s	90	51
MOD-RRT* (Map2)	16.32s	82	58

TABLE I
TIME SPACE COMPLEXITY

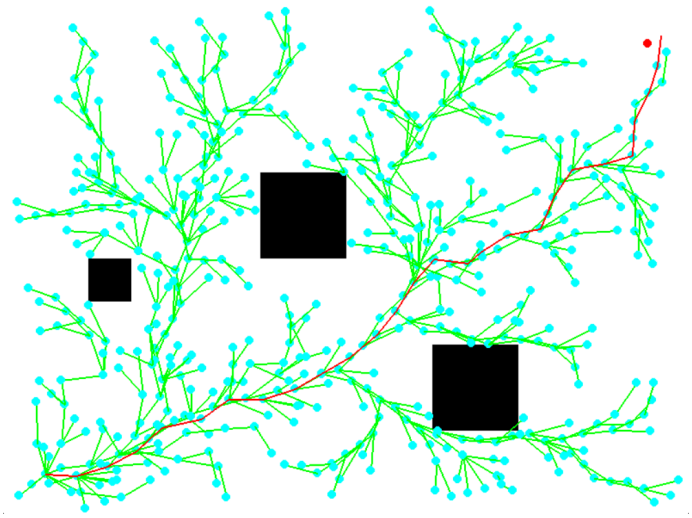

Fig. 1. RRT* for Map1

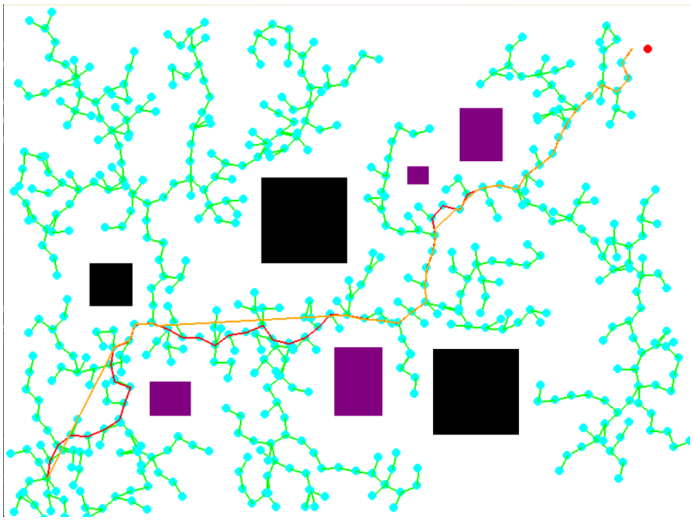

Fig. 3. Dynamic RRT* for Map1

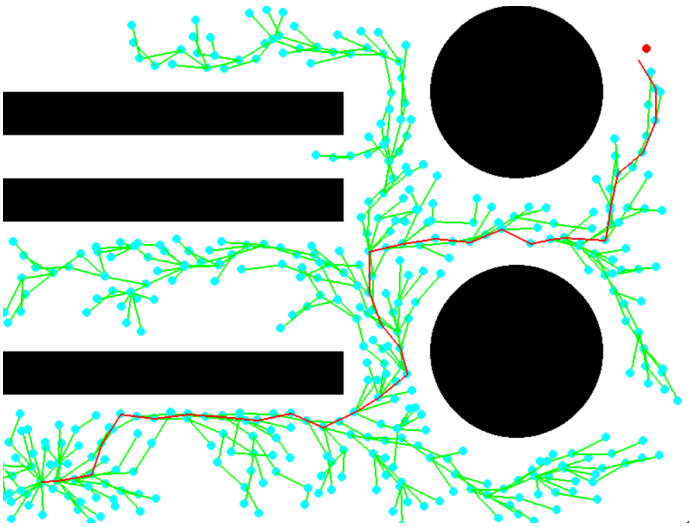

Fig. 2. RRT* for Map2

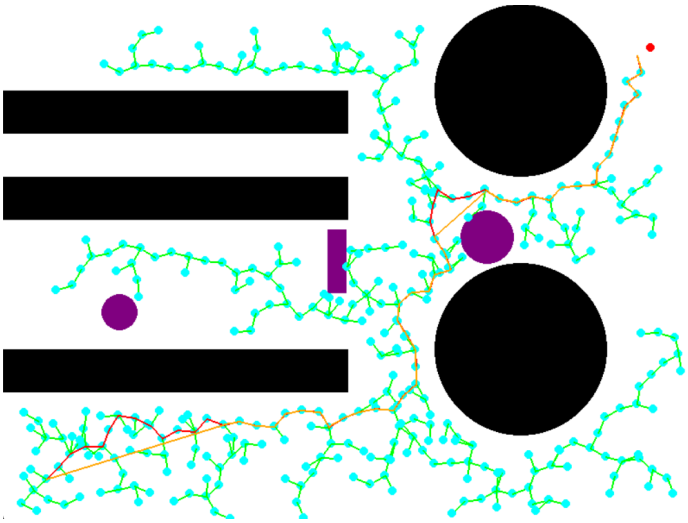

Fig. 4. Dynamic RRT* for Map2

IV. CONCLUSION

After comparing the RRT*, Dynamic RRT*, and MOD-RRT* algorithms, several key observations can be made. RRT* demonstrates a robust and efficient path planning approach, effectively exploring the search space and finding near-optimal paths. However, in environments with dynamic obstacles, its performance may degrade due to frequent collisions and recalculations.

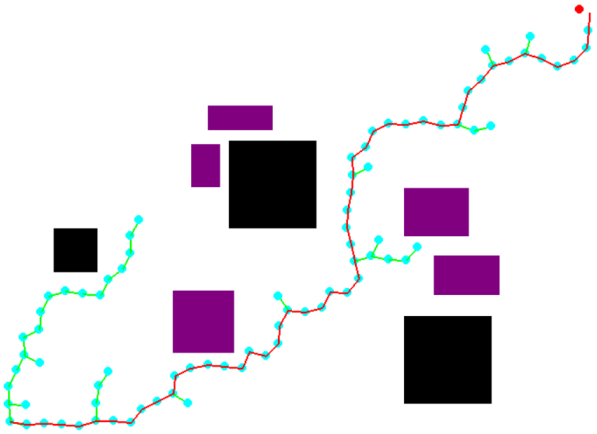

Fig. 5. MOD-RRT* for Map1

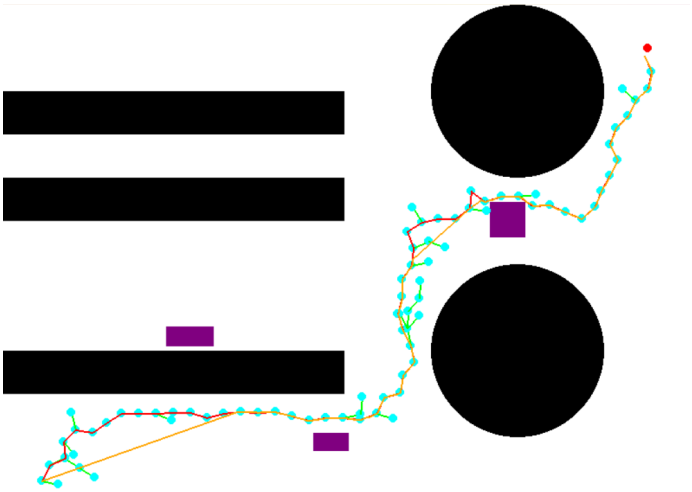

Fig. 6. MOD-RRT* for Map2

Dynamic RRT* addresses this limitation by considering dynamic obstacles, leading to better adaptability and path quality in such environments. By updating the tree structure to avoid dynamic obstacles, it achieves improved path planning results compared to RRT*.

MOD-RRT* further enhances the capabilities by incorporating Pareto dominance theory, ensuring that each new node added to the tree is non-dominated in terms of cost and distance to the goal. This results in a diverse set of non-dominated solutions, enabling more efficient exploration of the search space and better path quality.

In conclusion, while RRT* provides a strong baseline for path planning, Dynamic RRT* and MOD-RRT* offer advancements that cater to specific challenges, such as dynamic obstacles and multi-objective optimization. The choice of algorithm depends on the specific requirements and constraints of the robotics application, with MOD-RRT* standing out for its ability to handle dynamic environments and diverse solution sets.

REFERENCES

[1] O. Salzman and D. Halperin, "Asymptotically near-optimal RRT for fast, high-quality motion planning," IEEE Trans. Robot., vol. 32, no. 3, pp. 473– 483, Jun. 2016.

[2] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," Int. J. Robot. Res., vol. 30, no. 7, pp. 846–894, 2011.

[3] C. Tsai, H. Huang, and C. Chan, "Parallel elite genetic algorithm and its application to global path planning for autonomous robot navigation," IEEE Trans. Ind. Electron., vol. 58, no. 10, pp. 4813–4821, Oct. 2011.

[4] F. Kamil and K. N, "A review on motion planning and obstacle avoidance approaches in dynamic environments," Adv. Robot. Autom., vol. 4, pp. 134– 142, Jan. 2015.

[5] T. Oral and F. Polat, "MOD* Lite: An incremental path planning algorithm taking care of multiple objectives," IEEE Trans. Cybern., vol. 46, no. 1, pp. 245–257, Jan. 2016.

[6] F. O. Coelho, J. P. Carvalho, M. F. Pinto, and A. L. Marcato, "Direct- DRRT*: A RRT improvement proposal," in Proc. 13th APCA Int. Conf. Autom. Control Soft Comput., Jun. 2018, pp. 154–158.

[7] H.-I. Lin and C.-S. Yang, "2D-span resampling of bi-RRT in dynamic path planning," Int. J. Autom. Smart Technol., vol. 5, no. 1, pp. 39–48, 2015.

[8] Y. Zhang, D. W. Gong, and J. H. Zhang, "Robot path planning in un- certain environment using multi-objective particle swarm optimization," Neurocomputing, vol. 103, pp. 172–185, 2013.

[9] O. Adiyatov and H. A. Varol, "A novel RRT*-based algorithm for motion planning in Dynamic environments," 2017 IEEE International Conference on Mechatronics and Automation (ICMA), Takamatsu, Japan, 2017, pp. 1416-1421, doi: 10.1109/ICMA.2017.8016024.

[10] J. Nasir, F. Islam, U. Malik, Y. Ayaz, O. Hasan, M. Khan, and M. S. Muhammad, "RRT*-SMART: A rapid convergence implementation.