# Shared Search Tree-Based Multi-Objective Dynamic Path Planning: An Enhanced D* Lite Approach

Minyu Wu
College of Engineering, Nanjing Agricultural University, Nanjing, 211512, China
20050627wmy@gmail.com

## ABSTRACT

In order to solve the problem of computational redundancy in multi-objective dynamic path planning with D Lite algorithm, this paper proposes an improved version of D Lite algorithm based on Shared Search Tree, called SST-D Lite. In terms of methods, this new algorithm builds a search tree with a structure of "shared trunk+independent branches", which enables multiple targets to reuse the extended information of common nodes and reduce repeated calculation from the architectural level. We also set up a static target point lookup table based on hash mapping to reduce the complexity of target authentication to O (1). In addition, a lazy deletion mechanism is designed to maintain the node state in the priority queue and reduce the time complexity from O(N) to O (1). The algorithm also adds a weighted cost mechanism to balance the path length and security risks, and uses Breadth-First Search (BFS) to make a local reset strategy, which can accurately repair the affected areas when encountering dynamic obstacles. The simulation results show that in a complex 200×200 dynamic environment, although SST-D Lite needs a longer initialization time (1256.11 milliseconds), it shows obvious advantages in core dynamic performance indicators. Specifically, the average re-planning time of SST-D Lite is only 11.45 milliseconds, which is about 26% faster than the original D Lite and 67% shorter than the A algorithm. In terms of search efficiency, the total number of nodes expanded by SST-D Lite (3.40× 10) is about 70% of that of D Lite and only 14.7% of that of A. Experiments show that SST-D Lite can effectively filter out redundant search space while maintaining high real-time response capability and engineering practical value, and is suitable for multi-objective planning in dynamic environment.

**Keywords:** Dynamic Path Planning; D* Lite; Shared Search Tree; Multi-Objective Optimization; Incremental Search.

## 1. INTRODUCTION

With the rapid development of mobile robot, autonomous driving and intelligent storage system, path planning in complex dynamic environment has become the core technology of autonomous navigation. Traditional static path planning algorithms, such as A and Dijkstra, often need to recalculate the whole path when they encounter sudden environmental changes or dynamic obstacles. This calculation is very large, and it is difficult to meet the real-time requirements.

As an efficient incremental heuristic search algorithm, D Lite can update the path locally by using the information previously searched, which greatly improves the efficiency of replanning in dynamic environment. However, most of the existing D Lite algorithms and their variants can only deal with single agent and single target. In practical applications, such as warehouse logistics and search-and-rescue tasks, the system often needs to deal with multi-objective inspection or multi-agent collaborative planning, and at the same time optimize many objectives such as path length, safety and smoothness.

If you just run D Lite separately for each target, there will be a lot of repeated calculations and memory waste. Shared search tree technology can make the search process of different targets share the node extension information, which is a good way to solve this problem. Therefore, in this study, we put the shared search tree mechanism and a multi-objective optimization strategy into the framework of D Lite, in order to solve the problem that the calculation of multi-objective path planning is too repetitive and the real-time performance is not good in dynamic environment. This research has important theoretical significance and practical engineering value.

## 2. RELATED WORK

### 2.1 Classical Graph Search Algorithms for Path Planning

Early path planning methods mainly relied on Dijkstra's algorithm and the A* algorithm [1,2]. The A* algorithm proposed by Hart et al. accelerates the search process by introducing a heuristic function h(n) and is widely regarded as an optimal method in static environments. However, A* has two critical limitations in complex tasks. First, in dynamic environments, once an unknown obstacle blocks the original path, A* must discard the existing search tree entirely and perform global replanning from scratch. This not only causes considerable computational delay, but also prevents the reuse of historical topological features and cost values from previous planning steps. Second, A* is essentially a single-target planning algorithm. When dealing with tasks involving one start point and multiple goal points, it must be executed independently N times [3]. This accumulation for the target will lead to a linear or even exponential increase in computational complexity, resulting in a large number of repeated node expansion. In large-scale, high-resolution grid maps, it is easy to cause serious "curse of dimensionality", which makes the memory usage soar sharply and even leads to stack overflow, thus wasting a lot of computing resources [4].

### 2.2 Incremental and Dynamic Path Planning

In order to make the robot re-plan the route faster when the environment changes, Stentz proposed the D (Dynamic A) algorithm, and later Koenig and Likhachev developed the D Lite algorithm based on LPA (Lifelong Planning A) [5-7]. D Lite combines reverse search with incremental update. By keeping the rhs value and G value of each node consistent, it can quickly repair the damaged path in the unknown dynamic environment [8-9]. In recent years, researchers have also made various upgraded versions of D Lite, such as Field D by interpolation and 3D D Lite [10-11] which can be used in three-dimensional space. Although the incremental search performs well in the dynamic single-target scenario, there will be a serious performance bottleneck in the multi-target scenario. To find n targets in the same dynamic environment, the traditional method is to start N independent D Lite planners, each responsible for one target. This "independent multi-instance" mechanism will lead to the serious overlap of search fronts in the same free space, and the same grid node may be repeatedly calculated and stored by multiple priority queues. This not only greatly increases the space complexity (represented by the sharp increase of the maximum open list size), but also causes serious computational redundancy, which makes it difficult for multi-objective dynamic programming to meet the real-time requirements.

### 2.3 Multi-Objective and Multi-Agent Path Planning

In the field of multi-objective and multi-agent planning, the MO-A algorithm proposed by Stewart et al. mainly helps us find Pareto optimal solution set in multiple cost dimensions (such as time and energy consumption), rather than solving the multi-destination problem in space. CBS (Conflict-Based Search), proposed by Standley et al., is a mainstream approach for multi-agent path planning. However, when dynamic obstacles appear, its conflict-tree-based underlying logic can easily cause exponential growth in conflict tree size at narrow passages, or "chokepoints," resulting in extremely high maintenance overhead during replanning [12-14].

To reduce the spatial complexity of multi-Objective pathfinding, introducing the concept of a shared search tree or an approximation inspired by the Steiner tree has become a feasible direction. The Multi-Heuristic A* method proposed by Hernandez et al. uses multiple heuristics to share the same search space, thereby escaping local minima [15]. However, most existing shared-tree or Steiner-tree-based algorithms are limited to static environments. Research on integrating a shared search tree structure with the incremental updating mechanism of D* Lite is still very scarce [16-18]. The main difficulties are as follows: first, reconstruction of a traditional Steiner tree is an NP-hard problem, and if the shared topology must be recomputed whenever the environment changes, the resulting cost may far exceed that of independent replanning. Second, in dynamic re-planning, if the interrupted shared branches are not cut off in time, they may become "zombie branches" which occupy a lot of memory but are actually useless, which will greatly slow down the subsequent incremental update speed.

## 3. METHODOLOGY

In this study, an improved dynamic path planning algorithm called Shared Search Tree (SST-D Lite) is proposed based on the classic D Lite algorithm. Through multi-objective shared search, weighted heuristic cost evaluation and optimized local re-planning mechanism, this method greatly improves the computational efficiency of multi-objective tasks in dynamic unknown environment. The core design of the algorithm is as follows.

### 3.1 Construction of the Multi-Objective Shared Search Tree (SST)

In the multi-objective scene, in order to avoid the problem of repeated space exploration caused by the traditional algorithm, our proposed method uses the backpropagation mechanism and builds a search tree with the structure of "shared trunk+independent branches". Each node in the map has a node_goals status attribute, and its associated target is recorded with a binary bit mask. In the process of searching and updating, the common accessible areas are automatically identified and summarized through the propagation of parent-child relationship of nodes, thus forming a shared trunk. In order to reduce the computational cost of checking the node state under multi-objective, the algorithm introduces a static target point lookup table based on the idea of hash mapping, which reduces the complexity of target verification when the node is expanded to O (1) and replaces the traditional cyclic comparison method.

### 3.2 Weighted Cost Evaluation Incorporating Distance and Safety

In order to ensure the smooth and safe route, the algorithm we designed has developed a comprehensive scoring method. Every time a node is extended, its heuristic cost will consider two points at the same time: one is the linear distance to the target, and the other is the repulsive force caused by the expansion of obstacles. According to the position of static obstacles on the map, we will first calculate the straight-line distance d from each idle area node to the nearest obstacle. The safety cost function is defined as follows:

$$\mathrm{csafe}=\frac{1}{\mathrm{d}+0.5} \tag{1}$$

The closer a node is to an obstacle, the more sharply its penalty cost increases in a nonlinear manner. The cost inside an obstacle is set to 0 because such nodes are unreachable. The algorithm further defines a distance weight $\boldsymbol{W}_{\mathrm{dist}}$ and a safety weight.

$\boldsymbol{W}_{\mathrm{safe}}$. During path extraction, a guidance weight $\boldsymbol{W}_{\mathrm{g}}$ based on the shared-tree cost values is also introduced, so as to balance absolute shortest-path reachability and safety redundancy in dynamic environments.

### 3.3 Priority Queue Lazy Deletion Mechanism

In the D Lite algorithm, when the environment is updated, we often need to delete some useless nodes from the priority queue with double keywords. The previous methods, such as checking the whole queue or rebuilding a queue, are very slow and take O(N) so much time. In order to solve this speed problem, we came up with a new method called lazy deletion. We have prepared a version number book (U_ver) for each node in the map. When a node is updated and put back into the queue, its global version number will increase. And the old record in the queue, because the version number doesn't match, becomes outdated.

When we take a record from the front of the queue, the algorithm will first check whether its version number is the same as the latest version number in the record book. If it's not the same, throw this record away immediately. This method reduces the time to maintain the node state in the priority queue from O(N) to O (1), so that the calculation speed is much faster at the beginning of re-planning the path.

### 3.4 Dynamic Impact Scope Control and BFS-Based Local Reset

When dynamic obstacles appear and destroy the existing path structure, the algorithm will adopt different incremental repair strategies according to the attributes of the affected node (that is, whether it belongs to an independent branch or a shared trunk) to avoid unnecessary global reconstruction. If the damaged node is associated with only one target (indicating that it does not belong to the shared trunk), the algorithm will follow the local consistency update principle of D Lite, and only propagate the incremental cost to the affected node and its nearby area. However, if the damaged node belongs to a shared trunk shared by two or more targets (goals_using ≥2), the traditional method will clear all the shared states and rebuild the whole map, which is very resource-consuming. In order to solve this problem, the new method designs a local reset strategy based on Breadth-First Search (BFS): starting from the damaged point, the algorithm traces back to the starting point along the parent pointer, and only resets the subtree area that depends on the damaged node, while trying to preserve the unaffected shared search tree structure.

## 4. EXPERIMENTAL SETUP AND RESULTS

To verify the effectiveness of the SST-D* Lite algorithm, this study established a multi-objective dynamic path planning simulation environment on the MATLAB platform and designed benchmark tests and comparative experiments.

### 4.1 Simulation Environment and Parameter Configuration

A two-dimensional discrete grid map of size 200 × 200 was first constructed. Static obstacles were generated using a random distribution model, with an obstacle coverage rate of 20%. The fixed start point was set at (10, 10). To prevent the start point from being completely blocked by obstacles, a 20 × 20 obstacle-free protection zone was created around it. Three heterogeneous target points were placed at the far side and corners of the map, for example at (186, 186), (186, 33), and (100, 186). Each target point was assigned a protection zone with a radius of 8 grid cells to ensure its initial reachability. The distance weight was set to $W_{dist}$=1.0, the safety weight to $W_{safe}$=0.5, and the shared-path guidance weight to $W_{g}$=0.4.

### 4.2 Dynamic Obstacle Event Design

In order to make the unknown dynamic interference in the experiment appear more real, we set several dynamic triggering steps in advance. When the planner runs to a certain stage, such as step 4, step 8, step 12, step 17 and step 22, the environment will suddenly trigger an obstacle event. Each event will randomly generate 2 to 5 new dynamic obstacles. A conflict detection mechanism is also added in the generation process to ensure that these dynamic obstacles will not directly cover the target point or the current actual position of the car.

### 4.3 Comparative Algorithms and Evaluation Metrics

In order to fairly compare who is better in different algorithms, we set up three groups to run separately under the same random starting point and unexpected events:

Group A (A): The traditional heuristic search algorithm is used. It plans the route from scratch every time in a dynamic environment, and will not use the information calculated before.

Group B (Original D Lite): this is a classic incremental re-planning algorithm. It runs a separate instance for each target point, and these instances do not communicate with each other.

Group C (SST-D Lite): This is a new algorithm proposed in this paper, which is based on a shared search tree.

We selected the following three core indicators to rate them:

Initialization Time: the computing time required to generate all the target paths for the first time, in milliseconds (ms), which shows the efficiency of the algorithm in building trees in a complex environment.

Average Replanning Time: the average time taken to repair the damaged path after the dynamic change of the environment, in milliseconds (ms), which is the most important index to evaluate the dynamic response capability.

Total Expanded Nodes: The total number of times nodes are ejected from the priority queue during the whole simulation process, which reflects the degree of computational redundancy of the algorithm.

### 4.4 Experimental Results

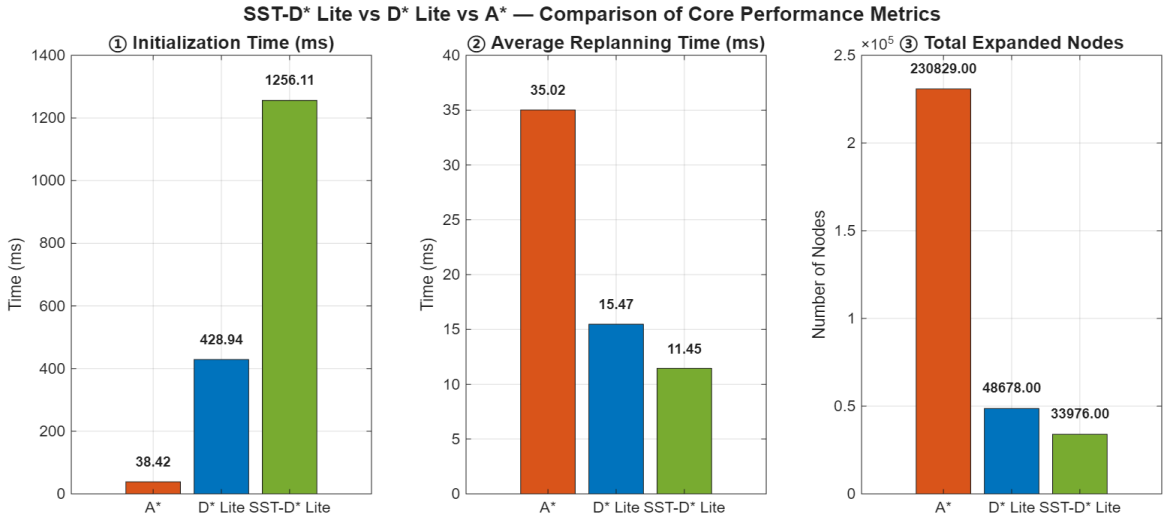


Figure 1. Comparison of Core Parameters: SST-D* Lite, D* Lite, and A*.

By comparing A, D Lite and our SST-D Lite algorithm in initialization time, average re-planning time and the total number of extended nodes, we can find that although SST-D Lite needs the longest initialization time (1256.11 milliseconds), far exceeding D Lite (428.94 milliseconds) and A (38.42 milliseconds), it is mainly due to search-space partitioning and State. However, our proposed algorithm shows great advantages in the core index of multi-objective dynamic performance.

Specifically, in terms of average replanning time, SST-D Lite requires only 11.45 ms, which is approximately 26% faster than D Lite (15.47 ms) and 67% shorter than A (35.02 ms), demonstrating its superior real-time response efficiency when coping with dynamic obstacle changes. Meanwhile, in terms of search efficiency and spatial complexity, the total number of expanded nodes for SST-D Lite is only 3.40×104, which is about 70% of that of D Lite (4.87×104) and only 14.7% of that of A (23.08×104). This fully verifies that the algorithm can filter redundant search space more effectively through its optimized heuristic guidance mechanism.

In summary, although SST-D Lite incurs a relatively high initialization cost in the early stage, its outstanding performance in dynamic replanning speed and search efficiency demonstrates its reliability and superiority for efficient multi-objective path planning in complex dynamic environments.

## 5. CONCLUSION AND FUTURE WORK

In short, this paper puts forward SST-D Lite algorithm to solve the problem of computational redundancy in multi-objective dynamic path planning. By constructing a search tree structure with shared trunk and independent branches, combined with lazy deletion mechanism and local reset strategy based on BFS, this method can effectively reuse historical path information and accurately correct the affected areas. The experimental results show that compared with the original D Lite algorithm, the efficiency of the new method is improved by about 26%, and the total number of extended nodes is greatly reduced, which proves that it is both real-time and reliable in complex dynamic environment.

Although this algorithm still has some shortcomings in start-up cost and adaptability to high-dimensional space, its new ideas in structure and strategy provide important theoretical support for us to solve multi-objective dynamic optimization problems. Our future work will focus on improving startup efficiency and exploring the engineering application value of this algorithm in high-dimensional complex space and large-scale scenes, such as unmanned aerial vehicles and robotic manipulators.

## REFERENCES

[1] Dijkstra, E. W. (2022). A note on two problems in connexion with graphs. In Edsger Wybe Dijkstra: his life, work, and legacy (pp. 287-290).
[2] Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. IEEE transactions on Systems Science and Cybernetics, 4(2), 100-107.
[3] Choset, H., Lynch, K. M., Hutchinson, S., Kantor, G. A., & Burgard, W. (2005). Principles of robot motion: theory, algorithms, and implementations. MIT press.
[4] Korf, R. E. (1985). Depth-first iterative-deepening: An optimal admissible tree search. Artificial intelligence, 27(1), 97-109.
[5] Stentz, A. (1994, May). Optimal and efficient path planning for partially-known environments. In Proceedings of the 1994 IEEE international conference on robotics and automation (pp. 3310-3317). IEEE.
[6] Koenig, S., Likhachev, M., & Furcy, D. (2004). Lifelong planning A ∗. Artificial Intelligence, 155(1-2), 93-146.
[7] Koenig, S., & Likhachev, M. (2002, July). D* lite. In Eighteenth national conference on Artificial intelligence (pp. 476-483).
[8] Koenig, S., & Likhachev, M. (2005). Fast replanning for navigation in unknown terrain. IEEE transactions on robotics, 21(3), 354-363.
[9] Likhachev, M., Ferguson, D. I., Gordon, G. J., Stentz, A., & Thrun, S. (2005, June). Anytime dynamic A*: An anytime, replanning algorithm. In ICAPS (Vol. 5, pp. 262-271).
[10] Ferguson, D., & Stentz, A. (2007). Field D*: An interpolation-based path planner and replanner. In Robotics Research: Results of the 12th International Symposium ISRR (pp. 239-253). Berlin, Heidelberg: Springer Berlin Heidelberg.
[11] Carsten, J., Ferguson, D., & Stentz, A. (2006, October). 3d field d: Improved path planning and replanning in three dimensions. In 2006 IEEE/RSJ international conference on intelligent robots and systems (pp. 3381-3386). IEEE.
[12] Stewart, B. S., & White III, C. C. (1991). Multiobjective a. Journal of the ACM (JACM), 38(4), 775-814.

[13] Standley, T. (2010, July). Finding optimal solutions to cooperative pathfinding problems. In Proceedings of the AAAI conference on artificial intelligence (Vol. 24, No. 1, pp. 173-178).
[14] Sharon, G., Stern, R., Felner, A., & Sturtevant, N. R. (2015). Conflict-based search for optimal multi-agent pathfinding. Artificial intelligence, 219, 40-66.
[15] Hernandez, C., Baier, J. A., & McIlraith, S. A. (2020). Multi-heuristic A* for dynamic environments. In 2020 IEEE International Conference on Robotics and Automation (ICRA) (pp. 5354-5360). IEEE.
[16] Wagner, G., & Choset, H. (2015). Subdimensional expansion for multirobot path planning. Artificial intelligence, 219, 1-24.
[17] Mandow, L., & De la Cruz, J. P. (2005, July). A New Approach to Multiobjective A* Search. In IJCAI (Vol. 8).
[18] Ren, J., Zhang, J., & Cui, Y. (2021). Autonomous obstacle avoidance algorithm for unmanned surface vehicles based on an improved velocity obstacle method. ISPRS International Journal of Geo-Information, 10(9), 618.