Suez Canal Engineering Energy and Environmental Science Journal
Faculty of Engineering – Suez Canal University
2025, Vol. 3 NO. 4, pages 20-29

Conflict-Based Search for Multi-Agent Path Planning Using D*Lite for Dynamic Re-planning: Analysis and Verification

Ahmed S. Abdel-Rahman[1], S.F. Nafea[1], Basem E. Elnaghi[1], and Ayman A. El-Badawy[2]

[1]Electrical Engineering Department, Faculty of Engineering, Suez Canal University, Ismailia, Egypt.
[2]Mechatronics Engineering Department, Faculty of Engineering, German University in Cairo, Cairo, Egypt.

**Corresponding author: Ahmed S. Abdel-Rahman, Email address: ahmed.soliman@eng.suez.edu.eg*
DOI: 10.21608/sceee.2025.375386.1074

Article Info:

Article History:
Received: 14\04\2025
Accepted: 05\08\2025
Published: 30\10\2025

DOI: 10.21608/sceee.2025.375386.1074

Abstract

Conflict-Based Search (CBS) is a widely used approach in multi-agent pathfinding due to its flexibility and effective conflict resolution capabilities. However, CBS encounters challenges in terms of path optimality and computational cost, especially in dynamic environments where obstacles frequently move and change. This study gives an analysis for integrating the D Lite algorithm as a replacement for A*. D* Lite introduces incremental replanning, allowing the algorithm to efficiently update paths without recalculating them entirely. This analysis shows the enhancement of CBS's adaptability to real-time dynamic environments, significantly reducing computational costs and improving responsiveness. By leveraging the ability of D* Lite to reuse portions of existing paths, CBS can resolve conflicts faster and maintain path optimality even as obstacles and agents move unpredictably. Simulations demonstrate that the optimized CBS with D* Lite achieves significantly reduced computation times than traditional CBS. This enhancement makes it more suitable for large-scale multi-agent systems operating in dynamic settings, such as autonomous vehicle coordination, swarm robotics, and warehouse automation, where real-time path adjustments are crucial.*

Keywords: *Conflict-Based Search (CBS), Multi-Agent Path Planning, D** Lite *Algorithm, Real-Time Pathfinding*

Suez Canal Engineering, Energy and Environmental Science Journal (2025), Vol. 3, No.4.

1. Introduction

Multi-agent Pathfinding (MAPF) is a crucial problem in systems such as robotics, logistics, and swarm systems, where multiple agents navigate shared spaces without collisions. Among the existing methods for solving this problem, CBS is one of the most promising approaches. The core idea of CBS follows a two-level hierarchy: a low-level planner that computes single-agent paths and a high-level planner that resolves conflicts between agents. This framework balances optimality and computational efficiency, thereby establishing CBS as the state-of-the-art method for solving MAPF in structured environments (Sharon et al., 2012).

Importantly, the algorithmic appeal of CBS extends to dynamic environments, such as Simultaneous Localization and Mapping (SLAM), where agents must simultaneously map, navigate, and explore a changing environment while efficiently resolving conflicts with other agents or obstacles. The hierarchical structure of CBS, which separates conflict detection from path optimization, makes it particularly well-suited for handling such tasks in a scalable and real-time adaptive manner (Chen et al., 2021).

Although CBS is highly effective, it has significant limitations. One major challenge is scalability: as the number of agents increases, the number of conflicts and constraints that the high-level search must resolve grows exponentially. Additionally, conventional CBS relies on the A* algorithm for low-level planning, which exhibits poor performance in dynamic environments due to frequent re-planning. This often results in suboptimal paths, increased computation time, and reduced efficiency, especially in scenarios involving moving obstacles or agents.

To address these limitations, this work presents an analysis of the enhanced version of CBS by replacing the A* algorithm with D*Lite as the low-level planner. *D**Lite is an incremental pathfinding algorithm that, unlike A*, does not recompute paths from scratch but instead reuses previous computations (S. Koenig & M. Likhachev, 2002). This characteristic makes it significantly more efficient in environments that require frequent path updates. By integrating D*Lite, the improved CBS framework demonstrates superior performance in dynamic and large-scale multi-agent systems, reducing computation time and improving path efficiency.

This study shows the main advantages of replacing the A* algorithm with D*Lite, these advantages can be addressed as follows:

- Enhancement of Efficiency in Dynamic Environments: The integration of D*Lite enables CBS to adapt paths more effectively in real-time, thereby reducing the necessity for complete path recalculations.
- Reduction of Computational Time: The incremental nature of D*Lite significantly decreases computation time, particularly in scenarios involving numerous agents or highly complex environments.
- Expansion of CBS Applicability: The enhanced CBS framework is better suited for real-world applications such as warehouse automation, search-and-rescue missions, and urban mobility, where dynamic environmental changes and large agent populations are prevalent.

To demonstrate the effectiveness of Enhanced CBS using D*Lite*, various simulation tests were carried out. The results improved significantly with traditional CBS in execution time, nodes visited, and the length of the path for all scenarios. These results put the enhanced CBS algorithm as a scalable, robust solution to MAPF in any dynamic setting.

The structure of the paper is organized in the following manner: Sect. 2 presents the state of the art, focusing on the advancement of CBS and the related works in this context. Section 3 explains the approach in detail, with particular emphasis on D*Lite integration. Section 4 describes the features of the experimental environment and the parameters for assessing the performance of the system. Sections 5 and 6 provide the results with discussion and conclusions, respectively, while also indicating the important aspects and paths for the future and present work.

2. Related Work

Conflict-Based Search (CBS) has been extensively studied and adapted for MAPF due to its robust hierarchical design. By separating high-level conflict resolution from low-level pathfinding, CBS balances computational efficiency and scalability, making it a widely promising algorithm in both research and practical applications.

Since the development of CBS, a methodology in artificial intelligence, various studies and modifications have been done on the idea for successful applications. It was first set up by Sharon et al., and afterward, as a development in trails (Sharon et al., 2012), CBS presented an effective means of addressing agent conflicts in a two-level structure. Even in such shared spaces, CBS takes advantage of the absence of conflicts by using a separation of high-level conflict resolution through Conflict Tree CT from low-level pathfinding using A*. Improvements in the initial algorithm of CBS have focused on reducing the problem of computation while increasing scalability.

In particular, Boyarski et al. aimed at improving CBS (ICBS), which employed heuristics for effective narrowing down of the search area (Boyarski et al., 2015). Walker et al. (Walker et al., 2019) introduced advances, including Unbounded Sub-Optimal CBS to address difficult issues with relaxation of optimality. To solve those problems. Li et al. (Li et al., 2019) also optimized CBS for cases that involve a large number of agent traces by disjoint splitting as opposed to generating constraints overhead in doing so. In recent developments, the emphasis has mainly been on flexibility. For example, Gange et al. (Gange et al., 2019) proposed Lazy CBS, which makes use of lazy clause generation to postpone the evaluation of conflicts and, hence, improve the runtime of the system. Walker et al. (Walker et al., 2021) proposed a Conflict-Based Increasing Cost Search (CBICS) where the re-computation is done just before the search, and the course cost changes dynamically to reduce the appropriate calculations.

In addition, Lee et al. (Lee et al., 2021) presented Parallel Hierarchical Composition CBS, which improved the performance of the multi-agent system on a large scale by utilizing parallel computation strategies. Wen et al. (Wen et

al., 2022) proposed a Multi-Agent Path Finding for Car-Like robots (CL-MAP), an efficient method to solve the path planning problem for multiple car-like robots. This new method is based on Spatiotemporal Hybrid-State A* for low-level planner for a single agent, where A* serves as a high-level planner, producing an initial spatiotemporal path. And MPC refines and follows this path in real-time, ensuring smooth and dynamically feasible execution.

This hybrid approach balances global planning (A*) and local adaptation (MPC), which is essential for real-world, multi-agent navigation. In (Honig et al., 2022), Honig et al. proposed a db-A* as a single-robot motion planner designed to address time-optimal motion planning for systems with kinodynamic constraints, such as differential-drive robots, cars, airplanes, and multirotors. The key innovation of db-A* is its integration of graph-search algorithms with trajectory optimization: 1. Graph Search with Bounded Discontinuities: The algorithm extends traditional A* search into continuous space by allowing for small, user-specified discontinuities at the vertices, facilitating the use of precomputed motion primitives. 2. Trajectory Optimization: Locally repairs and optimizes these trajectories to ensure they meet the kinodynamic constraints of the robot, improving both feasibility and optimality. Moldagalieva et al. (Moldagalieva et al., 2024) proposed the Discontinuity-Bounded Conflict-Based Search (db-CBS) algorithm, which combines (CBS) with a single-robot kinodynamic motion planner called discontinuity-bounded A* (db-A*). This method operates on three levels: 1- Individual Trajectory Planning: Utilizes db-A* to compute initial trajectories for each robot, allowing for bounded discontinuities between precomputed motion primitives. 2- Collision Detection and Resolution: Identifies inter-robot collisions and resolves them by imposing constraints on the individual planners. 3- Trajectory Optimization: Refines the solution using joint space trajectory optimization, iteratively reducing the allowed discontinuity bounds to enhance solution quality. Lin et al. (Lin et al., 2024) proposed a priority-based Search algorithm tailored for heterogeneous mobile robots operating in continuous spaces. The primary focus is on addressing both geometric and kinematic constraints unique to each robot, ensuring collision-free paths that respect these individual limitations.

The traditional (CBS) algorithms, however, face challenges in dynamic and real-time environments, where obstacles and goals continuously change. To address this, the concept of subdimensional expansion has been introduced(Wagner & Choset, 2015), enabling agents to generate short-lived sub-problems for faster planning. Andreychuk et al. (Andreychuk et al., 2019) focused on continuous-time scenarios and explored the potential for addressing CBS's limitations by integrating SAT modulo theory to resolve the illumination problems of Safe Interval Path Planning (SIPP). In lifelong MAPF, CBS has been modified to handle dynamic pickup and delivery tasks. Ma et al (Ma et al., 2017). demonstrated a practical example of this approach, highlighting how agents with evolving missions can effectively utilize conflict-based strategies.

Recent advancements in CBS have also emphasized planning for UAV swarms—for instance, Wang et al. (Wang et al., 2024) the SL-CBS framework combines state lattices with CBS to facilitate motion planning under time and space constraints. These developments further extend CBS's applicability to complex and dynamic multi-agent systems. The performance of CBS is very much affected by its low-level planner. Though most practicing researchers still use A* due to its ease and proven effectiveness, it has been proven impractical in dynamic environments thus, other methods have been sought. To address this drawback, Koenig and Likhachev (S. Koenig & M. Likhachev, 2002) proposed a more dynamic-responsive D*Lite, which is an incremental replanning algorithm. More recent studies proposed to modify the CBS algorithm by incorporating other planning algorithms to reach a compromise between efficiency and optimality. Ivanashev et al.(Ivanashev et al., 2022) integrated adaptations of CBS with focal search methods to provide solutions that are bounded on optimality while still being efficient in computation. In their paper, Walker and Sturtevant (Walker & Sturtevant, 2024) examined the completeness of CBS in the presence of different levels of temporal constraints and introduced algorithms for temporally relative duplicate elimination for more efficient low-level pathfinding.

This study presents an analysis of CBS by incorporating D*Lite as the low-level planner. Unlike previous studies that primarily focused on tree optimizations and heuristic enhancements at the database level, this work addresses pathfinding improvements at the most fundamental level of operation. The integration of D*Lite enables incremental preplanning, thereby reducing unnecessary computational overhead. This enhancement prevents avoidable recalculations and improves overall performance. In cases where conventional CBS exhibits suboptimal execution, the proposed approach significantly improves both execution time and path quality, particularly in dynamic environments. Building upon the foundational work of Sharon et al. (Sharon et al., 2012), and Jin et al, (Jin et al., 2023) who incorporate contemporary advancements of CBS and D*Lite, This study presents a scalable, flexible, and deployable analysis for challenges in multi-agent systems. The inclusion of D*Lite with CBS's applicability makes it particularly beneficial for environments where terrain is non-static or where agent objectives may change dynamically.

3. Methodology

This work focuses on simulating the CBS algorithm in two scenarios. The first scenario was used to test CBS-A*, each scenario was done twice, first to test the (4-Points) neighbor search, which searches using the main four directions (north, east, south, and west) in 3D, and the second to do the same scenario for the same algorithm using the second way of neighbor's search (8-Points), means the main four directions plus the diagonal directions in 3D. Figure 1 shows a

pseudocode for solving a MAPF problem using CBS with D*Lite as the low-level planner. The following words explain these stages:

- Integration of D*Lite into CBS: The D*Lite algorithm replaces A* for pathfinding at the low level. This allows the system to quickly adapt to changes in the environment, such as new obstacles or moving agents, by reusing previous computations.
- Dynamic Path Re-planning: Instead of re-computing the entire path for an agent, D*Lite updates only the affected portions, significantly reducing execution time in dynamic settings.

```
    Input: MAPF instance
    Output: MAPF Solution
1   Constraints = 0
2   In it MAPF Solution = find individual paths using the low-level
3   Open list = 0
4   While OPEN list is not empty, do
5       Current node = lowest cost from the Open list
6       If the Current node has no conflict, then
7           Return solution
8       end
9       children = create Constrained Nodes
10      Re-plan the path for a constrained agent using D* Lite
11      Add valid children to the open list
12  End
```

Figure 1: Pseudocode for solving a MAPF problem using CBS with D*Lite

In D*Lite, the priority of each node s is calculated as:

$$key(s) = \left[min\big(g(s), rhs(s)\big) + h(s_{start}, s), min\big(g(s), rhs(s)\big)\right] \quad (1)$$

Here, $g(s)$ denotes the current cost estimate from the node s to the goal and $rhs(s)$ is the one-step lookahead value, which provides a more immediate estimate based on the costs of successors. The function $h(s_{start}, s)$ is a heuristic that estimates the cost from the start node s_{start} to the node s.

$$rhs(s) = \begin{cases} 0 & if\ S_i = G_i \\ min_{S_i' \in Succ(s)}[g(S_i') + c(S_i', S_i)] & otherwise \end{cases} \quad (2)$$

Where $Succ(s)$ is a set of successors nodes of s (neighbors in the graph), and $c(s', s)$ represents the cost of moving from the successors s' to s. Nodes are updated dynamically as the environment changes.

The objective is to compute a path P_i for each agent i, given its start and goal nodes $S_i$and G_i , and a set of high-level constraints $Constraints_i$, as follows:

$$P_i = D * Lite(G, S_i, G_i, Constraints_i) \quad (3)$$

where $G = (V, E)$ is the environment graph with the vertex set V and edge set E, and $Constraints_i$ They are derived from a high-level CBS framework to avoid collisions among multiple agents.

That minimizes the path cost while satisfying the constraints, the proposed method offers significant advantages in terms of scalability, making it highly efficient for large-scale applications. One of the key benefits is the reduced computational time, achieved through the incremental updates in D* Lite. Unlike traditional path-planning approaches that require recalculating the entire path when changes occur, D* Lite efficiently updates only the affected portions. This feature significantly reduces processing overhead, making the method suitable for real-time applications where rapid decision-making is crucial.

Another major strength of this approach is its dynamic adaptability. In environments that are constantly changing, such as those encountered in autonomous navigation or multi-robot coordination, the ability to quickly respond to new obstacles or altered routes is essential. The method effectively handles real-time modifications, ensuring continuous path optimization without compromising performance. This adaptability makes it particularly useful in scenarios where unpredictability is a key challenge, such as search-and-rescue missions or urban traffic navigation for autonomous

vehicles. Beyond its theoretical advantages, the practical applications of this method make it highly valuable in real-world scenarios. It improves performance in areas such as autonomous vehicles, robotic fleets, and drone swarms operating in dynamic environments. The ability to swiftly adjust to environmental changes allows these systems to function with greater efficiency and safety. Whether deployed in automated warehouses, traffic management, or aerial surveillance, the method provides a robust solution for complex, real-time navigation challenges.

4. The environments and dataset

To evaluate the performance of the D* Lite-Enhanced CBS approach, we conducted a series of simulation tests in environments designed to represent dynamic and complex multi-agent scenarios. These environments were carefully constructed to include varying obstacle densities, dynamic obstacles created by moving agents, and diverse start and goal configurations. The primary objective was to assess the algorithm's ability to handle real-time adjustments and effectively resolve conflicts in multi-agent path-planning situations. One of the test environments was the Static Maze Environment, which consists of a structured maze with predefined narrow corridors and static obstacles. This setup represents a low-dynamic environment where obstacle placement remains constant throughout the simulation. The primary purpose of this scenario was to evaluate the algorithm's efficiency in finding optimal paths in constrained spaces without the need for frequent path adjustments. The second environment, Dynamic Open Field, was designed as a spacious area with sparsely placed mobile obstacles that change positions at regular intervals. This scenario was specifically chosen to test the adaptability of the D* Lite-Enhanced CBS approach in handling real-time environmental changes. By continuously altering the obstacle positions, the environment simulates unpredictable navigation challenges, requiring the algorithm to dynamically update paths and maintain optimal routing.

Lastly, the Crowded Warehouse Layout was developed to replicate a dense, grid-like warehouse setting with numerous agents and obstacles that randomly appear and disappear. This environment presents a highly dynamic and conflict-prone scenario, where multiple agents must frequently re-plan their paths due to the continuous emergence of obstacles. The goal of this test was to assess the algorithm's capability to resolve frequent conflicts and maintain efficient path planning in a highly congested and unpredictable space. Each of these environments was carefully selected to mimic realistic multi-agent path-planning challenges in applications such as autonomous warehouses, swarm robotics, and real-time navigation in obstacle-rich settings. By evaluating the D* Lite-Enhanced CBS approach across diverse scenarios, we aimed to demonstrate its robustness and adaptability in solving complex, real-world path-planning problems. In each environment, agents were assigned random start and goal locations, ensuring variability across different simulation runs. For every iteration of the experiment, a new set of positions was generated, allowing for a diverse range of test cases. In dynamic environments, the positions and movement patterns of obstacles were also randomized to simulate realistic and unpredictable conditions. This approach ensured a robust evaluation of the algorithm's adaptability, as it was tested under various levels of environmental complexity and dynamism. By incorporating randomized datasets, we aimed to provide a thorough assessment of the algorithm's capability to handle changing scenarios efficiently. To assess the performance of the Enhanced CBS approach using D* Lite, we employed several key evaluation metrics. One of the primary metrics was Execution Time (s), which measures the total time required to compute paths for all agents. This metric reflects the computational efficiency of the algorithm and is crucial for real-time applications where rapid decision-making is required. Another important metric was Nodes Explored, which counts the total number of nodes expanded by the low-level planner. This value serves as an indicator of the algorithm's search efficiency and effectiveness in minimizing redundant computations. A lower number of nodes explored suggests that the algorithm is able to generate paths more efficiently without excessive re-exploration.

Lastly, we evaluated Path Length, which represents the total distance traveled by each agent from its start position to its goal. This metric assesses the optimality of the paths generated by the D* Lite-Enhanced CBS approach. Shorter path lengths indicate that the algorithm is capable of finding efficient routes, minimizing travel distance and time for agents navigating through dynamic environments. These evaluation metrics provide a comprehensive assessment of the algorithm's execution efficiency, search performance, and path quality. By comparing the results against traditional CBS and other CBS variants, we demonstrate how the integration of D* Lite enhances CBS performance in dynamic environments. The results highlight the advantages of reducing computational overhead while simultaneously improving path optimality, making the method well-suited for real-world applications in autonomous navigation and multi-agent coordination.

5. Results and Analysis

The figure (2) shows a 3D voxel-based environment where two agents, represented by red and blue paths, navigate through a structured space filled with obstacles. The primary objective is to ensure that both agents reach their respective destinations while avoiding conflicts or collisions along their trajectories. The central part of the figure (2) depicts a 3D

voxel representation of the environment, with black cubic structures of varying heights representing obstacles. Within this space, two agents are following their planned paths, indicated by dashed blue and red lines.

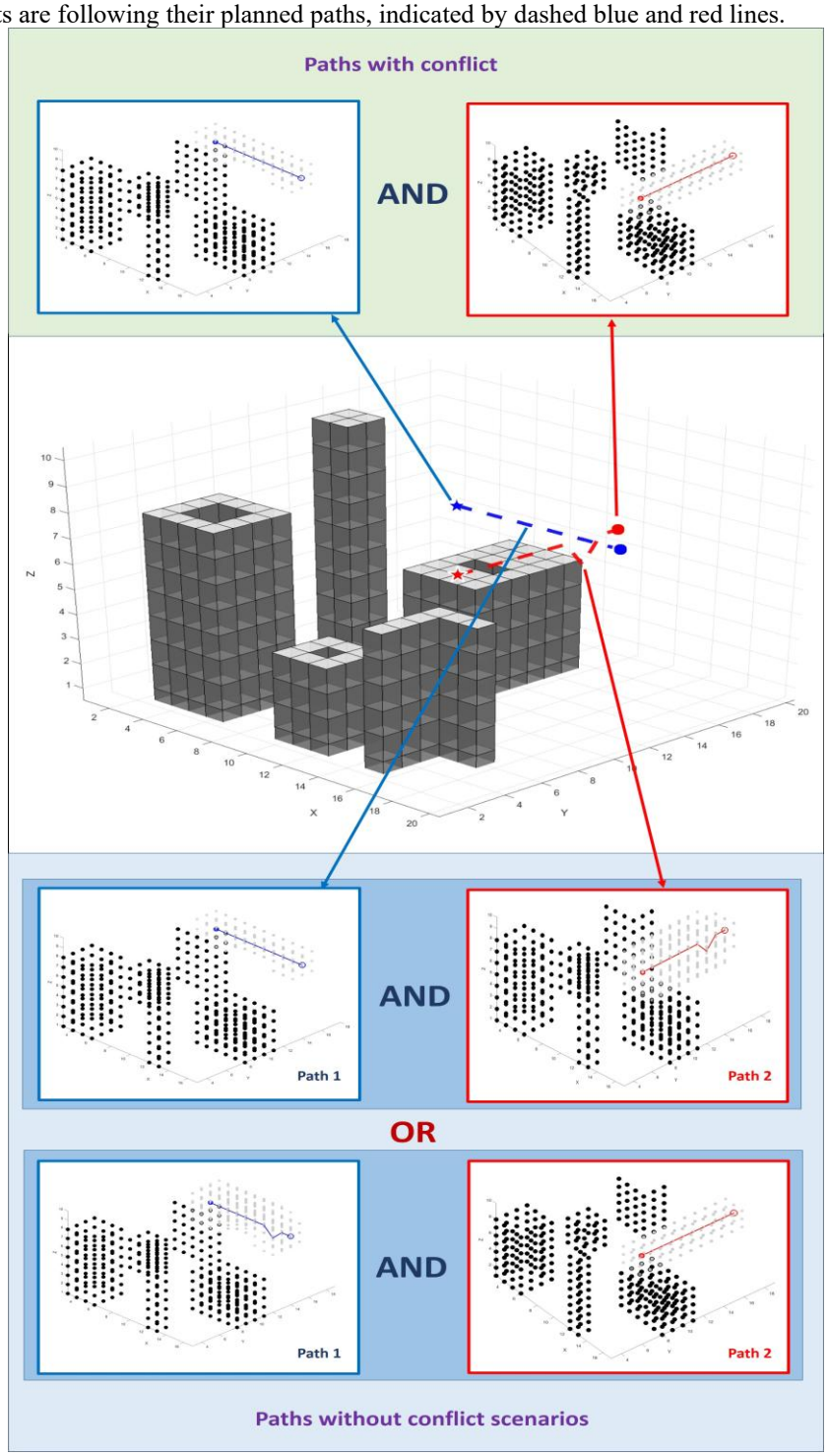

Figure 2: two agent conflict path modification scenarios

Their starting positions are marked with stars, and dots represent their step-by-step movement through the environment. Initially, the paths intersect at a particular location, highlighting a conflict scenario where the agents would collide if their paths are not modified as shown in in the upper section of the figure (2) that labeled "Paths with conflict," the initial path planning is displayed, showing a direct conflict between the two agents. These panels provide a 2D view of the agents' movement patterns, illustrating how both paths cross at the same point, which leads to a collision. Without intervention, both agents would attempt to occupy the same voxel at the same time, making this an unsafe trajectory.

To resolve the conflict, the lower section of the figure (2), labeled "Paths without conflict scenarios," presents conflict-free path planning. Here, the paths have been adjusted using CBS to prevent collisions while maintaining optimal travel efficiency. The diagram provides two possible solutions to avoid intersection while still allowing the agents to reach their destination efficiently. Figure (2) provides a clear visual representation of the necessity of conflict resolution in multi-agent path planning. The comparison between the upper and lower sections highlights the importance of optimizing paths to avoid collisions while maintaining efficient navigation. Through the use of CBS, the agents can successfully navigate the environment without interfering with each other's movement, ensuring a smooth and coordinated travel path.

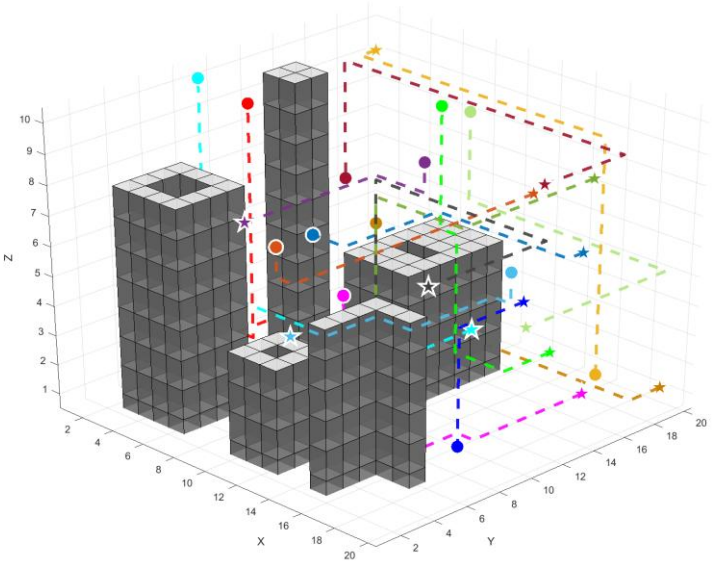

Figure 3: CBS-A* using 4-Points neighbors

The performance of the Enhanced CBS using D*Lite algorithm was evaluated and compared to standard CBS (using A* as the low-level planner) based on quantitative metrics and qualitative observations. The analysis highlights improvements in execution time, nodes explored, and path length across various testing environments. Figure (3) illustrates a 3D path-planning scenario using CBS-A* for multi-agent navigation.

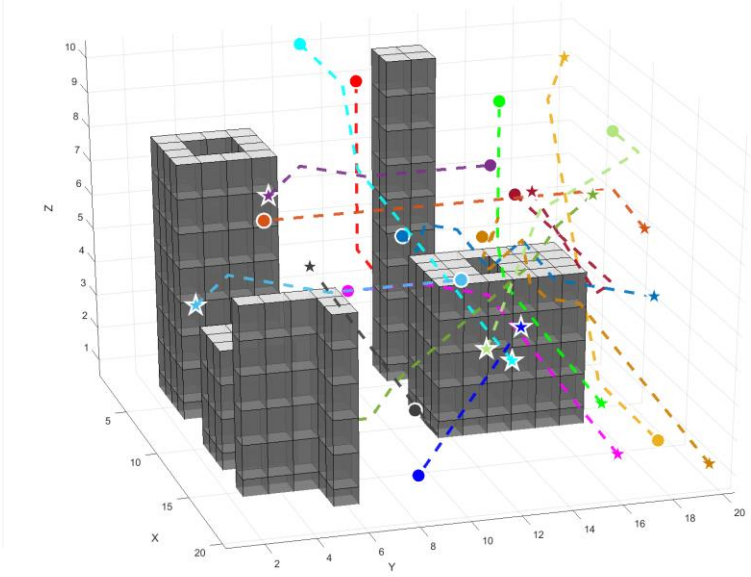

Figure 4: CBS-A* using 8-Points neighbors

The environment comprises a voxel-based grid with obstacles illustrated as dark gray stacked cubes. In this test, 15 agents are deployed to navigate from their respective start positions to their goal positions while avoiding collisions with other agents. The agents' paths are visualized using dashed colored lines, with each agent represented by an individually colored sphere. The CBS-A* algorithm effectively coordinates the agents, resolving conflicts dynamically

while ensuring efficient pathfinding. The successful traversal of all agents without conflicts highlights the robustness of CBS-A* in handling complex multi-agent planning in 3D environments. In this scenario, the CBS-A* version 4-Points created paths for 15 agents in 8.64 seconds, while the visited points for all 15 agents were 4106 points, and the average path length was 19.60 units. Figure (4) illustrates CBS-A* version 8-Points, which created paths for 15 agents in 0.51 seconds, while the visited points for all 15 agents were 178 points, and the average path length was 11.60 units. In Figure (5), the CBS-D*Lite version 4-Points created paths for 15 agents in 0.03 seconds, while the visited points for all 15 agents were 303 points, and the average path length was 19.53 units. Figure (6) illustrates CBS-D*Lite version 8-Points, which created paths for 15 agents in 0.05 seconds, while the visited points for all 15 agents were 198 points, and the average path length was 11.93 units.

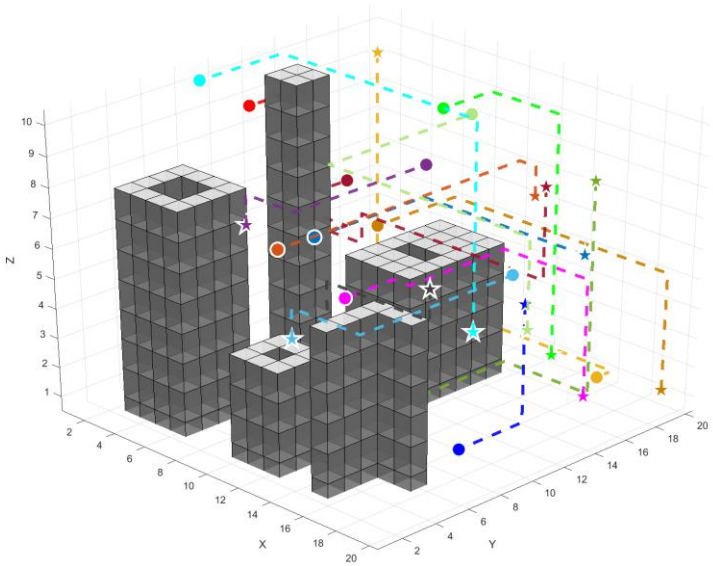

Figure 5: CBS-D* using 4-Points neighbors

The results demonstrate clear enhancements in performance with the integration of D* Lite into CBS. These enhancements highlight improved efficiency, faster re-planning, and better adaptability in dynamic environments, making the approach more effective for real-time applications as follows:

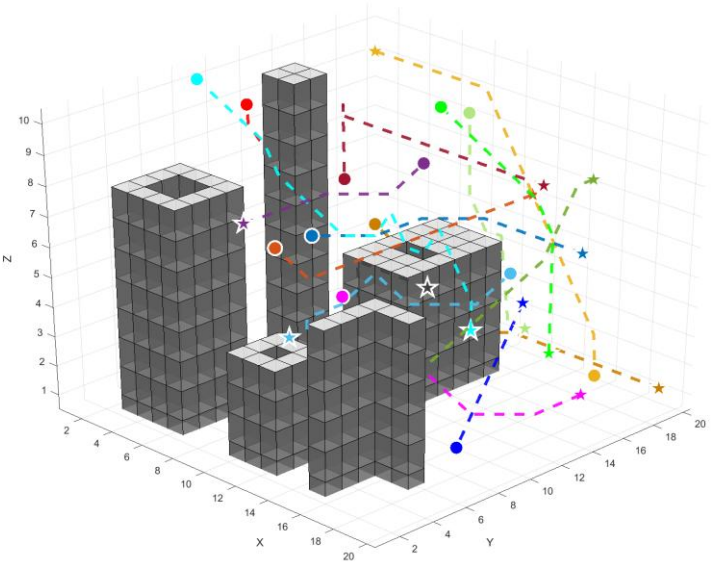

Figure 6: CBS-D* using 8-Points neighbors

Firstly, the Execution Time of the Enhanced CBS using D* Lite significantly reduced computation time, achieving an average improvement of 99.2% over standard CBS. The incremental path updates in D* Lite allowed for faster re-planning, particularly in dynamic environments where frequent path recalculations were necessary. This efficiency makes the method well-suited for real-time applications requiring rapid decision-making. Secondly, the Nodes Explored as

number of nodes explored per agent decreased by approximately 88% with Enhanced CBS using D* Lite. This reduction highlights the efficiency of the D* Lite incremental approach, which minimizes redundant node expansion. By focusing on updating only the affected portions of the path, the algorithm improves the efficiency of low-level pathfinding, reducing computational overhead.

The following section presents a qualitative evaluation of the Enhanced CBS using D* Lite approach. Table 1 provides a comparison between A* and D* Lite when used with 15 agents, demonstrating the efficiency gains of the latter in dynamic environments. Table 2 summarizes the average results across multiple test scenarios, offering a comprehensive overview of execution time, nodes explored, and path length improvements. Additionally, qualitative observations, including adaptability in dynamic settings, conflict resolution efficiency, and scalability for multi-agent scenarios, further illustrate the advantages of the proposed approach.

This enhancement presents an Adaptability in Dynamic Environments as in environments with moving obstacles, Enhanced CBS using D*Lite adapted to changes more smoothly than standard CBS. The algorithm required fewer full recalculations, reducing disruptions and allowing agents to continue toward their goals without significant detours. This adaptability is crucial for real-time applications where obstacles may appear unpredictably.

Also, the Efficiency in Conflict Resolution as Enhanced CBS using D*Lite displayed improved responsiveness in resolving conflicts among agents. In scenarios with high agent density, where conflicts occurred frequently, the incremental updates of D*Lite allowed each agent to find alternative paths with minimal delay. This approach led to quicker conflict resolution and less congestion within the environment.

In addition, the Scalability for Multi-Agent Scenarios as the Enhanced CBS using D*Lite approach scaled better with increasing agent numbers. In tests with a large number of agents (e.g., 15 + agents), standard CBS faced significant slowdowns as it recalculated paths more frequently. In contrast, Enhanced CBS using D*Lite maintained better performance, handling additional agents efficiently by leveraging D*Lite's capability to reuse parts of existing paths.

Table 1. Comparison between A* and D*lite while using 15 agents

Algorithm	Execution Time (s)		Nodes Explored		Path Length	
	4-Points	8-Points	4-Points	8-Points	4-Points	8-Points
A*	8.64	0.51	4106	178	19.60	11.60
D*Lite	0.03	0.05	303	198	19.53	11.93

Table 2: Summarization of average results for each metric across multiple test scenarios

Metric	Standard CBS (A*)	D*Lite Enhanced CBS	Improvement (%)
Execution Time (s)	5.575	0.04	99.2%
Nodes Explored	2142	250.5	88.3%
Path Length (avg.)	15.6	15.73	-0.8%

6. Conclusion

This study analyzes the integration of the D* Lite algorithm as a replacement for A* in pathfinding applications. It evaluates D* Lite efficiency in dynamic environments, highlighting its real-time adaptability and reduced computational cost compared to A*. The comparison emphasizes its potential for improving performance in uncertain, changing scenarios. The integration of D* Lite as the low-level planner within the CBS architecture has shown clear advantages over standard CBS with A*, improving computational efficiency and path quality across different agent counts. The experimental results confirm significant enhancements in execution time, search efficiency, and path optimality, demonstrating the effectiveness of Enhanced CBS using D* Lite.

Enhanced CBS using D* Lite consistently achieved lower execution times. For example, in the three-agent case, execution time dropped from 15.24 seconds with A* to just 0.22 seconds in the case of using D*Lite. This reduction is due to D* Lite has enhanced its cost function and its search way also the ability to reuse and modify existing paths instead of recalculating them entirely, improving efficiency, especially in real-time applications. The number of nodes explored also decreased significantly. With six agents, Enhanced CBS using D* Lite expanded only 39 nodes compared to 874 nodes with A*, highlighting its efficient search strategy. By focusing only on affected path segments, the algorithm reduces unnecessary computations, making it well-suited for dynamic environments. Overall, Enhanced CBS using D* Lite optimizes multi-agent pathfinding by reducing computational costs and improving search efficiency. These improvements make it highly suitable for real-time applications in dynamic environments, such as robotics and autonomous navigation.

How to Cite this Article:
Abdel-Rahman A.S. et al. (2025). Conflict-Based Search for Multi-Agent Path Planning Using D*Lite for Dynamic Re-planning: Analysis and Verification. *Suez Canal Engineering, Energy and Environmental Science Journal,* 3 (4), p.p. 20-29.

References

Andreychuk, A., Yakovlev, K., Atzmon, D., & Sternr, R. (2019). Multi-agent pathfinding with continuous time. *IJCAI International Joint Conference on Artificial Intelligence*. https://doi.org/10.24963/ijcai.2019/6

Boyarski, E., Feiner, A., Sharon, G., & Stern, R. (2015). Don't split, try to work it out: Bypassing conflicts in multi-agent pathfinding. *Proceedings International Conference on Automated Planning and Scheduling, ICAPS*, *2015-January*, 47–51. https://doi.org/10.1609/icaps.v25i1.13725

Chen, J., Li, J., Fan, C., & Williams, B. C. (2021). Scalable and Safe Multi-Agent Motion Planning with Nonlinear Dynamics and Bounded Disturbances. *35th AAAI Conference on Artificial Intelligence, AAAI 2021*, *13A*, 11237–11245. https://doi.org/10.1609/aaai.v35i13.17340

Gange, G., Harabor, D., & Stuckey, P. J. (2019). Lazy CBS: Implicit conflict-based search using lazy clause generation. *Proceedings International Conference on Automated Planning and Scheduling, ICAPS*. https://doi.org/10.1609/icaps.v29i1.3471

Honig, W., Ortiz-Haro, J., & Toussaint, M. (2022). db-A*: Discontinuity-bounded Search for Kinodynamic Mobile Robot Motion Planning. *IEEE International Conference on Intelligent Robots and Systems*, *2022-October*, 13540–13547. https://doi.org/10.1109/IROS47612.2022.9981577

Ivanashev, I., Andreychuk, A., & Yakovlev, K. (2022). Analysis of the Anytime MAPF Solvers Based on the Combination of Conflict-Based Search (CBS) and Focal Search (FS). *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. https://doi.org/10.1007/978-3-031-19493-1_29

Jin, J., Zhang, Y., Zhou, Z., Jin, M., Yang, X., & Hu, F. (2023). Conflict-based search with D* lite algorithm for robot path planning in unknown dynamic environments. *Computers and Electrical Engineering*, *105*. https://doi.org/10.1016/j.compeleceng.2022.108473

Lee, H., Motes, J., Morales, M., & Amato, N. (2021). Parallel Hierarchical Composition Conflict-Based Search for Optimal Multi-Agent Pathfinding. *IEEE Robotics and Automation Letters*. https://doi.org/10.1109/LRA.2021.3096476

Li, J., Harabor, D., Stuckey, P. J., Felner, A., Ma, H., & Koenig, S. (2019). Disjoint splitting for multi-agent path finding with conflict-based search. *Proceedings International Conference on Automated Planning and Scheduling, ICAPS*. https://doi.org/10.1609/icaps.v29i1.3487

Lin, W., Song, W., Zhu, Q., & Zhu, S. (2024). Multi-Agent Path Finding with Heterogeneous Geometric and Kinematic Constraints in Continuous Space. *IEEE Robotics and Automation Letters*. https://doi.org/10.1109/LRA.2024.3511435

Ma, H., Kumar, T. K. S., Li, J., & Koenig, S. (2017). Lifelong multi-Agent path finding for online pickup and delivery tasks. *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS*. https://doi.org/DOI: 10.48550/arXiv.1705.10868

Moldagalieva, A., Ortiz-Haro, J., Toussaint, M., & Hönig, W. (2024). Db-CBS: Discontinuity-Bounded Conflict-Based Search for Multi-Robot Kinodynamic Motion Planning. *Proceedings - IEEE International Conference on Robotics and Automation*, 14569–14575. https://doi.org/10.1109/ICRA57147.2024.10610999

S. Koenig, & M. Likhachev. (2002). D-Lite*. *Proceedings of the AAAI Conference on Artificial Intelligence*, 476–483.

Sharon, G., Stern, R., Felner, A., & Sturtevant, N. (2012). Conflict-Based Search For Optimal Multi-Agent Path Finding. *Proceedings of the 26th AAAI Conference on Artificial Intelligence, AAAI 2012*. https://doi.org/10.48175/ijarsct-8161

Wagner, G., & Choset, H. (2015). Subdimensional expansion for multirobot path planning. *Artificial Intelligence*. https://doi.org/10.1016/j.artint.2014.11.001

Walker, T. T., & Sturtevant, N. R. (2024). *On the Completeness of Conflict-Based Search: Temporally-Relative Duplicate Pruning*.

Walker, T. T., Sturtevant, N. R., & Felner, A. (2019). Unbounded sub-optimal conflict-based search in complex domains. *Proceedings of the 12th International Symposium on Combinatorial Search, SoCS 2019*. https://doi.org/10.1609/socs.v10i1.18492

Walker, T. T., Sturtevant, N. R., Felner, A., Zhang, H., Li, J., & Kumar, T. K. S. (2021). Conflict-Based Increasing Cost Search. *Proceedings International Conference on Automated Planning and Scheduling, ICAPS*. https://doi.org/10.1609/icaps.v31i1.15984

Wang, Z., Zhang, Z., Dou, W., Hu, G., Zhang, L., & Zhang, M. (2024). *Extending Conflict-Based Search for Optimal and Efficient Quadrotor Swarm Motion Planning*. https://doi.org/10.20944/preprints202410.1212.v1

Wen, L., Liu, Y., & Li, H. (2022). CL-MAPF: Multi-Agent Path Finding for Car-Like robots with kinematic and spatiotemporal constraints. *Robotics and Autonomous Systems*, *150*. https://doi.org/10.1016/j.robot.2021.103997