Optimizing Warehouse Throughput via Conflict-Based Search for Multi-Agent Pathfinding

Professor Fangzhen Lin, Jinhong Huo
Hong Kong University of Science and Technology (HKUST)
jhuoab@connect.ust.hk

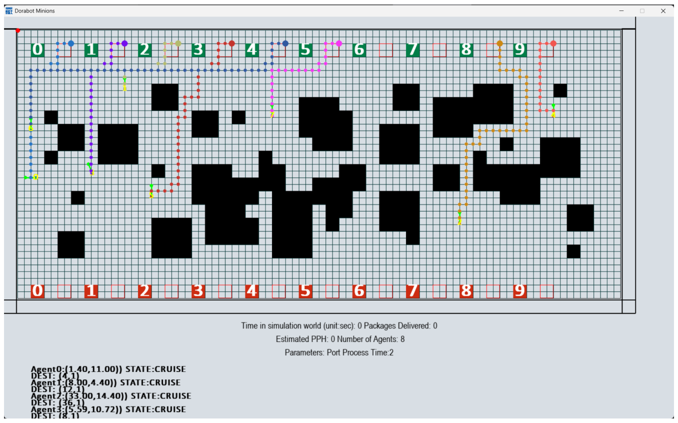
***Abstract*—The rapid growth of e-commerce has led to the proliferation of large-scale automated warehouses deploying hundreds of mobile robots. A critical challenge in these environments is the Multi-Agent Pathfinding (MAPF) problem: computing collision-free paths for all agents to maximize throughput. As the number of agents increases, the computational complexity of finding optimal paths grows exponentially, often leading to gridlock and reduced efficiency. This paper addresses this challenge by implementing and evaluating the Conflict-Based Search (CBS) algorithm. CBS decomposes the problem by finding individual agent paths and then resolving conflicts in a structured, two-level search. We hypothesize that CBS can significantly improve warehouse throughput by generating globally coordinated, optimal paths, thereby reducing congestion and agent idle time. The algorithm is tested in a high-fidelity simulation environment provided by Shenzhen Dorabot Inc.**

***Index Terms*—Multi-Agent Pathfinding (MAPF), Conflict-Based Search (CBS), Robotics, Warehouse Automation, Logistics, Path Planning**

I. Introduction

The logistics industry is in the midst of a robotic revolution, driven by the immense pressure of e-commerce. While deploying hundreds of robots promises immense gains in efficiency, it introduces a formidable operational challenge: the Multi-Agent Pathfinding (MAPF) problem. At its core, MAPF is the task of computing collision-free paths for every robot, allowing each to travel from its start point to its destination within a shared space. The difficulty lies in the problem's computational complexity. This means that as the number of robots increases, the computational effort required to find the perfect set of paths can grow exponentially, leading to traffic jams, gridlock, and deadlocks that throttle warehouse throughput.

To address this coordination bottleneck, this project implements and evaluates Conflict-Based Search (CBS), a powerful and widely-studied algorithm for solving the MAPF problem. Unlike methods that tackle the massive combined state space of all robots, CBS decomposes the problem. It first finds an optimal path for each robot individually and then systematically identifies and resolves the specific points of conflict where collisions would occur. To validate our approach, we will test the algorithm's performance using a simulation environment provided by Shenzhen Dorabot Inc. The central hypothesis is that by generating optimal, collision-free paths, CBS can significantly reduce robot idle time and congestion, leading directly to a measurable increase in task completion rates and optimizing overall warehouse throughput.



```
Time in simulation world (unit:sec): 0 Packages Delivered: 0
Estimated PPH: 0 Number of Agents: 8
Parameters: Port Process Time:2

Agent0:(1.40,11.00)) STATE:CRUISE
DEST: (4,1)
Agent1:(8.00,4.40)) STATE:CRUISE
DEST: (12,1)
Agent2:(33.00,14.40)) STATE:CRUISE
DEST: (36,1)
Agent3:(5.59,10.72)) STATE:CRUISE
DEST: (8,1)
```

Fig. 1. The 2D simulation environment from Shenzhen Dorabot Inc., showing agents (Yellow number), static obstacles (Black Blocks), loading ports (Green Squares), and unloading ports (Red Squares).

II. Problem Definition

The core challenge is to solve the Multi-Agent Pathfinding (MAPF) problem within the specific constraints of the provided simulation environment from Shenzhen Dorabot Inc. As shown on the simlulation figure 1, the environment is defined by a set of configuration files that specify: a 2D map with a given height and width; a list of static, square obstacles of varying sizes; a fleet of N homogenous agents; and a set of designated loading and unloading port locations. The primary objective is to assign tasks (a journey from a loading port to an unloading port) to idle agents and compute a complete, collision-free path for every agent to execute its task concurrently.

This pathfinding is managed by a two-level planning hierarchy. A global planner is responsible for determining the high-level route for each agent, considering the entire map, static obstacles, and the paths of other agents to solve the core MAPF problem. This route is typically a sequence of waypoints. A local planner is then responsible for navigating between these waypoints, generating a smooth, feasible trajectory that accounts for the agent's kinematic constraints and performs immediate, low-level collision avoidance. Therefore, the specific problem is to design and implement an effective global planning algorithm that can work in concert with a given local planner to maximize the overall throughput of the simulated warehouse system.

III. Existing Methods and Their Limitations

The simulation's baseline global planners, Layered A* and RRT*, serve as performance benchmarks but reveal critical weaknesses in this multi-agent context.

Layered A* extends the standard A* algorithm into a space-time grid (x, y, time). While it can find time-optimal paths, it is a decoupled planner that calculates routes for agents sequentially. This often leads to suboptimal system performance and gridlock, as early agents claim paths that make it impossible for later agents to find a solution.

RRT* is a sampling-based planner effective in complex continuous spaces. However, it is non-deterministic and, like Layered A*, its sequential application in a dense multi-agent environment results in poor coordination and frequent planning failures. Also, the algorithm takes an extensive amount of time to search for a path, which is not ideal.

The practical failure of these methods is exposed by their interaction with local planners. When paired with a `VirtualForcePlanner` that only follows the general direction of a path, agents deviate and collide with obstacles and each other. Conversely, using a `DullPlanner` that strictly follows the path also fails, as it cannot resolve the unavoidable conflicts created by the decoupled global plan, again resulting in collisions. These issues highlight the fundamental need for a coupled planning approach that generates globally consistent, conflict-free paths.

IV. Proposed Method: Conflict-Based Search (CBS)

To address the limitations of traditional and decoupled planners, this project implements the Conflict-Based Search (CBS) algorithm, a modern and effective optimal solver for the MAPF problem. CBS is a two-level algorithm that cleverly decomposes the exponentially large multi-agent problem into a set of independent, single-agent searches, making it significantly more efficient in many scenarios.

The high-level search of the CBS algorithm, as detailed by Sharon et al. [1], operates on a constraint tree (CT). This is a binary tree where each node represents a set of constraints for the agents. The root node contains no constraints, and its solution is found by planning individual paths for each agent independently. The algorithm then proceeds by finding conflicts in the current best solution and resolving them by creating new child nodes with added constraints.

The algorithm operates on two distinct levels:

1) **High Level (The "Conflict" Level):** The high level performs a best-first search on a Constraint Tree (CT). Each node in this tree represents a set of constraints imposed on the agents and begins at a root node with zero constraints. When a node is evaluated, it checks the solution for conflicts. A conflict is defined as two agents occupying the same vertex at the same time (a vertex conflict) or swapping adjacent vertices at the same time (an edge conflict). If a conflict is found, the node is split into two children. Each child inherits the parent's constraints and adds one new constraint to resolve the conflict. For a conflict involving agents a_i and a_j at vertex v and time t, one child node will add the constraint (a_i, v, t) (prohibiting a_i from being at v at time t), and the other will add (a_j, v, t). This process continues until a node is found whose solution is conflict-free.
2) **Low Level (The "Search" Level):** The low level is responsible for finding paths for individual agents. For any given node in the high-level CT, the low-level search is invoked for each agent to find an optimal path from its start to its goal, subject to the set of constraints defined by that CT node. In our implementation, we used an A* search as our low-level search method. The collection of these individual paths forms the "solution" that is evaluated by the high level.

The core strength of CBS is that it avoids searching the massive combined state space of all agents. Instead, it only reasons about agents that are in direct conflict. This allows it

Algorithm 1 Conflict-Based Search (CBS)

```
Require: Set of agents $A = \{a_1, a_2, \dots, a_k\}$, graph $G$, start
    positions $S$, goal positions $T$
Ensure: Set of collision-free paths $\Pi = \{\pi_1, \pi_2, \dots, \pi_k\}$
 1: Initialize root node $N$ with empty constraint set
    $N.constraints = \emptyset$
 2: $N.solution \leftarrow$ find individual optimal paths for all agents
    ignoring other agents
 3: $N.cost \leftarrow$ sum of individual path costs in $N.solution$
 4: $OPEN \leftarrow \{N\}$
 5: while $OPEN \neq \emptyset$ do
 6:   $P \leftarrow$ node in $OPEN$ with lowest cost
 7:   Remove $P$ from $OPEN$
 8:   Validate the solution in $P$
 9:   $conflict \leftarrow$ first conflict in $P.solution$
10:   if $conflict =$ NULL then
11:     return $P.solution$ {Solution found}
12:   end if
13:   Let $conflict = \langle a_i, a_j, v, t \rangle$ {Agents $a_i, a_j$ conflict at
      vertex $v$ at time $t$}
14:   for each agent $a \in \{a_i, a_j\}$ do
15:     Create new node $A$
16:     $A.constraints \leftarrow P.constraints \cup \{\langle a, v, t \rangle\}$
17:     $A.solution \leftarrow P.solution$
18:     Update $A.solution[a]$ by replanning path for agent
        $a$ with constraints $A.constraints$
19:     if no valid path found for agent $a$ then
20:       Continue to next agent {Discard this branch}
21:     end if
22:     $A.cost \leftarrow$ sum of path costs in $A.solution$
23:     Add $A$ to $OPEN$
24:   end for
25: end while
26: return FAILURE {No solution exists} =0
```

TABLE I
PERFORMANCE COMPARISON OF PATHFINDING ALGORITHMS

Algorithm	Agents	PPM	Collisions	Exec. Time (s)
RRT*	5	N/A	N/A	N/A
Layered A*	5	378	2	366
CBS	**5**	**398**	**1**	**392**
RRT*	8	N/A	N/A	N/A
Layered A*	8	636	17	506
CBS	**8**	**640**	**5**	**552**

to quickly find optimal solutions, especially in environments with bottlenecks or corridors where only a few agents interact at any given time. By systematically resolving conflicts one by one, CBS is guaranteed to find a globally optimal solution.

V. RESULT COMPARISON

The performance of the implemented Conflict-Based Search (CBS) algorithm was evaluated against two other multi-agent pathfinding algorithms: RRT* and Layered A*. The experiments were conducted in two scenarios: a low-density environment with 5 agents and a medium-density environment with 8 agents, both operating with 10 loading and 10 unloading ports. Performance was measured against three Key Performance Indicators (KPIs): Packages Per Minute (PPM) to evaluate path efficiency, Collision Count to measure safety, and Execution Time to assess computational performance over a 3-minute simulation.

The results of these experiments are summarized in Table I.

A. Algorithm Performance Breakdown

RRT*: This algorithm proved to be unsuitable for this application. Its extensive search time meant that the simulation would trigger a re-plan before agents could reach their destinations. In cases where the search iteration limit was reached, the algorithm's fallback behavior—moving directly toward the goal—resulted in immediate collisions with obstacles and other agents. Consequently, it failed to produce a viable solution in either scenario.

Layered A*: This algorithm served as a strong baseline, demonstrating good performance in terms of throughput and execution speed. As shown in Table I, it achieved a PPM of 378 in the 5-agent test and 636 in the 8-agent test, with competitive execution times of 366 and 506 seconds, respectively. However, its primary weakness was safety. The algorithm recorded 2 agent-agent collisions in the low-density scenario, which escalated to 17 collisions in the medium-density scenario, indicating a significant degradation in path coordination as agent count increased.

Conflict-Based Search (CBS): The implemented CBS algorithm consistently outperformed the other methods in the most critical areas. In the 5-agent experiment, it increased PPM to 398 (+5.3% vs. Layered A*) and reduced collisions by 50% to just 1. In the more challenging 8-agent experiment, it delivered a PPM of 640 and, most notably, reduced the collision count by over 70% to just 5. This superior path quality and safety came at a modest computational cost, with execution times increasing to 392 seconds (+7.1%) and 552 seconds (+9.1%) for the two scenarios, respectively.

The results clearly demonstrate the effectiveness of the CBS algorithm for this multi-agent pathfinding problem. While Layered A* provides a fast solution, it lacks the sophisticated coordination needed to prevent collisions in environments with moderate agent density. In contrast, CBS successfully finds more efficient, conflict-free paths, leading to higher overall throughput (PPM) and a dramatic reduction in collisions. This improvement in solution quality is achieved with a slight, but acceptable, increase in computational execution time.

VI. DISCUSSION

The experimental results highlight the superiority of the Conflict-Based Search (CBS) algorithm for this multi-agent coordination task. The fundamental reason for its success lies in its decoupled, two-level architecture. Unlike monolithic planners that search in a massive, combined state space of all agents, CBS tackles the problem more intelligently. The low-level A* planner efficiently solves the simpler problem of finding an optimal path for a single agent, while the high-level search focuses only on resolving the specific points of contention—the conflicts.

This structure allows CBS to find globally coordinated, conflict-free solutions without the prohibitive computational cost of a fully centralized planner. In contrast, Layered A*'s sequential, priority-based planning cannot guarantee global optimality or collision avoidance; once an agent's path is set, it cannot be easily altered to accommodate lower-priority agents, leading to the high collision counts observed. RRT* failed because its random-sampling nature is ill-suited for finding precise, optimal paths in a constrained, grid-based environment, resulting in excessively long search times.

The CBS implementation successfully balanced path efficiency (PPM) and safety (Collision Count), delivering the highest throughput with the fewest collisions. The small number of remaining collisions can be attributed to implementation-specific factors rather than a flaw in the core algorithm, as discussed below.

VII. FUTURE WORK

While the current CBS implementation is effective, its performance can be further enhanced by addressing several limitations in the current system architecture.

1) **Implement Asynchronous, Agent-Specific Replanning:** The current framework triggers a complete replan for all agents whenever any single agent requires a new path. This is computationally redundant and scales poorly. Future work should focus on modifying the code structure to support asynchronous planning. In this model, the CBS algorithm would only be invoked for the specific agent that has completed its task, while the paths of all other agents remain fixed. This would dramatically reduce the computational load, especially as the number of agents increases.

2) **Enforce Strict Waypoint Adherence:** The current agent control system allows an agent to consider a waypoint ”reached” when it enters its general vicinity. This creates a mismatch between the precise space-time path calculated by CBS and the agent’s actual physical trajectory, leading to unforeseen conflicts. This can be resolved by tightening the agent’s control logic to require it to pass through the exact waypoint coordinates. This would ensure the agent’s physical movements perfectly align with the algorithm’s predictions, eliminating this class of collisions.
3) **Develop a Proactive, ”Just-in-Time” Conflict Resolution System:** The current system plans paths reactively. A more robust approach would be to implement a continuous, forward-looking conflict detection mechanism. Agents could constantly project their paths a few steps into the future. If this projection reveals an imminent collision that was not part of the original plan (e.g., due to minor delays or path deviations), the CBS algorithm could be triggered preemptively for only the involved agents. This would create a more dynamic and resilient system capable of resolving potential impacts right before they occur.

VIII. Conclusion

This paper presented an implementation and evaluation of the Conflict-Based Search (CBS) algorithm for solving the Multi-Agent Pathfinding problem in a simulated warehouse environment. Experimental results demonstrated that CBS significantly outperforms baseline methods like Layered A* and RRT*. It achieved higher overall throughput, measured in Packages Per Minute, while drastically reducing the number of collisions between agents. The success of CBS stems from its two-level approach, which efficiently finds globally optimal, conflict-free paths without the computational burden of traditional centralized planners. The findings confirm that CBS is a highly effective and robust solution for optimizing throughput in complex, multi-agent robotic systems. Future work will focus on architectural improvements to enable asynchronous replanning and more resilient conflict resolution, further enhancing the system’s scalability and performance.

References

[1] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, “Conflict-based search for optimal multi-agent path finding,” *Artificial Intelligence*, vol. 219, pp. 40–66, 2015.