Warebots: Simultaneous Agent Routing

Aman Chulawala, Aneesh Sinha, Pratik Tyagi

I. Abstract

This project tackles the challenging Multi-Agent Pathfinding (MAPF) problem, focusing on one-shot MAPF and its dynamic counterpart, Lifelong MAPF (LMAPF). LMAPF involves agents continuously adapting to new goals, measuring performance through throughput in dynamic environments. The study also explores Conflict-Based Search Algorithm (CBS), a two-level approach addressing conflicts between agents with a Conflict Tree (CT). CBS, applied to small-scale scenarios inspired by the League of Robot Runners competition, outperforms traditional methods like A* by examining fewer states while ensuring optimality.

II. Introduction

Multi-agent pathfinding (MAPF) poses a significant challenge as an NP-hard problem with practical applications in surveillance, search and rescue, and warehouses. Specifically, in one-shot MAPF, the objective is to determine collision-free paths for a team of agents from their initial positions to their respective goal positions, minimizing a predefined objective function like makespan (i.e., the time until all robots reach their targets) or the sum of their path lengths. Real-world scenarios, however, often involve dynamic challenges where agents must address a sequence of targets rather than remaining stationary after reaching the first one. Lifelong multi-agent pathfinding (LMAPF) is a variant of MAPF in which agents continually receive new goal locations and must reactively compute paths online. The performance of LMAPF is typically assessed in terms of throughput, representing the average number of targets reached per unit of time. In contrast to traditional one-shot MAPF, LMAPF, especially in factory-like environments, introduces additional complexities, requiring online algorithms capable of frequent replanning as goals change. The challenges intensify in densely populated, structured worlds typical of factory environments due to the increased number of conflicts between individual agent paths.

CBS (Conflict-Based Search) [4] is a two-level algorithm that diverges from converting the problem into a single 'joint agent' model. Instead, it adopts a distinctive approach where, at the high level, a search is conducted on a Conflict Tree (CT). This tree is structured based on conflicts between individual agents, with each node in the CT representing a set of constraints on the agents' motion. Simultaneously, at the low level, swift single-agent searches are executed to fulfill the constraints dictated by the high-level CT node. The unique two-level formulation of CBS often results in the examination of fewer states compared to A*, all while maintaining optimality in many cases.

In the duration of this project, we implemented a lifelong variant of the Conflict Based Search Algorithm for small scale environment. The problem statement for this project was adopted from the League of Robot Runners, an annual Multi Agent Path Finding Competition hosted by Amazon Robotics. Below we describe our approach, implementation and results.

III. Literature Review

The CBS algorithm, introduced by Sharon et al. in "Conflict-Based Search for Optimal Multiagent Path Finding," significantly advances the field of multi-agent pathfinding (MAPF). Utilizing a two-level Conflict Tree (CT) structure, CBS efficiently resolves conflicts between individual agents, outperforming traditional methods like A* by exploring fewer states while ensuring optimality. This work holds practical implications for applications such as surveillance, search and rescue, and warehouse logistics. Sven Koenig's influential contributions, including "Lifelong Planning A*" [2] have shaped the landscape of multi-agent systems and path planning, emphasizing efficiency and optimality in addressing challenges in lifelong planning scenarios through innovative heuristic search techniques.

The research on Multi-Agent Path Finding (MAPF) has witnessed significant advancements, with a particular focus on enhancing heuristics and algorithmic approaches. Li et al. (2019) contributed to this field by presenting "Improved Heuristics for Multi-Agent Path Finding with Conflict-Based Search. [3]" Their work, featured in the International Joint Conference on Artificial Intelligence (IJCAI), delves into refining heuristics within the Conflict-Based Search (CBS) framework. The authors addressed the intricacies of MAPF by proposing novel heuristics, aiming to further optimize the resolution of conflicts among agents. This study adds valuable insights to the ongoing discourse on efficient MAPF solutions.

Building upon this foundation, Boyarski et al. (2015) presented the "Improved Conflict-Based Search Algorithm for Multi-Agent Pathfinding

(ICBS)" [1] in the same IJCAI proceedings. Their research aimed at enhancing the Conflict-Based Search algorithm, a significant approach in addressing MAPF challenges. By introducing improvements to the ICBS algorithm, Boyarski et al. sought to optimize the resolution of conflicts, ultimately enhancing the overall efficiency of multi-agent pathfinding. This work contributes to the broader understanding of algorithmic strategies for MAPF, highlighting the continual evolution and refinement of conflict-based approaches.

Both studies, appearing in the prestigious IJCAI proceedings, underscore the importance of Conflict-Based Search in addressing the complexities of multi-agent pathfinding. Li et al. (2019) and Boyarski et al. (2015) have significantly advanced the field by proposing improved heuristics and algorithms, offering valuable contributions to the ongoing efforts aimed at developing more efficient and effective solutions for multi-agent pathfinding problems. These works collectively demonstrate the progressive nature of research in this domain and provide a foundation for further exploration and refinement of algorithms addressing the challenges inherent in multi-agent systems.

IV. Methodology

The algorithm we used to solve the life long multi-agent path finding problem is based on Conflict Based Search. In the images, we outline a strategy for Lifelong Multi-agent Conflict-Based Search (LCBS), designed for resolving the complex Lifelong Multi-Agent Pathfinding (LMAPF) problem, particularly in dynamic environments where agents receive continuous goal updates.

*a) **LifeLong Multi-agent Conflict-Based Search**:* This high-level algorithm maintains a list of paths and a boolean flag for each agent, indicating whether a path needs to be planned for them. The system runs in a loop for a specified simulation time, 'T_sim'. Within each iteration, if the 'run' flag is true, it calls the CBS function to plan paths for the agents. After planning, the 'run' flag is set to false, and all agents' planning flags are reset to false as well. The agents then execute actions based on the current paths. If an action leads an agent to its goal, the 'run' flag and the agent's planning flag are set to true, indicating that the paths need to be recalculated in the next iteration due to the changed environment. The time step increments, and the loop continues until the simulation time ends.

This iterative, dynamic approach allows agents to continuously adapt to new goals while managing and resolving conflicts. It is particularly suited to applications requiring high adaptiveness and reactivity, such as robotics in warehouse management, where agents represent robots or autonomous vehicles continuously navigating and fulfilling tasks. The Life-long CBS strategy ensures throughput and efficiency by minimizing conflicts and recalculating paths as the environment and goals change.

Algorithm 1 Life Long Multi-agent Conflict Based Search

```
Paths = ∅
list<bool> agents_to_plan = {True for all}
bool run ← true
while t ≤ T_sim do
    if run then
        Paths ← CBS(Paths, goal_list, agents_to_plan)
        run ← false
        agents_to_plan= {False for all}
    end if
    action ← Paths[t]
    for each agent a do
        if action leads to goal then
            run ← true
            agents_to_plan[a] = true
        end if
    end for
    t++
end while
```

```
1:  function CBS(Paths, goal_list, agents_to_plan)
2:      Root.solution = Paths[t,End]
3:      Update Root.solution using low_level(goal_list[t],agents_to_plan)
4:      Root.cost = Sum_of_cost(Root.solution)
5:      Root.constraints = ∅
6:      push Root to OPEN
7:      while OPEN is not empty do
8:          P ← best node from OPEN
9:          Check the paths in P for conflicts
10:         if P has no conflict then
11:             return P.Solution
12:         end if
13:         C ← first conflict(a_i, a_j, v, t) in P
14:         for each agent a in C do
15:             A ← new node
16:             A.constraints ← P.constraints + (a, v, t)
17:             A.solution ← P.solution
18:             Update A.solution
19:             A.cost = Sum_of_cost(A.solution)
20:             push A to OPEN
21:         end for
22:     end while
23: end function
```

*b) **CBS function**:* This function initializes with a given set of paths and a goal list for the agents to plan their routes. The root node solution is updated using low-level planning, which can be A* or Dijkstra. The low-level planner will assign initial paths to the agents based on their goals. The cost of the root solution is calculated by summing the costs of individual paths of all agents. The root node, with no initial constraints, is then pushed onto the OPEN list, which presumably contains nodes that represent possible solutions to be explored.

The algorithm then enters a loop, continuously selecting the best node from the OPEN list based on cost of nodes and checking for conflicts among the paths in the selected node. If no conflicts are detected, the solution is returned. Otherwise, the algorithm identifies the first conflict and for each agent involved, a new node is created with added constraints to avoid the conflict. The solution paths and cost are updated accordingly, and this new node is then pushed to the OPEN list for further

consideration. This highlights the binary nature of the algorithm. As conflicts involving two agents are resolved, each node expansion results in two child nodes.

We can in short summarize that CBS is an two level planning algorithm where on the higher level best first search is performed on a binary tree. Where each node of the binary tree contains solution, constraints and sum of cost of all path.

V. Experimental Results

A. Understanding the Environment

The two environments were utilized in this planning project offer distinct challenges and characteristics, contributing to a comprehensive evaluation of the algorithm's performance. The first environment is represented by a random map, characterized by its smaller size and a dynamic arrangement of obstacles scattered randomly across its dimensions. This scenario emulates a dynamic and unpredictable setting, where the algorithm must navigate through an ever-changing landscape.

In contrast, the second environment adopts the form of a warehouse map, a larger and more organized space with well-defined pathways and clear placements of objects. The controlled and structured nature of this environment allows for a focused investigation into the algorithm's efficacy when faced with a predefined and ordered layout. The absence of random obstacles and the presence of organized paths simulate a scenario akin to warehouse logistics, where efficiency and systematic movement of agents are paramount.

By juxtaposing these two environments, the project aims to explore the algorithm's adaptability to diverse and realistic planning scenarios. The random map introduces an element of uncertainty and unpredictability, while the warehouse map provides insights into the algorithm's ability to optimize paths within a structured environment. This dual approach enhances the project's robustness, ensuring a more nuanced understanding of the algorithm's strengths and limitations across different planning contexts.

B. Analysis and Insights:

The larger map with 33x57 cells and more agents (25) resulted in a higher total task completion (3666) but also required more computational time (72.722 seconds). The larger search space likely contributed to increased solve time and the need for a greater number of CBS calls. The smaller map with 32x32 cells and fewer agents (20) showed slightly lower total task completion (3416) but also had a lower solve time (70.821 seconds) and fewer CBS calls (2492). This suggests that the algorithm may perform more efficiently in smaller search spaces with fewer agents.

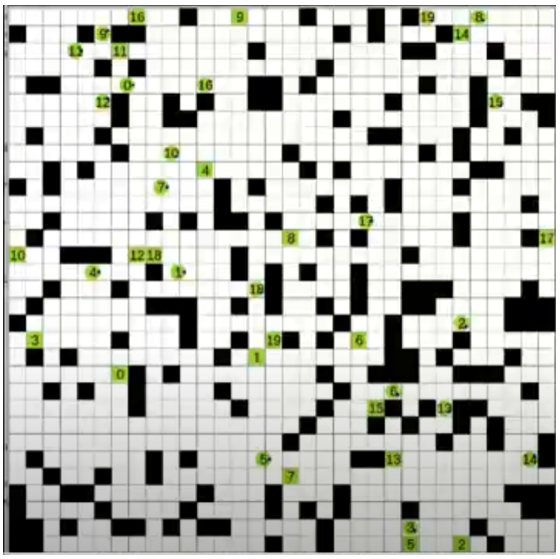

Fig. 1. Random Map

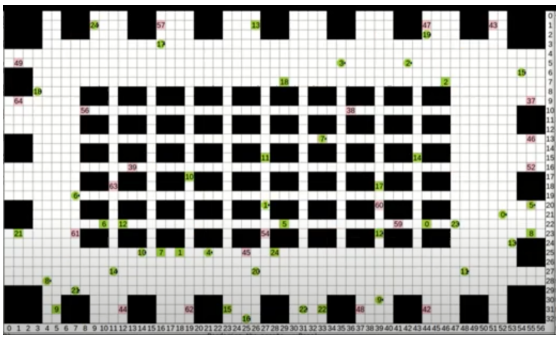

Fig. 2. Warehouse Map

The analysis of the algorithm's performance in the two distinct environments reveals compelling insights into its adaptability and efficiency. Notably, when navigating the warehouse map, characterized by a larger and organized layout with clear paths and structured object placements, the algorithm exhibits a noteworthy reduction in tree size. The streamlined nature of paths in the warehouse environment contributes to a more straightforward exploration of the search space, resulting in a smaller tree size during the planning process. This reduction can be attributed to the optimized and efficient nature of paths in organized environments, where the algorithm encounters fewer potential branching points and can identify optimal solutions more efficiently. The contrast between the smaller tree size in the warehouse map and the potentially larger tree size in the random map underscores the algorithm's ability to leverage the inherent structure of the environment, demonstrating its capacity to adapt to different planning scenarios. This finding further emphasizes the algorithm's versatility, showcasing

Metric	Random Map	Small Warehouse Map
Dimension	33 X 57 cells	32 X 32 cells
Number of Agents	20	25
Tasks done	3666	3416
Compute Time (sec)	72.722	70.821
σ in compute time (sec)	3	3
Average tree expansion	8.3 per CBS call	5.2 per CBS call
Total CBS calls	2625	2492

TABLE I
RESULTS

its effectiveness in scenarios characterized by clear and organized spatial configurations.

VI. ACKNOWLEDGEMENT

We would like to express sincere gratitude to the professor Maxim Likhachev and teaching assistants Abigail Breitfeld and Tejus Gupta whose guidance and support have been invaluable in completing this project. We would also like to acknowledge the organisers of the competition https://www.leagueofrobotrunners.org/ from whom we borrowed the simulation environment.

REFERENCES

[1] Eli Boyarski et al. "Icbs: The improved conflict-based search algorithm for multi-agent pathfinding". In: *Proceedings of the International Symposium on Combinatorial Search*. Vol. 6. 1. 2015, pp. 223–225.

[2] Sven Koenig, Maxim Likhachev, and David Furcy. "Lifelong planning A". In: *Artificial Intelligence* 155.1-2 (2004), pp. 93–146.

[3] Jiaoyang Li et al. "Improved heuristics for multi-agent path finding with conflict-based search: Preliminary results". In: *Proceedings of the International Symposium on Combinatorial Search*. Vol. 10. 1. 2019, pp. 182–183.

[4] Guni Sharon et al. "Conflict-based search for optimal multi-agent pathfinding". In: *Artificial Intelligence* 219 (2015), pp. 40–66.