BCBSD: Anytime Bounded Conflicted-Based algorithm for Dynamic Environments

Yue Yang[1], Jing Liang[2], Jia Pan[3]

***Abstract*— We present a noval algorithm for MAPF(multi-agent path finding) problem in real world scenarios. Considering the inaccuracies in perception and dynamic properties of real-world events, we use accurate decentralized perception to enhance global detection of obstacles. Based on anytime BCBS algorithm we develop the low level Focal Search to consider the dynamic obstacles and unpredictable events in real-world situations.**

I. Introduction

MAPF (multi-agent path finding) problem aims at finding paths for multiple agents from start positions to goal positions without conflicts [1]. MAPF problem is defined as a graph, $\mathscr{G}(V,E)$, with m agents denoted as a_1, a_2 ... a_m. The start and goal locations for each agent a_i is $s_i \in V$ and $g_i \in V$ respectively.

There are a wide variety of algorithms for solving the MAPF problem, including modified version of A*, CBS and its variants for different applications [2], [1], [3]. CBS (Conflict-Based Search) is a two-level algorithm, where the top level is used to find a path for each agent with the constraint tree (CT), and the low level is designed to resolve the conflicts caused by high level searches. The applications of MAPF are used in videos games, robotics and traffic control [3], [4]. In this paper, we propose a developed CBS algorithm to do path finding particularly in scenarios where there might be unpredictable obstacles.

In well known environment, where finding an optimal solution of MAPF problem is possible, it is NP-hard to find an optimal solution [5]. Specifically, for the CBS algorithm, the complexity of the constraint tree is exponential in the number of conflicts. To alleviate such issue, many sub-optimal algorithms are developed [6], [7], [8], [9], and some of them can scale to MAPF problem with more than 100 agents.

However in real-world scenarios, there might be many unpredictable events, and the time left for adjusting the current plan is limited and random [5], [10]. Therefore, the MAPF planning algorithm must also have the capability to always provide a valid adjusted solution even if it is interrupted due to the time limits before it achieves a high-quality plan. In other words, an interruptible or anytime MAPF planning is desirable in a dynamic scenario, which in this paper is modeled as a set of dynamic obstacles with the Gaussian movement uncertainty.

Besides the dynamic events, inaccuracy of detection may also effect path planning. We used decentralized detection to help centralized planning which can replan all robots which have probability to run into collision. And we also developed methods to satisfy anytime requirements in scenarios with dynamic and uncertain events. Our contributions include:

- We improve anytime BCBS(Bounded CBS) by using decentralized perception to help centralized multi-agent path planning.
- We propose a method to deal planning in scenarios with dynamic obstacles.

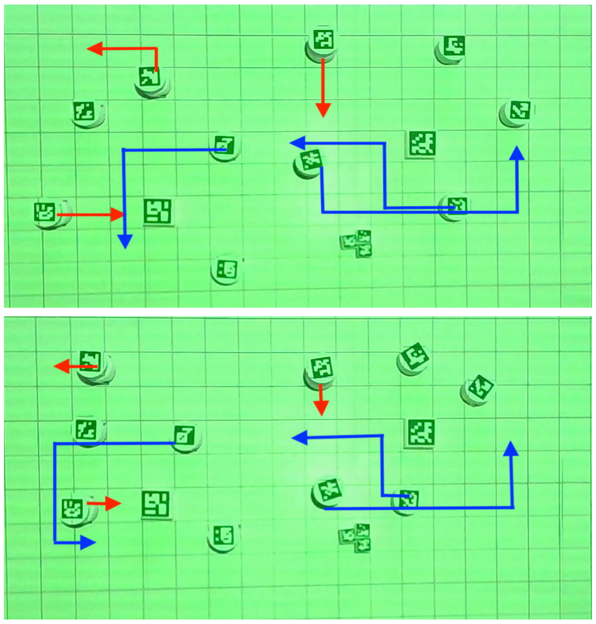
Fig. 1: In the figure, taller robots are dynamic obstacles, which central perceptions cannot get their positions accurately nor in real-time. Their motions are marked as red arrows. Small robots are agents we can control by our algorithm, with certain start and goal locations. Their calculated paths are marked as blue arrows. The rectangular boxes are static obstacles, which cannot move and their positions are exactly known by central server. As Our method is CBS based algorithm. We developed anytime BCBS to handle real world scenarios where there might be unpredictable events and the central perception suffered from inaccurate detection of dynamic obstacle. We use detection from decentralized robot to enhance the centralized perception for more accurate path planning.

II. Prior Work and Background

A. Optimality of Algorithm

For MAPF problem, there are unbounded solvers and bounded solvers to trade off between optimality and efficiency. Unbounded solvers include CA*, WHCA* [11],for MAPF problem. These algorithms aim at making searching faster. WHCA* based on Cooperative A* iteratively searches paths in dynamic widows with given window size. However

Authors 1, 2 are with WaterMirror (Shanghai) Technology Limited and author 3 is with University of Hong Kong

they cannot provide guarantees to the quality of returned path[12].

Bounded suboptimal methods can guarantee to find a solution that cost is no more than a given factor of optimal. Some methods are proposed for bounded search, Weighted CBS, Enhanced CBS [12] and inflated M*, where ECBS shows better result than others and it is also faster than others [12]. ECBS-HWY has a better performance than ECBS but still need users to provide highways to guide agents [13]. Based on the work of MACBS [14], ICBS (improved-CBS) [15] improves the searching algorithm using two methods, giving different priority to conflicts and also restarting search from scratch when new merge occurs. According to [4], ICBS has higher successful rate than M* and ICTS (increasing cost tree search) [3] when agent number is smaller than 55. But When agent number is bigger, ICBS becomes much slower and has lower successful rate than ICTS. Similar as CBS, ICTS also has two levels, where high level searches tree called increasing cost tree (ICT) in which each node stores cost vector per agent and low level gives goal test on the ICT's nodes. Inspired by Weighted A*, Barer et al. [12] proposed BCBS (Bounded CBS) using Focal list to bound the CBS algorithm and also ECBS (Enhanced CBS) to improve the efficiency of BCBS by increasing the flexibility in high level search. Based on the work of ECBS, Li et al. [16] proposed EECBS (Explicit Estimation CBS) by using online learning to estimate cost of solution and EES (Explicit Estimation Search) to help high level search. Boyarski et al. [2] proposed IDCBS which uses IDA* to substitute high-level search in CBS. However those algorithms do not take into account the dynamic properties of searching environment.

B. Uncertainty in Environment

For real world applications, uncertainty is an issue when we cannot accurately observe or predict exact locations or movement of obstacles. Atzmon et al. [5] provide algorithm pR-CBS, in which if the probability of conflict will not occur is larger than $p \in [0,1]$, the solution is called p-robust. We propose probabilitics deal with the uncertainty of dynamic obstacles.

Considering dynamic property of searching environment comes up with anytime problem. Anytime problem requires algorithms to find a solution with limited deliberation time. Some works are done by replanning using the algorithms which searching in static environments. Also algorithms such as D* Lite [17] or RRA* [11] do replanning only the part between agent's current location and goal. Likhachev et al. proposed ARA* (Anytime Repairing A*) [10] running A* with inflated heuristics, and they also tells that when scenario is dynamic and the dynamic events are not known at prior sub-optimal searching is still possible. Then DA* (Anytime Dynamic A*) [18] is proposed to decrease inflation factor and continuously improve the planing result within deliberation time. Based on these work, Cohen et al. [19] proposed anytime focal search method to improve the efficiency of planning. In this paper based on BCBS we propose a new algorithm dealing with dynamic properties of environment also considering anytime focal list to help searching.

III. Overview

We define the problem as MAPF problem, and here are some assumptions in addition to classical MAPF problem:

- There are static and also dynamic obstacles in the environment. The uncertain events may also happen in the environment.
- we assume movement of dynamic obstacle is fit in Guassian distribution with isotropic movement in each direction where there is open space.
- Centralized planning is used and agents all have accurate decentralized perception.
- Because of real world complexity, centralized perception cannot give accurate positions of dynamic obstacles.

Focal Search is used by BCBS. It contains OPEN list and FOCAL list. OPEN list is similar to A* and FOCAL list is the subset of OPEN list. As algorithm 1 FOCAL list is defined as:

$$FOCAL = \{n \in OPEN, f_c(n) \leq C\} \tag{1}$$

Where $f_c(n)$ is the cost of node n and C is the given bound of cost function. The minimum value of FOCAL list is $f_{min} = argmin_{n \in OPEN} f_c(n)$. BCBS uses Focal search in both high level search and also low level search. The high level Focal Search (g, h_c) is used to search conflict tree where g is the cost and h_c is the heuristic value. Low level Focal Search (f_a, h_c) solve the conflicts in the conflict tree, where $f_a = g + h_c$. As the function shows, the BCBS uses the cost function the same as A*, but in our approach the cost function is f_c which is mentioned in algorithm 1.

1 Algorithm: Low level of anytime BCBS for dynamic obstacles

```
Input: $n_{start}$, $isGoal(n)$, $succ(n)$, $w$
Output: $Solution$
 1: $OPEN = FOCAL = \{n_{start}\}$
 2: while $FOCAL \neq \emptyset$ do
 3:     $f_{min} \leftarrow f_c(head(OPEN), dangerPosList)$
 4:     $n \leftarrow head(FOCAL)$
 5:     $FOCAL \leftarrow FOCAL \setminus \{n\}$
 6:     $OPEN \leftarrow OPEN \setminus \{n\}$
 7:     if $isGoal(n)$ then return solution
 8:     for $each\ n' \in succ(n)$ do
 9:         $OPEN \leftarrow OPEN \cup \{n'\}$
10:         if $f_c(n') \leq \omega f_{min}$ then
11:             $FOCAL \leftarrow FOCAL \cup \{n'\}$
12:     if $OPEN \neq \emptyset\ and\ f_{min} \leq f_c(head(OPEN))$ then
13:         $updateLowerBound(\omega f_{min}, \omega f_c(head(OPEN)))$
14: function UPDATELOWERBOUND($B_o, B_n$)
15:     for each n in OPEN do
16:         if $(f_c(n) > B_o) \wedge (f_c(n) \leq B_n)$ then
17:             $FOCAL \leftarrow FOCAL \cup \{n\}$
18: function $f_c$(n) return $\alpha_1 N + \alpha_2 D + \alpha_3 f_a$
```

The goal of Focal Search is to find a sub-optimal solution bounded by C as fast as possible. Therefore nodes saved in FOCAL list and node expanded in each step should all be carefully chosen. The priority function $h_{pri}(n)$ is defined to select best node in each step to expand:

$$h_{pri}(n) = (C - g(n))/h_c(n) \tag{2}$$

Function 2 is also called potential function which is used by potential search. [19]. For nodes put to FOCAL list, in order to satisfy anytime requirements Cohen et al. [19] provide bound to costs of FOCAL list, and by adaptively update the bound in each iteration the algorithm, Anytime Focal Search, can converge quickly to a sub-optimal result. The function to choose the bounding of cost is as equation 4

$$C_i = S_{i-1} - \varepsilon \tag{3}$$

$$C_i = \omega_i f_{min} \tag{4}$$

Where i is the step time, ε is a small error, S_{i-1} is the last actual cost, and C_i is the bound to the cost function in the i_{th} time step. As algorithm 1 in line 10-11 we select the node which has lower cost value than weighted threshold C_i which is defined as equation 4

As algorithm 1 in line 12-13, in each time step we need to choose w_i to update lower bound of Low-level Focal Search. As line 14-17 the B_o and B_n represent old bound and new bound respectively.

In our approach we keep the method of choosing w_i same as Cohen et al.[19]. To make the algorithm an anytime planning we make w_i decrease while time step i increases. Further more, instead of recalculating all agents when conflicts happen, we only compute the agents involved in the conflicts to decrease the running time of this algorithm.

IV. Anytime Properties

We develop anytime BCBS by using decentralized perception to enhance the localization of dynamic obstacles to give more accurate positions of obstacles and real-time condition of current environment.

In real world environment, there might be uncertainty and inaccuracy in detection of obstacles. In our approach we update the centralized planning enhanced by decentralized perception. As Figure **??**, each agent has a range of detection $R > kd$ in each step, where d is the walking distance of the agent in deliberation time and $k \in [1,2,3...]$ which indicates how many steps the agent have to do replanning if unpredictable obstacle is detected. If agent detect obstacles which are not in previous planned environment, it will send the information to central server, and the agents whose paths are effected by the updated obstacles will need to re-plan.

When a new obstacle is detected, we treat it as an uncertain obstacle with changing position, then the map would be updated. In algorithm 2, L_o is the new detected obstacles, L_p is the list of all agents path, and the L_d is the dangerous points returned by this algorithm, P_a is the current position of each agent, and P_p is the position of node in current path. As line 6-11 we take into account the dangerous obstacles which are near the paths of each agents. specifically as line 9-12, if the path is away from the path or the obstacle is far from all the agents, then it won't be put in the list for planning.

Assuming dynamic obstacle would mostly keep the original speed, so we estimate the position of the new detected obstacle in next time step as a Gaussian distribution.

2 Algorithm: Detect dangerous points

```
Input: $L_o$, $L_p$
Output: $L_d$
 1: $L_d = \{\}$
 2: $THRESHOLD \leftarrow \lambda\sigma_p$
 3: for each agent do
 4:     $path \leftarrow L_p[agent]$
 5:     for $P_p \in path$ do
 6:         for each obstacle $\in L_o$ do
 7:             $distance_1 \leftarrow Distance(P_p, P_a)$
 8:             $distance_2 \leftarrow Distance(P_p, P_o)$
 9:             if $distance_1 >= distance_2$ then
10:                 $score \leftarrow Score(P_a, P_o, P_p)$
11:                 if $score >= THRESHOLD$ then
12:                     $L_d \leftarrow L_d \cup \{score\}$
    return $L_d$
```

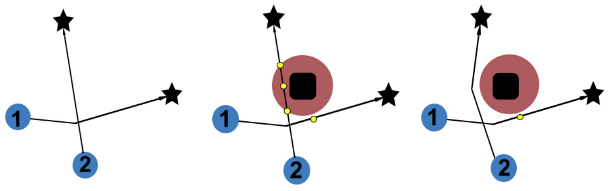

Fig. 2: The left figure shows the path before unpredictable obstacle appears. In the middle picture, when the obstacle is detected by robot 1, the position of this obstacle is updated to central server, and the yellow points are the critical points each path has nearest to the dangerous zone of obstacle. The path of robot 1 intersects with the dangerous range. After replanning of robot 1, it will get a new path which would keep the distance with obstacle.

For the agents which has paths near the obstacle, in algorithm 1 we denote the distance between current node, n, and the obstacle as D, and generate a new cost function f_c as in algorithm 1. As line 18-19, the cost function is defined as:

$$f_c = \alpha_1 N + \alpha_2 D + \alpha_3 f_a \tag{5}$$

Where N is the number of conflicts among agents, and f_a is the original cost function considering the cost-to-come and cost-to-goal. In this equation α_1, α_2 and $\alpha 3$ are weighted to balance the optimality and speed.

When doing path planning, we want the expanded node not to collide with the detected obstacle, $f_p > 0$ where f_p is the distance between an agent and the detected obstacle. Given position of the dynamic obstacle as a Gaussian distribution, $f_p \sim (\mu_p, \sigma_p)$. We set a confidence value λ to keep the path away form the detected obstacle.

Theorem 4.1: Given confidence factor λ, the probability of collision avoidance for each candidate point is bounded by $\frac{\lambda^2}{1+\lambda^2}$

Proof: The distance between obstacle and agent for planning is (μ_p, σ_p) and we have confidence factor λ.

We define the function Prob$(f_p > 0)$ as the probability when $f_p > \mu_p \pm \lambda\sigma_p$, according to Cantelli's inequality:

$$\text{Prob}(f_p > \mu_p + \lambda\sigma_p) \leq \frac{1}{1+\lambda^2} \tag{6}$$

$$\text{Prob}(f_p > \mu_p - \lambda\sigma_p) \geq \frac{\lambda^2}{1+\lambda^2} \tag{7}$$

For the confidence factor λ, we known the distance is non-negative $\mu_p \pm \lambda\sigma_p \geq 0$. Therefore the equation 7 is tighter than equation 6. ■

Theorem 4.1 give us an lower bound of agent to avoid collision with dynamic obstacle with the confidence of λ. To satisfy the sub-optimal purpose, we also need to give a bound to the distance based cost f_a.

Lemma 1.1: The optimal cost value, f_a in equation 5, is bounded by $H^* = g(w_i, \lambda, \alpha_1)$, where $\alpha_1 > 0$

Proof: Directly from the bound of Focal Search algorithm, we have:

$$f_c < w_i C^* \tag{8}$$

where w_i is an anytime factor chosen in each step, C^* is the optimal cost. We also have the lower bound of f_a according to Theorem 4.1, $\mu_p - \lambda\sigma_p$. Then we have:

$$\alpha_3 f_a = f_c - \alpha_1 N - \alpha_2 D \tag{9}$$

$$\alpha_3 f_a \leq w_i C^* - \alpha_2(\mu_p - \lambda\sigma_p) \tag{10}$$

$$f_a \leq \frac{1}{\alpha_1}(w_i C^* - \alpha_2(\mu_p - \lambda\sigma_p) \tag{11}$$

Here we define $H^* = \frac{1}{\alpha_3}(w_i C * - \alpha_2(\mu_p - \lambda\sigma_p)$, then we have $f_a \leq H^*$ ■

With the bounds given by Theorem 4.1 and Lemma 1.1 our approach can achieve some sub-optimal result for path planning.

As figure 2 shows, agent 1 whose path is below the confidence threshold, the algorithm would replan the robot. Figure 2 shows one step of replanning. When robots are running, the dynamic obstacle may also move, so the planning algorithm need to update each time robots detect any obstacle with unrecorded positions which threatens path of any agent.

References

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

[2] E. Boyarski, A. Felner, D. Harabor, P. J. Stuckey, L. Cohen, J. Li, and S. Koenig, "Iterative-deepening conflict-based search," in *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI)*, 2020, pp. 4084–4090.

[3] G. Sharon, R. Stern, M. Goldenberg, and A. Felner, "The increasing cost tree search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 195, pp. 470–495, 2013.

[4] A. Felner, R. Stern, S. E. Shimony, E. Boyarski, M. Goldenberg, G. Sharon, N. Sturtevant, G. Wagner, and P. Surynek, "Search-based optimal solvers for the multi-agent pathfinding problem: Summary and challenges," in *Tenth Annual Symposium on Combinatorial Search*, 2017.

[5] D. Atzmon, R. Stern, A. Felner, N. R. Sturtevant, and S. Koenig, "Probabilistic robust multi-agent path finding," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 30, 2020, pp. 29–37.

[6] G. Wagner and H. Choset, "Subdimensional expansion for multirobot path planning," *Artificial Intelligence*, vol. 219, pp. 1–24, 2015.

[7] A. Felner, J. Li, E. Boyarski, H. Ma, L. Cohen, T. S. Kumar, and S. Koenig, "Adding heuristics to conflict-based search for multi-agent path finding," in *Twenty-Eighth International Conference on Automated Planning and Scheduling*, 2018.

[8] E. Lam, P. Le Bodic, D. D. Harabor, and P. J. Stuckey, "Branch-and-cut-and-price for multi-agent pathfinding." in *IJCAI*, 2019, pp. 1289–1296.

[9] M. Barer, G. Sharon, R. Stern, and A. Felner, "Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem," in *Seventh Annual Symposium on Combinatorial Search*. Citeseer, 2014.

[10] M. Likhachev, G. J. Gordon, and S. Thrun, "Ara*: Anytime a* with provable bounds on sub-optimality," in *Advances in neural information processing systems*, 2004, pp. 767–774.

[11] D. Silver, "Cooperative pathfinding." *AIIDE*, vol. 1, pp. 117–122, 2005.

[12] M. Barer, G. Sharon, R. Stern, and A. Felner, "Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem," in *Seventh Annual Symposium on Combinatorial Search*. Citeseer, 2014.

[13] L. Cohen and S. Koenig, "Bounded suboptimal multi-agent path finding using highways." in *IJCAI*, 2016, pp. 3978–3979.

[14] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 219, pp. 40–66, 2015.

[15] E. Boyarski, A. Felner, R. Stern, G. Sharon, O. Betzalel, D. Tolpin, and E. Shimony, "Icbs: The improved conflict-based search algorithm for multi-agent pathfinding," in *Eighth annual symposium on combinatorial search*. Citeseer, 2015.

[16] J. Li, W. Ruml, and S. Koenig, "Eecbs: A bounded-suboptimal search for multi-agent path finding," *arXiv preprint arXiv:2010.01367*, 2020.

[17] S. Koenig and M. Likhachev, "Improved fast replanning for robot navigation in unknown terrain," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, vol. 1. IEEE, 2002, pp. 968–975.

[18] M. Likhachev, D. I. Ferguson, G. J. Gordon, A. Stentz, and S. Thrun, "Anytime dynamic a*: An anytime, replanning algorithm." in *ICAPS*, vol. 5, 2005, pp. 262–271.

[19] L. Cohen, M. Greco, H. Ma, C. Hernández, A. Felner, T. S. Kumar, and S. Koenig, "Anytime focal search with applications." in *IJCAI*, 2018, pp. 1434–1441.