Conflict-Based Model Predictive Control for Scalable Multi-Robot Motion Planning

Ardalan Tajbakhsh[1], Lorenz T. Biegler[2], and Aaron M. Johnson[1]

***Abstract*— This paper presents a scalable multi-robot motion planning algorithm called Conflict-Based Model Predictive Control (CB-MPC). Inspired by Conflict-Based Search (CBS), the planner leverages a similar high-level conflict tree to efficiently resolve robot-robot conflicts in the continuous space, while reasoning about each agent's kinematic and dynamic constraints and actuation limits using MPC as the low-level planner. We show that tracking high-level multi-robot plans with a vanilla MPC controller is insufficient, and results in unexpected collisions in tight navigation scenarios. Compared to other variations of multi-robot MPC like joint, prioritized, and distributed, we demonstrate that CB-MPC improves the executability and success rate, allows for closer robot-robot interactions, and reduces the computational cost significantly without compromising the solution quality across a variety of environments. Furthermore, we show that CB-MPC combined with a high-level path planner can effectively substitute computationally expensive full-horizon multi-robot kinodynamic planners.**

***Index Terms*— Multi-robot motion planning, model predictive control, collision avoidance**

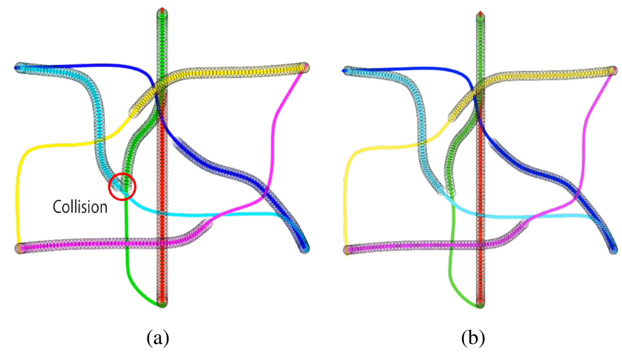


Fig. 1. (a) Tracking a conflict-based search (CBS) generated plan with a vanilla MPC controller does not guarantee collision-free execution due to tracking error and unaccounted for kinematic constraints. (b) CB-MPC allows for closer robot-robot interactions while remaining collision-free in execution.

I. Introduction

In order to truly unlock the potential of robots in real-world applications, they need to be deployed in numbers performing multiple tasks. These applications include picking and replenishment in warehouse fulfillment, environmental monitoring, coordinated search and rescue, material handling in hospitals, assembly operations in manufacturing, and more. Enabling such applications for multi-robot systems requires generating scalable and executable motion plans that operate in continuous time. In addition, these plans must respect the robot kinematics (or dynamics) and actuation limits, reason about robot-robot and robot-environment constraints, and be adaptable to changes in real-time.

Many multi-agent path finding (MAPF) algorithms have attempted to solve these problems [1–5]. Although scalable, they often resort to simplifying assumptions like ignoring the kinematics, actuation limits, and continuous states and actions. As shown in Fig. 1, this can result in infeasible execution when such plans are tracked with a controller under realistic conditions with tracking error, non-negligible turning radius, delays, and other execution imperfections. Some approaches like MAPF-POST [6] and action dependency graph [7] have attempted to resolve this problem by introducing an additional execution layer. However, they are not predictive, and often result in conservative execution. Other approaches resort to reactive controllers to prevent collisions [8–11]. These methods often generate aggressive control commands, and do not generalize well to tightly constrained environments with many robots. Multi-agent motion planning algorithms (MAMP) have also attempted to address this problem by generating dynamically-feasible motion plans for multiple robots while avoiding collisions and obstacles [12–17]. These methods often suffer from long solve times and/or poor solution quality. This is a major shortcoming as real-time changes in system states require an effective planner to continuously update the plans for some or all robots. To address the adaptability problem, model predictive control (MPC) based approaches generate receding-horizon trajectories that respect kinematic, dynamic, and actuation limits, and avoid collisions and obstacles [18–20]. Although effective, these approaches do not scale well to large number of robots or tightly constrained environments due to the presence of many non-convex collision and obstacle constraints.

Our approach differs from other multi-robot motion planning algorithms in that:

- It does not include constraints from all other agents at every timestep in each optimization solve as it uses a conflict tree to resolve conflicts collaboratively.
- The collisions with other robots and obstacles are efficiently resolved as constraints instead of an additional term in the cost function. This improves feasibility, solve time, and guarantees constraint satisfaction.

We introduce a two-level motion planning algorithm called conflict-based MPC (CB-MPC), which leverages the conflict resolution mechanism of CBS combined with MPC as the

[1] Department of Mechanical Engineering and [2] Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA, USA, (atajbakh, lb01, amj1)@andrew.cmu.edu

low-level planner to generate collision-free and executable motion plans for multiple robots simultaneously. These plans respect the physical limits of the system, leverage the receding horizon property of MPC to speed up the constrained optimization by seeding with a prior solution, and resolve conflicts between agents efficiently through a CBS-like conflict tree. The proposed algorithm is implemented in a centralized fashion, however it can also be deployed in a decentralized manner with full communication between agents for practical purposes.

This paper aims to validate the following hypotheses:

1) Using CB-MPC as the local planner for real-time receding horizon multi-robot motion planning results in a higher success rate than tracking a CBS plan with vanilla MPC.
2) Compared to other MPC formulations (joint, prioritized, and distributed), CB-MPC is more computationally efficient and has a higher success rate across a variety of environments.
3) Using CB-MPC, larger scale multi-robot motion planning problems can be solved in a receding horizon fashion that would otherwise take significantly longer to solve with one-shot MAMP algorithms with a similar solution quality.

The remainder of this paper is organized as follows. Sec. II summarizes the related works. Sec. III describes the preliminaries. Sec. IV presents a detailed description of the CB-MPC algorithm. Sec. V describes the experiments and Sec. VI presents results in a variety of environments. Sec. VII concludes the paper and notes future work.

II. Related works

There have been many works on addressing the multi-agent path finding (MAPF) problem. The most notable example among centralized methods are conflict based search (CBS) [1] and its variants [2–5], which leverage a high-level conflict tree to resolve robot-robot conflicts without planning over the joint state space of all robots. Another promising approach is M^* [21], which plans for each robot independently and only plans jointly for the robots that interact with each other using sub-dimensional expansion. Although scalable, these methods often require an additional execution layer like MAPF-POST [6] and action dependency graphs [7] as they do not consider robot kinematics, actuation limits, or execution delays and they rely on discrete states and actions. Some methods resort to local reactive controllers to prevent imminent collisions during execution [8–11]. However, they are not predictive and are prone to collisions or deadlocks in constrained environments with many agents. In [22], a receding horizon version of MAPF is introduced that enables continuous re-planning for all agents in dynamic environments. However, similar to other MAPF algorithms, this approach does not account for robot kinematics and actuation limits. Another group of methods resort to assigning priorities to agents and enforce lower priority agents to avoid the trajectories of higher priority counterparts [23–26]. This approach has been shown to perform well in environments with fewer constraints, but does not work well in highly constrained environments with many robot-robot interactions. Furthermore, it is unclear how agent priorities must be assigned in practical scenarios without sacrificing the solution quality.

In contrast, multi-agent motion planning (MAMP) algorithms aim to generate dynamically feasible plans that robots can execute while satisfying robot-robot and robot-environment constraints. Among sampling-based methods, [12–15] adapt RRT and RRT* motion planners to work for multi-robot cases on discrete graphs and geometrically embedded composite roadmaps. These have been shown to outperform M* and other variants of RRT that plan over the composite state space of all agents. Recently, [14] demonstrated scalable and dynamic multi-robot motion planning by leveraging the structure of CBS with a modified version of RRT to plan conflict free motion plans. Although these methods have been shown to be effective for problems with many robots and variety of environments, their run-time does not allow for fast online re-planning. This can be a major drawback for hardware deployment as imperfections in execution and changes in the environment would often necessitate adapting the motion plans of some or all agents.

Trajectory optimization has also been used for trajectory generation for multiple robots. Some methods in this category leverage sequential constraint tightening [16,17], since the full problem with all the constraints is often intractable. They have been shown to be effective for up to 10 robots in constrained environments. However, they are not suitable for real-time applications due to their large solve times. Other methods like [18–20] use model predictive control (MPC) for online trajectory generation in multi-robot systems. In these methods, robots share their predicted MPC trajectories with the rest of the fleet and conflicts are resolved by constraining them from occupying the same locations at the same times. These methods require inter-robot collision constraints from all robots to be added at each timestep instead of resolving them collaboratively, resulting in higher solve times. In addition, they can cause deadlocks (especially, in problems with symmetry) due to decentralized collision resolution. Our approach ensures MAPF plans are executable by introducing an efficient MPC local planner that scales well with the number of robot-robot interactions through efficiently splitting constraints between the involved agents.

III. Preliminaries

A. Problem Formulation

Consider a system consisting of a set N_a of second order discrete time systems

$$x_i^{k+1} = f(x_i^k, u_i^k), \quad \forall i \in N_a \tag{1}$$

where $x_i^k = [p_i^{k,T}, v_i^{k,T}]^T \in X_i \subseteq R^{2n}$ and $u_i^k \in U_i \subseteq R^n$ are the ith agent's states and inputs, respectively, at time k; while $p_i^k \in R^n$ and $v_i^k \in R^n$ denote agent i's position and velocity, respectively. Note that we assume for simplicity that each agent has identical dynamics f, though that is not a requirement of this algorithm.

The control objective is to regulate all agents to a desired pose, x_i^d, while minimizing control effort and avoiding collisions with other robots and obstacles. This can be formally written as the following for every time k:

$$\lim_{k\to\infty} \|x_i^d - x_i^k\| \leq \epsilon_g \qquad \forall i \in N_a \tag{2}$$

$$\|p_j^k - p_i^k\| \geq D + \epsilon_r, \qquad \forall i, j \in N_a \tag{3}$$

$$\|p_p^o - p_i^k\| \geq \frac{D}{2} + \epsilon_o, \qquad \forall i \in N_a, p \in N_o \tag{4}$$

where, ϵ_g denotes the goal tolerance, D represents the robot footprint diameter, p_p^o denotes the position of the static obstacle p, N_o represents the set of static obstacles in the environment, and ϵ_r and ϵ_o denote the safety margins for robot and obstacle constraints, respectively. Note that we only consider circular and polyhedral obstacles in the presented experiments, but maintain that most irregular shaped obstacles can be safely approximated with collections of these two primitives. Other robots are treated as dynamic obstacles, where their current positions and predicted future trajectories are updated at each timestep.

B. *Conflict Based Search (CBS)*

CBS is a centralized, complete, and optimal multi-agent path finding algorithm that relies on collaborative conflict resolution to efficiently generate conflict-free paths for multiple agents simultaneously [1]. In particular, CBS uses a low-level path planner (for example A^* [27]) to plan single-robot plans. It then checks those generated plans for vertex or edge conflicts (i.e. robots occupying same locations at the same time, or traversing the same edge at the same time). Conflicts are stored in a constraint tree and resolved by invoking the low-level planner with the additional constraints. This process is repeated until a conflict-free plan is found for all the agents. The key idea with CBS is splitting constraints between the agents and adding them to the problem as needed, instead of solving the problem over the joint state spaces of all agents or including all the constraints at once. This property allows for better scalability as the problem size grows with the number of conflicts and not the number of agents. We refer the reader to [1] for more details on the CBS algorithm and results. We borrow a similar conflict resolution mechanism from CBS with modifications to the conflict definition (defined in IV-A) and the application of those conflicts as constraints in receding horizon planning.

C. *Model Predictive Control Planner (MPC)*

MPC is a receding horizon optimization framework that is commonly used for solving constrained optimal control problems. The optimization problem in MPC is often solved under dynamic, state, and input constraints to ensure solutions can be closely executed on hardware. In the case of MAMP, additional nonlinear constraints from other robots and obstacles are present, which make the optimization non-convex. However, most modern non-linear optimization solvers can effectively solve these problems using either sequential quadratic programming (SQP) or interior point (IP) methods. The MPC problem for each robot is formulated as follows:

$$\begin{aligned}\operatorname{argmin}_{u_i, x_i} J_i = \sum_{l=k}^{k+N-1} &((x_i^l - r_i^l)^T Q (x_i^l - r_i^l) \\ &+ u_i^{l,T} R u_i^l) + x_i^{(k+N),T} P x_i^{k+N}\end{aligned} \tag{5}$$

subject to (1), (3), (4),

$$x_i^k \in X_i \tag{6}$$

$$x_i^{k+N} \in X_f \tag{7}$$

$$x_i^l \in X_{feasible} \tag{8}$$

$$u_i^l \in U_{feasible} \tag{9}$$

In this formulation, $u_i = [u_i^k, u_i^{k+1}, ..., u_i^{k+N-1}]$, $x_i = [x_i^k, x_i^{k+1}, ..., x_i^{k+N}]$, and $r_i = [r_i^k, r_k^{k+1}, ..., r_i^{k+N}]$ denote the sequence of control inputs, states, and reference states over the planning horizon N for the ith robot. X_i and X_f denote the set of initial and final conditions for all agents. $X_{feasible}$ and $U_{feasible}$ represent the set of feasible states and control inputs. The objective function (5) penalizes reference tracking error, control input magnitude, and terminal cost. Q, R, and P are positive semi-definite tuning matrices. (1), (3), and (4) represent the kinematic (or dynamic) feasibility, inter-robot collision, and obstacle constraints respectively. (6) enforces the initial conditions for all agents. (7) represents the terminal constraint. (8) and (9) ensure state and control feasibility. The reference trajectory can come from any single or multi-robot path planner. In this paper, CBS is used as the high-level path planner, since it reasons about conflicts between agents and acts as an effective heuristic for the MPC local planner.

IV. Conflict-Based Model Predictive Control (CB-MPC)

A. *Conflicts in CB-MPC*

As shown in Fig. 2, for a given planning horizon t_h, conflicts can arise between pairs of agents, or agents and obstacles. For agent-agent conflicts between agents a_i and a_j, the conflict are defined as a tuple (a_i, a_j, [t_c, t_h]), where t_c denotes the predicted time of collision at which (3) is violated. Agent-obstacle conflicts are defined similarly for a given agent as a tuple (a_i, p_p^o, [t_c, t_h]), where (4) is violated. This differs from CBS in that the conflicts in CB-MPC are defined over location-time ranges instead of location-time pairs to support the continuous-time requirement of MPC. In addition, the conflict definition in CB-MPC allows it to generate a single constraint set between each pair of agents at each timestep.

B. *CB-MPC Algorithm*

CB-MPC combines the benefits of MPC and CBS in a unified motion planning framework to allow for receding horizon multi-robot motion planning. At each timestep, the algorithm summarized in Algorithm 1, takes as input the tuple $M = (N_o, X_i, X_f, N)$ that summarizes all the problem specific variables. Each robot plans its motion for a given

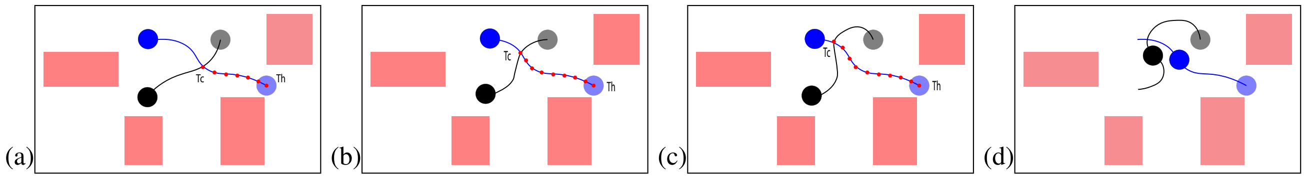


Fig. 2. (a) Agents black and blue are tasked to go from their start poses shown in solid colors) to their goal poses (shown in transparent colors). For a given planning horizon t_h, the initial predicted MPC trajectory of the agents has a collision at t_c. Collision constraints are added at each timestep between [t_c, t_h] (shown in red dots) for the black agent since it has a lower total cost. (b,c) The updated MPC solution resolves the prior constraints, but results in a new collision. (d) Collision-free trajectories are generated after collisions are resolved iteratively.

planning horizon using MPC with all the constraints excluding (3) and (4) (lines 2-4 in Algorithm 1). The resulting MPC trajectories and the total cost are stored in $R.solution$ and $R.cost$, and the node is added to an open list that is sorted by cost in an increasing order (lines 5-7).

The node cost J_n computed by the SIC operator is defined as the sum of the current trajectory length J_c, and the cost-to-go J_f, for all agents. J_f in our case is the Euclidean distance from the final position of the horizon to the goal. Note that in case of a heterogeneous fleet, where agents have different dynamic capabilities (e.g. drones and ground vehicles), this cost can also include the agent's velocity and acceleration to prioritize conflicts more intelligently.

The lowest cost node from the open list is retrieved and removed (line 10). The trajectories stored in this node are checked for collisions for the involved pair of agents at each timestep in the planning horizon. If these trajectories are collision-free with respect to (3) and (4), the solution is returned (lines 12-14). The first input is then applied for each robot, and the state is propagated based on the described dynamics. Otherwise, if one or multiple conflicts are detected, a new node is created for each agent involved in every conflict. For a given planning horizon t_h, the collision constraint is then applied at each time between [t_c, t_h] (refer to Fig. 2).

The application of constraints for all timesteps after t_c is done because constraining a single timestep will likely shift the collision to the immediate future in the receding horizon setting. Furthermore, in most practical scenarios after the initial condition, collisions will appear from the end of the horizon, which results in most optimization solves containing collision constraints for a small subset of the planning horizon. Each node is then added to the open list. For every collision, the agent with the lowest node cost attempts to resolve the conflict first by solving the updated MPC problem with the added constraints from the other agent involved. If the collision is not resolved, the other agent would do the same. This process continues for every conflict until all of them are resolved at a given timestep (lines 15-22).

The algorithm as defined requires synchronous communication among agents. This requirement can be easily relaxed due to the predictive nature of the planner. Not every robot needs to wait until the trajectory of all other robots are received to generate its plan as it can rely on the last predicted plan received from other agents even if it is delayed up to the duration of the prediction horizon. In such case, the safety margin ϵ_r can be increased proportional to the time elapsed since the last received communication, similar to [28].

Input: M
Output: Collision-free trajectories $P.solution$
1 $R \leftarrow$ new Node
2 **for** *each agent* a_i **do**
3 | $R.solution(a_i) = MPC(M, a_i, \{\})$
4 **end**
5 $R.cost = SIC(R.solution)$ ▷ (Sum of individual costs)
6 O = {} ▷ (Initialize open list)
7 Insert R in O
8 **while** *not all conflicts resolved:* **do**
9 | $P \leftarrow$ lowest cost node from O
10 | check P for conflicts
11 | **if** *P has no conflicts:* **then**
12 | | **return** $P.solution$
13 | **end**
14 | $C \leftarrow$ first conflict (a_i, a_j, $[t_{c_1}, t_{c_2}]$)
15 | **for** *each agent a_i in C* **do**
16 | | $A \leftarrow$ new Node
17 | | A.constraints = $P.constraints$ + $(a_i, a_j, [t_{c_1}, t_{c_2}])$
18 | | $A.solution = MPC(M, a_i, A.constraints)$
19 | | $A.cost = SIC(A.solution)$
20 | | Insert A to O
21 | **end**
22 **end**

Algorithm 1: Conflict-based MPC

V. Experiments

In this section, a series of experiments are presented to validate the efficacy of the CB-MPC algorithm and compare its performance against vanilla, joint, prioritized (Pr-MPC), and distributed (D-MPC) methods for different number of robots and a variety of environments.

A. Baseline Comparisons

The joint MPC refers to the centralized problem, which is structurally similar to [29]. In this case, the optimization is solved over the joint state space of all robots, and all obstacle and collision constraints are present in the problem prior to the execution. Pr-MPC is implemented similar to [26] with random priority assignment between the agents. In this case, the lower priority robots must avoid the predicted trajectory of the higher priority robots. D-MPC is another variation of multi-robot MPC [19], which detects conflicts on-demand similar to CB-MPC. However, conflicts are resolved by all the agents involved and not collaboratively due to the absence of a central coordinator. Note that for all the baseline comparisons, we only implement the core MPC algorithms. In particular, we do not perform any additional convexification of the collision constraints, or introduce a special high-level planner to keep the comparisons uniform.

B. Experimental Setup

Experiments are performed across four different environments: a narrow environment, an open environment, and a randomized cluttered environment. All experiments are implemented in MATLAB and run on a 6-core Intel core i7 @ 2.6 GHz with 16 GB of RAM. The optimization problems are modeled in YALMIP [30] and solved with IPOPT [31], a state-of-the art interior-point solver for non-convex optimization. Table I summarizes the default hyper-parameters used throughout the experiments.

TABLE I

SUMMARY OF HYPER-PARAMETERS USED IN THE EXPERIMENTS

Quantity	Description	Value
Q	Reference tracking penalty	5
R	Control effort penalty	1
P	Terminal penalty	40
N	MPC horizon	20
D	Robot footprint [m]	0.3
ϵ_g	Goal tolerance [m]	0.2
ϵ_r	Robot-robot collision tolerance [m]	0.05
ϵ_o	Robot-obstacle collision tolerance [m]	0.05

A trial is considered successful if all robots reach their goal state without any collisions. If an optimization solve returns an infeasible solution or any of the robots get stuck in a deadlock, we terminate that trial and mark it as a failure. The deadlock detection relies on tracking the distance to goal and the average velocity for each agent in the past 20 steps of the execution. If the reduction in distance to goal has been less than 0.1 and the average velocity has been less than 0.1, we flag that trial as a deadlock failure. This enables us to effectively terminate unsuccessful runs when running randomized batch simulations.

We compare makespan (i.e. sum of trajectory lengths of all robots), average and max solve times per timestep per robot (T_{avg} and T_{max}), and average number of constraints added per timestep per robot (C_{avg}) for each scenario. T_{avg} and T_{max} for CB-MPC, Pr-MPC, and D-MPC are summed for all robots at each timestep only when compared with the joint MPC. This is because joint MPC solves a single optimization at each step, while other algorithms are distributed. In addition, C_{avg} is computed as the average total number of constraints added to solve a given constrained optimization solve. For CB-MPC, this value is computed from the last iterative solve that resolved all detected collisions. Finally, note that for the randomized trials, these quantities are only computed for the successful runs of each algorithm to remove any bias from the results.

C. System Definition

We use a double integrator discrete-time model to demonstrate the effectiveness of the CB-MPC algorithm. The states and control inputs are defined as $z = [x, y, \dot{x}, \dot{y}]$ and $u = [\ddot{x}, \ddot{y}]$ respectively. The system dynamics are as follows:

$$x^{k+1} = Ax^k + Bu^k \tag{10}$$

where,

$$A = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, B = \begin{bmatrix} \frac{1}{2}\Delta t^2 & 0 \\ 0 & \frac{1}{2}\Delta t^2 \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix}. \tag{11}$$

where $\Delta t = 0.05$ is the discretization step size.

VI. RESULTS

A. Narrow Environment

The narrow environment, as shown in Fig. 3, is a challenging benchmark problem that requires two robots to swap positions through a narrow corridor in presence of tight obstacle and robot-robot collision constraints. The black and blue agents trade positions between (3,3) and (-3,-3), with zero initial and final velocity.

As shown in Table II, only Joint-MPC and CB-MPC are able to complete the task with $N = 20$. It is interesting to note that CB-MPC is able to do so with a similar makespan at approximately a quarter of the T_{avg} of the joint MPC. This is due to CB-MPC solving smaller optimization problems with fewer robot-robot constraints as demonstrated by the active constraints for the entire trajectory in red. The computational cost of the joint MPC comes with the advantage of better feasibility. This is enabled by the additional degrees of freedom in the optimization. In particular, at N = 10, only joint MPC finds a feasible solution. From this observation, we can conclude that the constraints of this problem require both robots to simultaneously modify their trajectories with shorter planning horizons. Comparing CB-MPC with other MPC variations, note that the collaborative conflict splitting mechanism enables CB-MPC to generate feasible motions through the narrow corridor where other approaches become infeasible or get stuck in a deadlock. This supports hypothesis 2, since CB-MPC outperforms other multi-robot MPC variations in this challenging environment.

B. Open Environment

The open environment demonstrates the worst-case scenario in terms of the number of robot-robot interactions. As shown in Fig. 4, four robots are tasked to switch positions with the robot across from them, while avoiding all other robots. The robots start at (0,-2), (0,2), (-2,0), and (2,0), with zero initial and final velocity. All algorithms track a CBS plan as the reference trajectory. Due to the difficulty of this scenario, the MPC planning horizon was set to $N = 60$,

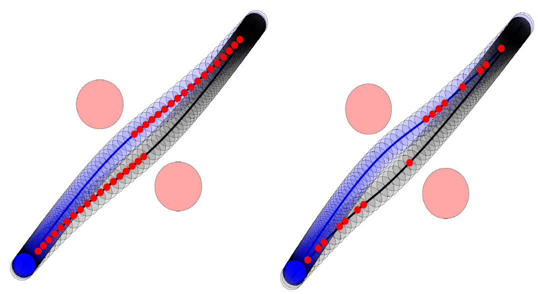


Fig. 3. Comparison between the Joint MPC (left) and CB-MPC (right) in the narrow environment. CB-MPC is able to get similar quality of solution with the makespan of 8.36 compared to 8.45 of the joint MPC at a significantly lower computation cost. Red dots denote the active constraints.

TABLE II

BENCHMARK RESULTS FOR THE NARROW ENVIRONMENT

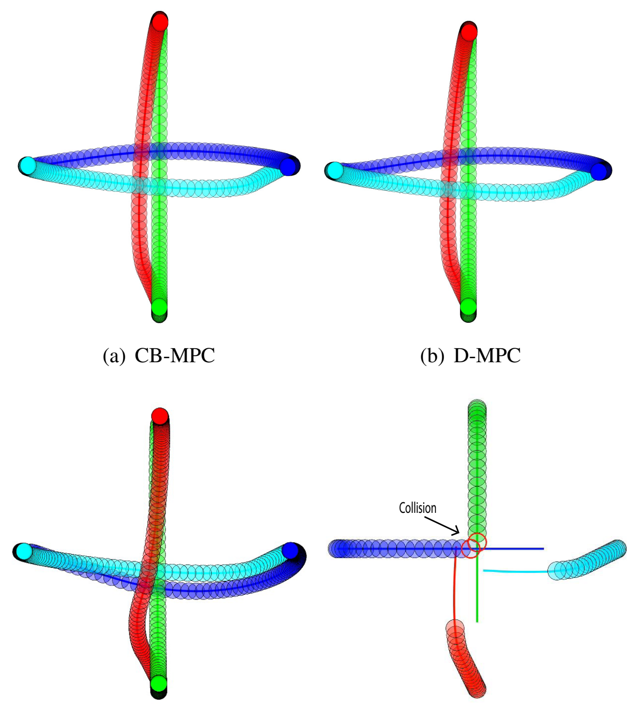
	N	Makespan	T_{avg}	T_{max}	C_{avg}
Joint-MPC	10	7.86	0.09	0.32	20
CB-MPC	10	-	-	-	-
Pr-MPC	10	-	-	-	-
D-MPC	10	-	-	-	-
Joint-MPC	20	8.45	0.38	1.10	40
CB-MPC	20	8.36	0.08	0.74	1.35
Pr-MPC	20	-	-	-	-
D-MPC	20	-	-	-	-

which is equivalent to a 3-second look-ahead, or about 15% of the time to execute the maneuver.

As shown in Fig. 4(d), vanilla MPC results in collision during execution, despite tracking the collision-free CBS plan. This is due to the kinematics and actuation limits of the robots that are not accounted for, and resolving the conflicts in discrete-time under the assumption of ideal execution. As a result, slight mismatches in agents' timing tracking the reference waypoints can result in unsafe behaviour. Comparing other MPC variations, CB-MPC performs slightly better than D-MPC and about an order of magnitude better than Pr-MPC in terms of T_{avg} and T_{max} (refer to Table III). It is important to note that the reduction in computation cost does not compromise the solution quality (i.e. makespan). This mainly results from splitting the non-convex collision constraints between the robots effectively and distributing the computation load among them, as evidenced by the C_{avg} comparison. As demonstrated in Fig. 4(c), Pr-MPC penalizes lower priority robots significantly more, since they have to avoid the predicted trajectories of all higher priority robots (note that red and dark blue robots have lower priorities in this case). This results in lower priority robots solving much harder optimization problems, resulting in delays updating the control action and higher probability of infeasible solves.

To summarize, we demonstrated that CB-MPC outperforms other multi-robot MPC variations on this worst case scenario (hypothesis 2). Furthermore, we showed that CB-MPC guarantees collision-free execution as opposed to tracking a MAPF plan with vanilla MPC (hypothesis 1). Finally, we provided evidence that CB-MPC can be integrated with any high-level MAPF planner (CBS in this case), which provides higher level conflict resolution, while ensuring safe execution under realistic conditions (hypothesis 3).

It is important to note that, similar to other flavors of MPC, the results presented in this section are dependent on the specific choice of the hyper-parameters. In particular, a planning scenario may become feasible/infeasible or result in deadlocks if the planning horizon N is shortened. Similarly, the number of predicted collisions may vary as a result of trajectory tracking error for different choices of Q, R, and P. For this particular scenario, the problem becomes infeasible at any planning horizon $N < 60$ steps.



(a) CB-MPC (b) D-MPC

(c) Pr-MPC (d) MPC

Fig. 4. Four robots swapping positions with tight interactions (open environment). Vanilla MPC fails due to collision during execution (d). CB-MPC and D-MPC provide similar solutions (a and b). Pr-MPC penalizes lower priority robots (red has the lowest priority), but completes the task

TABLE III

BENCHMARK RESULTS FOR THE OPEN ENVIRONMENT

	N	Makespan	T_{avg}	T_{max}	C_{avg}
CB-MPC	60	17.33	0.06	0.07	8.86
D-MPC	60	17.26	0.08	0.10	15.20
Pr-MPC	60	17.68	0.45	1.01	22.50
MPC	60	-	-	-	-

C. Cluttered Environment

The cluttered environment presents randomized trials for a varying number of robots. We exclude the vanilla MPC and the joint MPC from these trials due to the fundamental issues of executability and scalability discussed previously. Each environment is a 5x5 map with 6 randomly placed circular obstacles, where the diameter is sampled randomly from the range 0.4–0.8. This is to done to add variation in the map and obstacle density. The start and end positions of robots are also randomly selected for each scenario. We consider these smaller and more constrained maps to demonstrate the value of CB-MPC in tight navigation scenarios. Scaling to larger maps would depend on the scalability of the high-level MAPF planner, which is not in the scope of this paper. To ensure the generated problems are feasible, each scenario must satisfy the following conditions:

1) The start and goal positions of each agent are unique and must not intersect with another agent.
2) The start and goal positions of each agent must not lie inside any of the obstacles.
3) There is enough space between every pair of obstacles for at least two robots to navigate without colliding.

CB-MPC, D-MPC, and Pr-MPC are tested with 2, 3, and 4 robots. For each scenario, 5 randomized trials are run. This results in 45 total experiments. Note that we do not provide any MAPF reference plan to the MPC to increase the probability of robot-robot interactions.

As shown in Fig. 6(a), the success rate of Pr-MPC drops significantly as the number of robots increases, while CB-MPC maintains a 100% success rate across all the trials. D-MPC performs well for 2 and 3 robot trials, but its' success rate degrades in the 4 robot case. The advantage of CB-MPC in solving harder problems is mainly contributed by the use of a coordinator, which allows non-convex collision constraints to be resolved collaboratively. In comparison, D-MPC includes constraints from all agents at any given timestep, which results in additional collision constraints that may not be necessary for finding a feasible solution. In addition, D-MPC is prone to deadlocks as all interacting agents resolve the same set of constraints without any knowledge of how the other agents are reacting. In Pr-MPC, as the number of agents increases, lower priority robots are penalized more by including additional constraints from the higher priority robots. This results in them solving significantly harder optimization problems, causing infeasibilities.

The more efficient conflict resolution structure of CB-MPC also manifests itself in the computation cost. As shown in Fig. 6(b), CB-MPC maintains a stable computation load despite increasing the number of robots. In comparison, Pr-MPC's solve time grows significantly higher and also fluctuates more. D-MPC maintains a similar average solve time compared to CB-MPC in 2 and 3 robot trials. However, it's average solve time grows significantly for the 4 robot case. It is important to note that the saving in computation cost for CB-MPC does not compromise the solution quality, evidenced by Fig. 6(c), which compares the makespan across all the test cases.

To summarize, this experiment provides additional evidence for hypothesis 2, as CB-MPC is shown to out-perform other multi-robot MPC variations, without sacrificing solution quality over randomized trials.

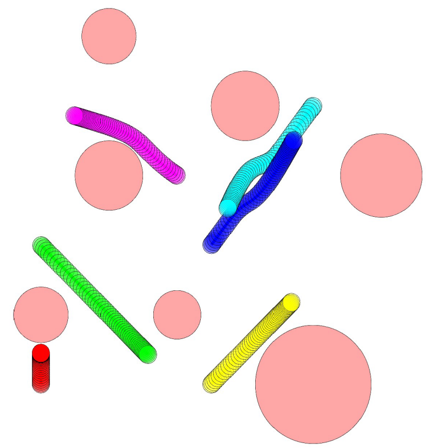


Fig. 5. Cluttered environment example scenario

VII. Conclusion and Future Work

This paper presents a scalable multi-robot motion planning algorithm combining the efficient high-level conflict resolution structure of CBS with MPC as the low-level planner to efficiently resolve inter-agent collision constraints through constraint splitting. We demonstrate that naively tracking the high-level plans generated from MAPF algorithms is insufficient and prone to execution failure under realistic conditions. In addition, we present results showing a significant advantage to CB-MPC in terms of success rate, computation cost, and solution quality compared to joint, prioritized, and distributed MPC across four different environments with varying number of robots and obstacles. Finally, we empirically show that CB-MPC integrated with a high-level MAPF planner like CBS can scale to a larger number of robots and environments and be an effective substitute for expensive full-horizon kinodynamic planners.

This research can be extended in multiple interesting directions. First, the low-level MPC planner in CB-MPC can be swapped with a robust version to reason about state and control uncertainty for each agent and deal with real sensor data. A chance-constrained planning framework can then be used to reason about inter-agent conflicts probabilistically similar to [32]. Second, although scalable and effective, the feasibility and convergence of CB-MPC is directly tied to the choice of hyper-parameters. As a result, there will still be MPC failures in some cases. In those cases, systematic relaxation of constraints similar to [33] can be key to resolving the failure. Third, recent reinforcement learning approaches have demonstrated promise in solving the MAPF problem. However, they often have poor convergence due to their sparse reward structure. A diverse dataset of CB-MPC trajectories can be used as expert demonstrations to improve convergence of these learning algorithms to train decentralized policies using imitation learning.

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] A. Andreychuk, K. Yakovlev, P. Surynek *et al.*, "Multi-agent pathfinding with continuous time," *Artificial Intelligence*, p. 103662, 2022.

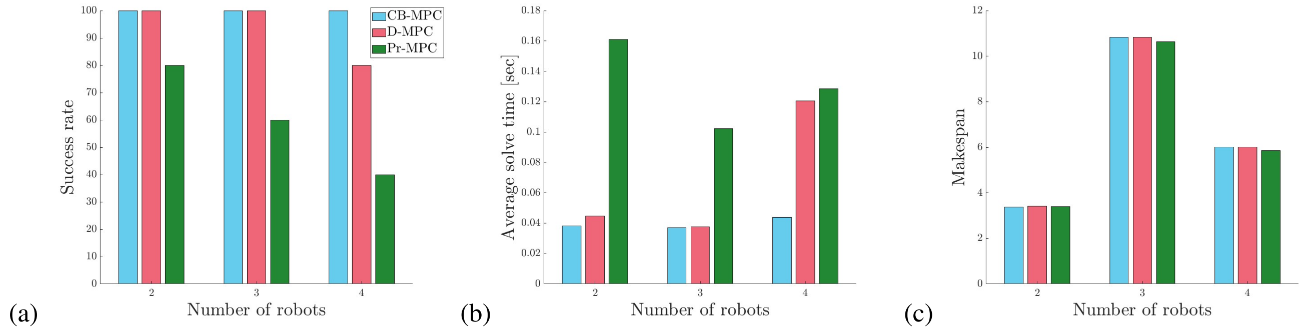


Fig. 6. Randomized cluttered environment results. (a) Success rate (higher is better) (b) Solve time (lower is better) (c) Makespan (lower is better). CB-MPC succeeds in all cases with significantly better average solve time compared to D-MPC and Pr-MPC, with similar solution quality (i.e. makespan).

[3] E. Boyarski, A. Felner, R. Stern *et al.*, "ICBS: Improved conflict-based search algorithm for multi-agent pathfinding," in *International Joint Conference on Artificial Intelligence*, 2015.

[4] 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*, 2014.

[5] J. Li, A. Felner, E. Boyarski *et al.*, "Improved heuristics for multi-agent path finding with conflict-based search," in *International Joint Conference on Artificial Intelligence*, 2019, pp. 442–449.

[6] W. Hönig, T. S. Kumar, L. Cohen *et al.*, "Multi-agent path finding with kinematic constraints," in *International Conference on Automated Planning and Scheduling*, 2016.

[7] W. Hönig, S. Kiesel, A. Tinka *et al.*, "Persistent and robust execution of mapf schedules in warehouses," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1125–1131, 2019.

[8] F. E. Schneider and D. Wildermuth, "A potential field based approach to multi robot formation navigation," in *IEEE International Conference on Robotics, Intelligent Systems and Signal*, vol. 1, 2003, pp. 680–685.

[9] H. G. Tanner and A. Kumar, "Towards decentralization of multi-robot navigation functions," in *IEEE International Conference on Robotics and Automation*, 2005, pp. 4132–4137.

[10] R. Gayle, W. Moss, M. C. Lin, and D. Manocha, "Multi-robot coordination using generalized social potential fields," in *IEEE International Conference on Robotics and Automation*, 2009, pp. 106–113.

[11] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[12] M. Čáp, P. Novák, J. Vokřínek, and M. Pěchouček, "Multi-agent RRT*: Sampling-based cooperative pathfinding," *arXiv preprint arXiv:1302.2828*, 2013.

[13] K. Solovey, O. Salzman, and D. Halperin, "Finding a needle in an exponential haystack: Discrete RRT for exploration of implicit roadmaps in multi-robot motion planning," in *Algorithmic Foundations of Robotics XI*. Springer, 2015, pp. 591–607.

[14] J. Kottinger, S. Almagor, and M. Lahijanian, "Conflict-based search for multi-robot motion planning with kinodynamic constraints," *arXiv preprint arXiv:2207.00576*, 2022.

[15] R. Shome, K. Solovey, A. Dobson *et al.*, "dRRT*: Scalable and informed asymptotically-optimal multi-robot motion planning," *Autonomous Robots*, vol. 44, no. 3, pp. 443–467, 2020.

[16] B. Li, Y. Ouyang, Y. Zhang *et al.*, "Optimal cooperative maneuver planning for multiple nonholonomic robots in a tiny environment via adaptive-scaling constrained optimization," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1511–1518, 2021.

[17] Y. Zhou, H. Hu, Y. Liu *et al.*, "A real-time and fully distributed approach to motion planning for multirobot systems," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 12, pp. 2636–2650, 2017.

[18] Y. Chen, M. Cutler, and J. P. How, "Decoupled multiagent path planning via incremental sequential convex programming," in *IEEE International Conference on Robotics and Automation*, 2015, pp. 5954–5961.

[19] C. E. Luis, M. Vukosavljev, and A. P. Schoellig, "Online trajectory generation with distributed model predictive control for multi-robot motion planning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 604–611, 2020.

[20] R. Firoozi, L. Ferranti, X. Zhang *et al.*, "A distributed multi-robot coordination algorithm for navigation in tight environments," *arXiv preprint arXiv:2006.11492*, 2020.

[21] G. Wagner and H. Choset, "M*: A complete multirobot path planning algorithm with performance bounds," in *IEEE/RSJ international conference on intelligent robots and systems*, 2011, pp. 3260–3267.

[22] J. Li, A. Tinka, S. Kiesel *et al.*, "Lifelong multi-agent path finding in large-scale warehouses," in *AAAI Conference on Artificial Intelligence*, vol. 35, no. 13, 2021, pp. 11 272–11 281.

[23] P. Velagapudi, K. Sycara, and P. Scerri, "Decentralized prioritized planning in large multirobot teams," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 4603–4609.

[24] M. Cáp, P. Novák, M. Seleckỳ *et al.*, "Asynchronous decentralized prioritized planning for coordination in multi-robot system," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 3822–3829.

[25] M. Bennewitz, W. Burgard, and S. Thrun, "Optimizing schedules for prioritized path planning of multi-robot systems," in *IEEE International Conference on Robotics and Automation*, vol. 1, 2001, pp. 271–276.

[26] J. Li, M. Ran, and L. Xie, "Efficient trajectory planning for multiple non-holonomic mobile robots via prioritized trajectory optimization," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 405–412, 2020.

[27] F. Duchoň, A. Babinec, M. Kajan *et al.*, "Path planning with modified a star algorithm for a mobile robot," *Procedia Engineering*, vol. 96, pp. 59–69, 2014.

[28] L. Ferranti, L. Lyons, R. R. Negenborn *et al.*, "Distributed nonlinear trajectory optimization for multi-robot motion planning," *IEEE Transactions on Control Systems Technology*, 2022.

[29] L. Riegger, M. Carlander, N. Lidander *et al.*, "Centralized MPC for autonomous intersection crossing," in *IEEE International Conference on Intelligent Transportation Systems*, 2016, pp. 1372–1377.

[30] J. Lofberg, "Yalmip: A toolbox for modeling and optimization in matlab," in *IEEE International Conference on Robotics and Automation*, 2004, pp. 284–289.

[31] L. T. Biegler and V. M. Zavala, "Large-scale nonlinear programming using IPOPT: An integrating framework for enterprise-wide dynamic optimization," *Computers & Chemical Engineering*, vol. 33, no. 3, pp. 575–582, 2009.

[32] M. Castillo-Lopez, P. Ludivig, S. A. Sajadi-Alamdari *et al.*, "A real-time approach for chance-constrained motion planning with dynamic obstacles," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3620–3625, 2020.

[33] R. J. M. Afonso and R. K. H. Galvão, "Infeasibility handling in constrained MPC," *Frontiers of Model Predictive Control*, pp. 47–64, 2012.