Conflict-Based Search for Multi-Robot Motion Planning with Kinodynamic Constraints

Justin Kottinger[1], Shaull Almagor[2], and Morteza Lahijanian[1,3]

Abstract— *Multi-robot motion planning* (MRMP) is the fundamental problem of finding non-colliding trajectories for multiple robots acting in an environment, under kinodynamic constraints. Due to its complexity, existing algorithms are either incomplete, or utilize simplifying assumptions. This work introduces *Kinodynamic Conflict-Based Search* (K-CBS), a decentralized MRMP algorithm that is general, scalable, and probabilistically complete. The algorithm takes inspiration from successful solutions to the discrete analogue of MRMP over finite graphs, known as *Multi-Agent Path Finding* (MAPF). Specifically, we adapt ideas from *Conflict-Based Search* (CBS)–a popular decentralized MAPF algorithm–to the MRMP setting. The novelty of our approach is that we work directly in the continuous domain, without discretization. In particular, the kinodynamic constraints are treated natively. K-CBS plans for each robot individually using a low-level planner and grows a conflict tree to resolve collisions between robots by defining constraints. The low-level planner can be any sampling-based, tree-search algorithm for kinodynamic robots, thus lifting existing planners for single robots to the multi-robot setting. We show that K-CBS inherits the (probabilistic) completeness of the low-level planner. We illustrate the generality and performance of K-CBS in several case studies and benchmarks.

I. INTRODUCTION

Multi-robot motion planning (MRMP) is a fundamental challenge in robotics and artificial intelligence. The goal of MRMP is to compute dynamically-feasible trajectories for multiple robots to reach their respective goal regions such that, when executed simultaneously, the robots do not collide with obstacles nor with each other. MRMP has vast applications, from warehouse robotics and delivery to planetary exploration and search-and-rescue. The difficulty of MRMP lies in the size of the planning space, which grows exponentially with the number of robots. In addition, there are constraints posed by the robots kinodynamics that need to be respected, which by itself is a difficult problem even for a single robot. For this reason, either simplifying assumptions are posed on the problem (e.g., removal of the dynamic constraints) or incomplete algorithms are accepted for the purposes of scalability. In this work, without any simplifying assumptions, we aim to develop a scalable kinodynamic planner for MRMP with (probabilistic) completeness guarantees.

MRMP has been extensively studied in the robotics community. The approaches are generally classified into two categories: *centralized* (coupled) and *decentralized* (decoupled). The centralized methods (e.g., [1]–[3]) operate over the composed state space of all the robots. The advantage of these approaches is that they allow the use of the rich library of existing single-robot motion planners (e.g., RRT [4] and KPIECE [5]) and automatically inherit their completeness and optimality. The disadvantage is that they are not scalable. In contrast, decentralized approaches divide the problem into several sub-problems and solve each separately [6], [7]. While successful in scalability, existing decentralized MRMP algorithms are mostly incomplete, i.e., do not guarantee to find a solution if one exists.

The discrete analogue of MRMP, dubbed *multi-agent path finding* (MAPF), has received much attention in the AI and discrete planning community. There, the agents travel along edges in a discrete graph (rather than a continuous space). The main focus in MAPF is scalability, and as a result, advanced algorithms have been developed that are not only scalable, but also have completeness and optimality guarantees (see [8] and references therein). A popular decentralized algorithm is *Conflict-Based Search* (CBS) [9], which has many variants [10], including an explainable one [11]. CBS plans for each agent separately using low-level search (e.g., A^*). It then identifies collisions in the proposed plans and re-plans for colliding agents after placing path-constraints that resolve the collision. Thus, CBS grows a "conflict tree", where each node represents a (possibly colliding) plan.

Unfortunately, such MAPF algorithms cannot be utilized to solve the MRMP problem in a straightforward manner, since MAPF abstracts the continuous workspace into a finite graph (e.g., a grid world), and ignores the shape of the robots. It also bypasses all dynamic constraints by assuming every edge of the graph is realizable with a known time duration.

Several works have focused on adapting MAPF algorithms to handle the challenges of MRMP, e.g., [12]–[15]. Work [12] combines continuous *probabilistic roadmaps* (PRM) planner [16] with CBS to solve MRMP by ignoring the kinodynamic constraints. Work [13] accounts for dynamics and uses CBS with optimization techniques for a swarm of quadrotors. Specifically, it first uses CBS to compute discrete paths over a discretized workspace. Then, it generates optimal controllers to follow the discrete paths by assuming sphere-shaped quadrotors with linear dynamics. While performing well for specific robots, those approaches do not generalize to robots with nonlinear dynamics.

To deal with general nonlinear dynamics, sampling-based, tree motion planners are a powerful approach. To this end, recent works [14], [17] combine discrete search with a

13494

sampling-based tree planner to solve the MRMP problem. The algorithm first discretizes the composed configuration space of all the robots using PRM, and then uses a discrete planner to find candidate paths. Then, a sampling-based motion tree is grown in the composed space, guided by the paths. If unsuccessful, a different set of paths is computed and the process repeats until a valid trajectory is found. The work uses a centralized discrete planner at the high-level and grows the motion tree in the composed configuration space. Hence, it is probabilistically complete (i.e., the probability of finding a solution tends to one as the planning time tends to infinity, if a solution exists), but it may be slow. To speed up the planner, a feedback PID controller is assumed to drive each robot from one discrete region in the guide to the next. While fast planning times can be achieved using this method, the choice for the size of PRM (discretization) is unclear, and the use of a PID controller is limiting, as they may not exist.

In this work, we introduce a decentralized, scalable MRMP planner that treats the kinodynamic constraints of the robots natively with probabilistic completeness guarantees. Unlike related work that directly adopts MAPF algorithms on a discretized version of the problem, we incorporate the ideas that make CBS successful into the continuous domain using a sampling-based method. Our algorithm, dubbed *Kinodynamic CBS* (K-CBS), like CBS, uses a low-level search and maintains a conflict tree. The low-level search can be any (kinodynamic) sampling-based tree planner (e.g., RRT and KPIECE). In each iteration of the algorithm, a plan is computed for an individual robot given a set of constraints (obstacles). Then, collisions between the robot trajectories is checked. If one exists, time-dependent obstacles are defined as constraints in the constraint tree, and a new planning query is specified accordingly. To ensure probabilistic completeness, we introduce a *merge* method, by which we merge robots whose plans often conflict, into a single meta-robot.

The main contribution of this work is a decentralized, probabilistically-complete MRMP algorithm that is capable of generating kinodynamically feasible plans efficiently. Our planner, K-CBS, naturally adapts CBS from MAPF to MRMP. This lifts every off-the-shelf (kinodynamic) sampling-based, tree-search planner to the multi-robot setting and removes all the limitations (assumptions) of state-of-the-art MRMP planners. Specifically, K-CBS operates completely in the continuous state space of the agents, hence, it does not require discretization nor a feedback-control design. It easily works with arbitrary, possibly heterogeneous, nonlinear dynamical models, and is capable of solving very complex MRMP instances efficiently. We empirically show the efficacy of K-CBS in many case studies, highlighting our algorithm's generality and performance improvements.

II. PROBLEM FORMULATION

Consider $k \in \mathbb{N}$ robotic systems, in a shared, bounded workspace $W \subseteq \mathbb{R}^d$, $d \in \{2, 3\}$, which includes a finite set of obstacles $\mathcal{O}$, where every obstacle $o \in \mathcal{O}$ is a closed subset of W. Furthermore, every robot $i \in \{1, 2, \ldots, k\}$ has a (rigid) body $\mathcal{S}_i$, and its motion is subjected to the following dynamic constraint:

$$\dot{\mathbf{x}}_i = f_i(\mathbf{x}_i, \mathbf{u}_i), \quad \mathbf{x}_i \in X_i \subseteq \mathbb{R}^{n_i} \quad \mathbf{u}_i = U_i \subseteq \mathbb{R}^{m_i}, \quad (1)$$

where $n_i \geq d$, X_i and U_i are robot i's state and input spaces, respectively, and $f_i : X_i \times U_i \to X_i$ is a continuous and possibly nonlinear function (vector field). With an abuse of notation, we denote by $\mathcal{S}_i(\mathbf{x}_i) \subseteq W$ the set of points that robot i's body occupies in the workspace at state $\mathbf{x}_i \in X_i$.

Given a time interval $[t_0, t_{fi}]$, where $t_0, t_{fi} \in \mathbb{R}_{\geq 0}$ and $t_0 \leq t_{fi}$, a controller $\mathbf{u}_i : [t_0, t_{fi}] \to U_i$, and initial state $\mathbf{x}_{i,0} \in X_i$ for robot i, function f_i can be integrated up to time t_{fi} to form a *trajectory* $T_i : \mathbb{R}_{\geq 0} \to X_i$ for robot i, where $T_i(t_0) = \mathbf{x}_{i,0}$. We assume all the robots' trajectories start at time t_0. Further, we assume that, once robot i reaches the end of its trajectory $T_i(t_{fi})$ at time t_{fi}, it remains there, i.e., $T_i(t') = T_i(t_{fi})$ for all $t' \geq t_{fi}$. Given trajectories $T_1, T_2, \ldots, T_k$, denote by $t_f = \max\{t_{f1}, t_{f2}, \ldots t_{fk}\}$ the final time of the longest trajectory. The trajectories are *collision-free* if the following conditions hold for $t \in [t_0, t_f]$:

1) $\forall 1 \leq i \leq k, \forall o \in \mathcal{O}, \quad \mathcal{S}_i(T_i(t)) \cap o = \emptyset$
2) $\forall 1 \leq i < j \leq k, \quad \mathcal{S}_i(T_i(t)) \cap \mathcal{S}_j(T_j(t)) = \emptyset$

Condition 1 indicates no collision with obstacles, and Condition 2 indicates no robot-robot collision.

Given a goal region for every robot, the objective in MRMP is to find a feasible trajectory for every robot to reach its goal without colliding with obstacles nor other robots:

Problem 1 (MRMP): Given k robots in workspace W with dynamics as in (1), initial states $\mathbf{x}_{i,0} \in X_i$ and goal regions $X_i^G \subseteq X_i$ for all $1 \leq i \leq k$, find a controller $\mathbf{u}_i : [t_0, t_{fi}] \to U_i$ for every $1 \leq i \leq k$ such that the induced trajectories $T_1, \ldots, T_k$ are collision-free, and T_i takes agent i from $T_i(t_0) = \mathbf{x}_{i,0}$ to $T_i(t_{fi}) \in X_i^G$ for every $1 \leq i \leq k$.

III. METHODOLOGY

We now present our solution to Problem 1, which is inspired by CBS. To this end, we describe CBS for MAPF and then present the necessary extensions to solve Problem 1. Finally, we introduce *Kinodynamic* CBS (K-CBS).

A. Conflict-Based Search for MAPF

CBS is a two-level search on the space of possible MAPF plans, and consists of a high-level conflict-tree search and a low-level graph search. At the high-level, CBS keeps track of a *constraint-tree*, in which each node represents a suggested plan, which might have collisions, referred to as *conflicts*. Initially, a root node is obtained by using a low-level graph search algorithm, typically A^*, to find a path for each robot from start to goal, ignoring the other robots (hence the decentralized nature of the method).

At each iteration, CBS picks an unexplored node from the constraint-tree (CT), based on some heuristic, and identifies the conflicts (namely collisions) in the node's corresponding plan. Then, CBS tries to resolve the conflicts by creating child nodes based on the conflicts, as follows: if robots i and j collide at time t in vertex v (denoted $\langle i, j, v, t \rangle$), then two child nodes are created, one with the constraint that robot i cannot be in vertex v at time t (denoted $\langle i, v, t \rangle$), and the

13495

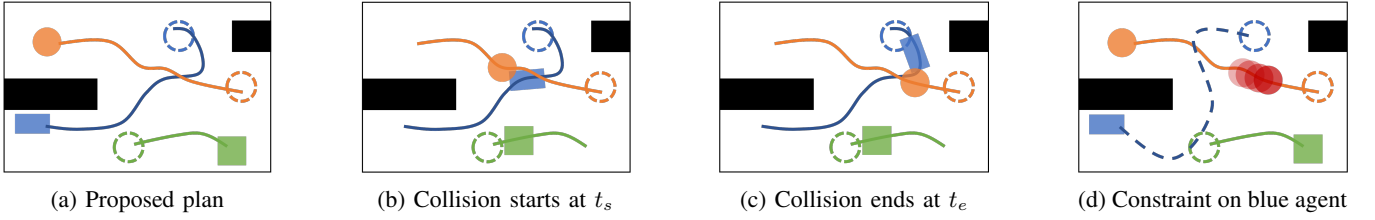| (a) Proposed plan | (b) Collision starts at t_s | (c) Collision ends at t_e | (d) Constraint on blue agent |

Fig. 1: Many conflicts occur during a single collision between the orange and blue robots within time $t \in [t_s, t_e]$. The set of conflicts creates one constraint for each robot, which are reasoned about separately. Constraint on orange agent not shown.

other dually for robot j. Then, in each child node, a low-level search is used to replan a path for the newly-constrained robot, given the set of constraints obtained thus far along the branch of the CT. This process repeats until either a non-colliding plan is found, or no new nodes are created in the CT, at which point CBS returns with no solution.

B. MRMP Constraints

CBS definitions of conflicts and constraints fail for MRMP because (i) robots do not occupy vertices, and (ii) robots could have unique bodies. Additionally, due to the continuous time nature of MRMP, collisions occur over time intervals rather than time points as in MAPF. Thus, formal definitions of collisions and constraints for MRMP are needed.

Given two trajectories T_i and T_j for robots i and j, respectively, recall that a *collision* occurs between robots i and j at every time point that Condition 2 is violated. The duration of a collision is the *continuous* time interval the two robots are in collision as they evolve along their respective trajectories, and we define a *conflict* to be $K = \langle i, j, [t_s, t_e] \rangle$, where t_s and t_e are the start and end time of this interval. Note that there may be multiple conflicts between trajectories T_i and T_j, in which case their time intervals are disjoint.

A conflict $K = \langle i, j, [t_s, t_e] \rangle$ is resolved by imposing a *constraint* $\mathcal{C}$ on either robot i or robot j. Intuitively, a constraint on robot i requires it to avoid colliding with robot j for all $t_r \in [t_s, t_e]$ by viewing $\mathcal{S}_j(T_j(t_r))$ as a moving (time-dependent) obstacle. Formally, the constraint on robot i for conflict K is $\mathcal{C}_i = \{\langle i, \mathcal{S}_j(T_j(t_r)), t_r \rangle : t_r \in [t_s, t_e]\}$. Thus, $\mathcal{C}_i$ treats $\mathcal{S}_j(T_j(t_r))$ as a moving obstacle rather than a static one for the entire time duration of the conflict (see Fig. 1). We dually define the constraint $\mathcal{C}_j$ for robot j. This splitting is the crux of the performance of the approach.

Next, we describe how to resolve a conflict by re-planning for an individual robot with the corresponding constraint.

C. Single-Robot Planning under MRMP Constraints

To (re-)plan for individual robots, we can use any existing sampling-based, tree-search algorithm (e.g. RRT) since they are efficient in finding kinodynamically feasible trajectories. A generic form of such planner $\mathcal{X}$ begins by initializing a graph G at the initial state. Then, for N iterations, the algorithm samples a state $\mathbf{x}_{rand}$ and an input, integrates (1) to extend toward $\mathbf{x}_{rand}$, and adds the trajectory to the graph after verifying that it is collision free. The process continues

until a trajectory from the initial state to the goal region is found, or N iterations is reached.

Such planners perform well for single-query, single-robot kinodynamic planning, but off-the-shelf, they are unable to guarantee the satisfaction of a set of constraints as defined in Sec. III-B. To this end, we present Constrained-$\mathcal{X}$ (CSTR-$\mathcal{X}$), an extension of planner $\mathcal{X}$ for finding trajectories that do not violate a set of constraints $\mathcal{C}$.

CSTR-$\mathcal{X}$ takes two additional inputs than $\mathcal{X}$: a set of constraints $\mathcal{C}$, and an existing graph (motion tree) $\mathcal{T}$. If $\mathcal{T}$ is non-empty, CSTR-$\mathcal{X}$ utilizes the existing graph. Otherwise, it initializes a new graph G with a single initial state $\mathbf{x}_0$. Then, to check if a trajectory is valid, CSTR-$\mathcal{X}$ checks against both static obstacles $\mathcal{O}$ and constraints (moving obstacles) in $\mathcal{C}$. Similar to existing validity checking functions, checking against $\mathcal{C}$ can be done efficiently, e.g., by discretizing the continuous trajectory $\overrightarrow{\mathbf{x}_{near}, \mathbf{x}_{new}}$ into a set of points $\{x(\tau_1), \ldots, x(\tau_L)\}$ by sampling time at a fine resolution Δt, and checking $\mathcal{S}(\mathbf{x}(\tau_l))$ against all $\mathcal{C}^r \in \mathcal{C}$ for all $1 \leq l \leq L$ such that τ_l is the time range of $\mathcal{C}^r$, i.e., $\tau_l \in [t_s^r, t_e^r]$.

The output of CSTR-$\mathcal{X}$ is a trajectory that respects all the constraints, if one is found in N iterations; otherwise, it returns the full motion tree, as well as the constraint $\mathcal{C}_{\max}$ that was violated the most. K-CBS uses the motion tree to continue planning when needed, and uses $\mathcal{C}_{\max}$ to identify tangled plans, as discussed below. Note that CSTR-$\mathcal{X}$ defaults to Planner $\mathcal{X}$ if both additional parameters $\mathcal{T}$ and $\mathcal{C}$ are empty sets. See [18] for the pseudocode of CSTR-$\mathcal{X}$.

D. Kinodynamic Conflict-Based Search (K-CBS)

We now introduce the high-level algorithm of K-CBS (Alg. 1). Recall that the input to Problem 1 comprises a state space X_i, input space U_i, vector field f_i, body $\mathcal{S}_i$, initial state $\mathbf{x}_{i,0}$, and goal region X_i^G for every robot $i \in \{1, \ldots, k\}$. For brevity, we define a model $\mathcal{M}_i = \{X_i, U_i, f_i, \mathbf{x}_{i,0}, X_i^G\}$ as the inputs pertaining to robot i. The input to K-CBS is then a model for every robot $\{\mathcal{M}_i\}_{i=1}^k$, a parameter N of the maximum number of iterations of CSTR-$\mathcal{X}$, and a merging parameter B. The output of K-CBS is a set of trajectories $\{T_1, \ldots, T_k\}$ that is the solution to the MRMP problem.

K-CBS begins by calculating trajectories for every robot individually by calling CSTR-$\mathcal{X}$ with an empty initial tree and empty set of constraints. These trajectories serve as the initial plan for the root node of the constraint-tree, which is added to a priority-queue Q (Lines 1- 2). The cost of node $n \in Q$ is the sum of control durations over all trajectories

Algorithm 1: Kinodynamic-CBS (K-CBS)

Input: $\{\mathcal{M}_i\}_{i=1}^k$, N, B
Output: Collision-free trajectories $\{T_1, \ldots, T_k\}$

1   $Q, n_0, p_0 \leftarrow \emptyset$; calc_init_plan($p_0$);
2   n_0.plan $\leftarrow p_0$; Q.add(n_0);
3   **while** solution not found **do**
4      $c \leftarrow Q$.top();
5   **if** c.plan is empty **then**
6   Q.pop(); $\mathcal{C}_{\max} \leftarrow$ replan(c);
7   **if** shouldMerge($\mathcal{C}_{\max}$, B) **then**
8   **return** mrg_&_rstrt($\{\mathcal{M}_i\}|_1^k$, N,B,$\mathcal{C}_{max}$)
9   Q.add(c);
10   **else**
11   K $\leftarrow$ validatePlan(c.plan);
12   **if** K is empty **then**
13   **return** c.plan
14   **else if** shouldMerge(K, B) **then**
15   **return** mrg_&_rstrt($\{\mathcal{M}_i\}_{i=1}^k$, N,B,K)
16   **else**
17   Q.pop();
18   **for** every robot i in K **do**
19                 $c_{new}.\mathcal{C} \leftarrow c.\mathcal{C} \cup$ K.getCSTR(i);
20                 $\mathcal{C}_{\max} \leftarrow$ replan(c_{new});
21   Q.add(c_{new});

Fig. 2: K-CBS tree. Robots must cross a narrow corridor.

in the plan of n (and ∞ if n does not contain a plan yet). The smaller the cost of n, the higher its priority in Q.

At every iteration of K-CBS, a node $c \in Q$ with minimal cost is selected (Line 4). If c has no plan, another call to CSTR-$\mathcal{X}$ is performed to try to obtain such a plan (Lines 5-9). If c has a plan, it is evaluated for conflicts (Line 11). If no conflict exists, then the plan is returned as the solution. Otherwise, an arbitrary conflict K is selected and for each robot in K, a constraint is generated and CSTR-$\mathcal{X}$ is called to calculate a valid trajectory, storing the result of the search in a new node, which is added to Q (Lines 17- 21).

As K-CBS explores the space of all plans, it is possible that many conflicts are found between a particular pair of robots. This may indicate that their motions are "coupled," and hence, it is difficult to plan for them separately. In that case, K-CBS can merge the pair into a meta-robot and plan for an MRMP problem with $k - 1$ robots.

Merge and Restart: To clarify the concept of "coupled" robots, consider the example in Fig. 2, where two robots must cross a narrow corridor that is just wide enough for the both of them. Assume that both robots have constant velocity toward their respective target and they only control their steering angle. When presenting K-CBS with this problem instance, the root node of the constraint tree could look like the one seen in at the top of Fig. 2, causing a collision. This creates a branching of the root node into two child nodes. One with a constraint on the blue robot to avoid the orange robot, and dually for the orange robot (constraints are visual-
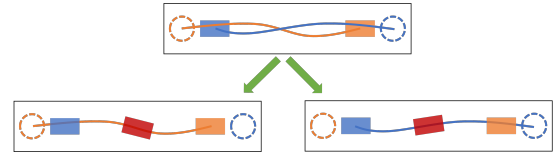
ized in red). In both cases, the constraint entirely prevents a valid plan for either agent, although a-priori a plan does exist. If this behavior is left unresolved, K-CBS would never find a valid plan. Thus, we employ the *merge and restart* technique, borrowed from the discrete setting [19]. Note that in the discrete setting, this is merely a speedup heuristic, whereas for the continuous case it is crucial for completeness. While many merge techniques are available, we utilize a simple, experimentally effective policy from [20], whereby robots i and j are merged if the number of conflicts between them exceeds some pre-defined threshold B.

Once agents i and j are to be merged, the two robots are composed together and another instance of K-CBS is solved for $k - 1$ robots where one of the robots in the newly formed problem is a meta-robot. Note that in the case of Fig. 2, eventually K-CBS merges the two robots together and plans trajectories that allow both robots to navigate the corridor simultaneously. Also note that we count as conflicts both nodes where a conflict occurs, as well as nodes where no plan is found, via $\mathcal{C}_{\max}$. See the extended version [18] for the detailed pseudocode for K-CBS with the merge policy.

Theorem 1 (Completeness): K-CBS is probabilistically complete if Planner $\mathcal{X}$ is probabilistically complete.

The proof of Theorem 1 is in [18]. We remark that it may be possible to further improve the performance of K-CBS using heuristics. Nonetheless, this requires careful treatment and deep understanding of the MRMP problem. Not every successful heuristic used in MAPF is beneficial in the continuous domain. For instance, work [21] shows that, in CBS, significant speedups can be achieved by *bypassing* (BP) a split of a constraint-tree node by generating a single constraint and attempting to resolve it by calling the low-level planner in the same node. Our studies however do not show such improvements by incorporating BP in K-CBS.

IV. EXPERIMENTS AND BENCHMARKS

Here, we show the performance of K-CBS in four different environments (Open, Narrow, Cluttered, and Large) shown in Fig. 3 with various numbers of robots, each with 2nd-order car dynamics (5-dimensional state space):

$$\dot{x} = v \cos\theta, \; \dot{y} = v \sin\theta, \; \dot{\theta} = \frac{v}{l}\tan\phi, \; \dot{v} = u_1, \; \dot{\phi} = u_2$$

where x and y define the position, θ is the orientation, v is the velocity, ϕ is the steering angle, u_1 is the acceleration rate and u_2 is the steering rate. Each robot has a rectangular shape with length $l = 0.7$ and width $w = 0.5$.

In each environment, we benchmarked the performance and scalability of K-CBS against two baseline approaches: centralized RRT (cRRT) and prioritized RRT (pRRT) [22]. cRRT

(a) Open Env. (10 × 10) (b) Narrow Env. (5 × 4) (c) Cluttered Env. (15 × 15) (d) Large Env. (33 × 33)
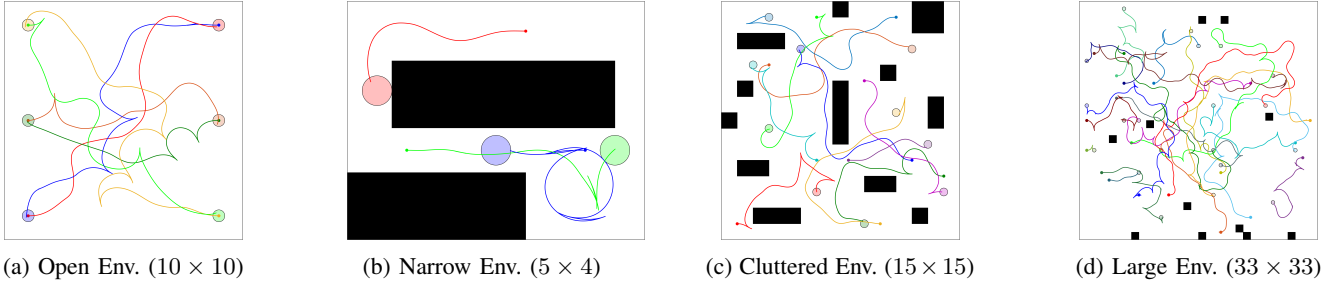
Fig. 3: MRMP benchmark environments. The initial states are shown with small circles and goal regions with large circles.

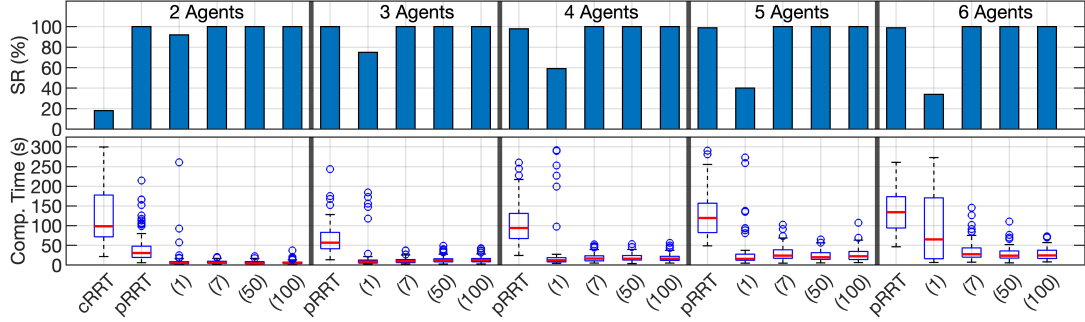Fig. 4: Benchmark results for the Open Env. in Fig. 3a. Column label (i) denotes K-CBS with merging threshold $B = i$.
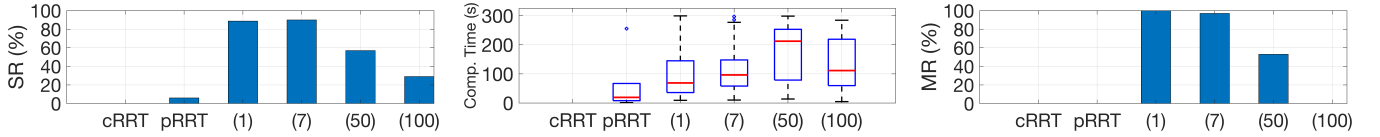

Fig. 5: Benchmark results for the Narrow Env. in Fig.3b Column label (i) denotes K-CBS with merging threshold $B = i$.
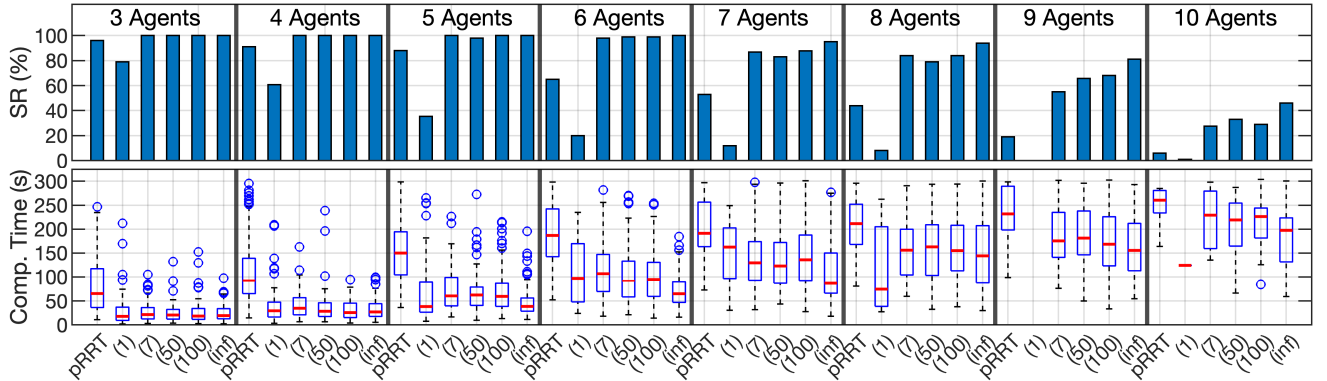

Fig. 6: Benchmark results for the Cluttered Env. in Fig. 3c. Column label (i) denotes K-CBS with merging threshold $B = i$.

is a centralized method that plans in the composite space of all the robots, whereas pRRT is a decentralized algorithm that plans for each robot i in turn, treating the plans for robots $1, \ldots, i - 1$ as timed obstacles. We also varied the merging parameter B to show its effect on the performance of K-CBS. Every benchmark consisted of 100 problem instances with a timeout of 5 minutes. The comparison metrics are *success rate (SR)*, *computation time* (of successful runs), and *merge rate (MR)*. The results are shown in Fig. 4-6 and Table I. We note that in Fig. 4-6, column labels (i) denote using K-CBS with merging threshold $B = i$.

Our implementation of K-CBS utilized the classical RRT

as the low-lever planner. The code is publicly available on GitHub [23]. The benchmarks were performed on AMD Ryzen 7 3.9 GHz CPU and 64 GB of RAM.

Open Environment: In this environment, the robots have to swap positions, i.e., each robot starts in the center of the goal region of another robot (Fig.3a). We varied the number of robots from 2 to 6. The results are shown in Fig. 4. Note that cRRT is only able to find solutions for the two-robot case (10-dimensional composed state space) with a low success rate. For larger number of robots, its success rate was zero (hence, not shown in Fig. 4). On the other hand pRRT, due to its decentralized nature, performs well in this environment

13498

TABLE I: Benchmark results for `K-CBS` in variations of the Large Env. (Fig. 3d). pRRT always timed out.

# Agents	# Obstacles	Succ. rate (%)	Comp. time (s)
10	10	92	161.7
15	10	74	226.7
20	10	30	290.8
10	5	98	126.7
15	5	92	203.1
20	5	50	279.8
10	0	100	104.8
15	0	94	152.6
20	0	74	239.8

with a success rate of 100% for all cases. However, as the number of robots increases, pRRT spends an increasingly large amount of time collision checking against prior robots during planning, resulting in long computation times. `K-CBS` does not consider other agents until an entire plan is found. Thus, it scales much better than pRRT with 1 to 2 orders of magnitude smaller computation times, especially if B is large. Observe that for $B = 1$, `K-CBS` merges robots after one conflict, leading to planning in a larger dimensional space. Clearly, in open environments such a merge is inadvisable.

Narrow Environment: The narrow corridors in Fig. 3b are only wide enough for a single robot to traverse at a given time. Thus, the blue robot must remain out of the corridor until the green agent leaves it, at which point it may travel safely to goal. The benchmark results are shown in Fig. 5. pRRT struggles in this space because there is no way to consider the blue and green agents simultaneously to ensure success. On the other hand, cRRT, which considers all three agents simultaneously, always fails due to the blow-up in the (15-dimensional) search space. `K-CBS` finds a balance in the middle by merging the blue and green robots but not the red one. This leads to a high success rate in this space, particularly for low values of B, where the composed agents have the most time to traverse the narrow corridor.

Cluttered Environment: We further consider the scalability of `K-CBS` against pRRT by analyzing it on a larger space with many obstacles and increasing the number of robots (Fig. 3c). The results are shown in Fig. 6. Once again, we see that pRRT spends most of its time collision checking and does not scale past 7 robots. For `K-CBS`, higher values of B are more successful and faster in all cases than pRRT.

Large Environment: We test the limits of `K-CBS` using the space from Fig. 3d. The results are in Table I. pRRT failed on every instance, but `K-CBS` reliably scales up to 20 agents.

Overall, our results show that `K-CBS` is scalable and merging is most useful when agents must carefully coordinate their movement in a small space (e.g., corridor). In open spaces, large values for B are best to avoid unnecessary merges. Still, `K-CBS` solves very complex MRMP instances reliably and quickly compared to current methods.

V. CONCLUDING REMARKS

This work introduces an MRMP algorithm that is scalable, probabilistically complete, and poses no simplifying assumptions on the problem. It initially operates as a fully-decentralized method by individually planning for each robot and inducing constraints on individual robots as collisions occur. As it progresses, it can merge coupled robots into a meta-robot. Thus, it utilizes centralized planning for only those robots while still treating others individually, leading to strong performance in various environments and settings.

REFERENCES

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

[2] R. Shome, K. Solovey, A. Dobson, D. Halperin, and K. E. Bekris, "drrt*: Scalable and informed asymptotically-optimal multi-robot motion planning," *Autonomous Robots*, vol. 44, no. 3, pp. 443–467, 2020.

[3] J. Kottinger, S. Almagor, and M. Lahijanian, "MAPS-X: Explainable multi-robot motion planning via segmentation," in *Int't Conf. on Robotics and Automation*. IEEE, May 2021.

[4] S. M. LaValle *et al.*, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[5] I. A. Sucan and L. E. Kavraki, "A sampling-based tree planner for systems with complex dynamics," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 116–131, 2011.

[6] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Batch informed trees (bit*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 3067–3074.

[7] S. Tang and V. Kumar, "A complete algorithm for generating safe trajectories for multi-robot teams," in *Robotics Research*. Springer, 2018, pp. 599–616.

[8] R. Stern, N. R. Sturtevant, A. Felner, S. Koenig, H. Ma, T. T. Walker, J. Li, D. Atzmon, L. Cohen, and T. S. Kumar, "Multi-agent pathfinding: Definitions, variants, and benchmarks," in *12th SoCS*, 2019, pp. 151–158.

[9] 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.

[10] J. Li, A. Felner, E. Boyarski, H. Ma, and S. Koenig, "Improved heuristics for multi-agent path finding with conflict-based search." in *IJCAI*, vol. 2019, 2019, pp. 442–449.

[11] J. Kottinger, S. Almagor, and M. Lahijanian, "Conflict-based search for explainable multi-agent path finding," in *Int'l Conf. on Automated Planning and Scheduling*. AAAI Press, 2022.

[12] I. Solis, J. Motes, R. Sandström, and N. M. Amato, "Representation-optimal multi-robot motion planning using conflict-based search," *IEEE Robot. and Auto. Letters*, vol. 6, no. 3, pp. 4608–4615, 2021.

[13] W. Hönig, J. A. Preiss, T. S. Kumar, G. S. Sukhatme, and N. Ayanian, "Trajectory planning for quadrotor swarms," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 856–869, 2018.

[14] D. Le and E. Plaku, "Multi-robot motion planning with dynamics via coordinated sampling-based expansion guided by multi-agent search," *IEEE Robot. and Auto. Letters*, vol. 4, no. 2, pp. 1868–1875, 2019.

[15] L. Wen, Y. Liu, and H. Li, "CL-MAPF: Multi-agent path finding for car-like robots with kinematic and spatiotemporal constraints," *Robotics and Autonomous Systems*, vol. 150, p. 103997, 2022.

[16] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *Trans. on Robot. and Auto.*, vol. 12, no. 4, pp. 566–580, 1996.

[17] D. Le and E. Plaku, "Cooperative multi-robot sampling-based motion planning with dynamics," *ICAPS*, vol. 27, no. 1, pp. 513–521, 2017.

[18] J. Kottinger, S. Almagor, and M. Lahijanian, "Conflict-based search for multi-robot motion planning with kinodynamic constraints," 2022. [Online]. Available: https://arxiv.org/abs/2207.00576

[19] E. Boyarski, A. Felner, R. Stern, G. Sharon, D. Tolpin, O. Betzalel, and E. Shimony, "ICBS: Improved conflict-based search algorithm for multi-agent path finding," in *24th IJCAI*, 2015.

[20] G. Sharon, R. Stern, A. Felner, and N. Sturtevant, "Meta-agent conflict-based search for optimal multi-agent path finding," in *SoCS*, vol. 3, no. 1, 2012.

[21] E. Boyrasky, A. Felner, G. Sharon, and R. Stern, "Don't split, try to work it out: Bypassing conflicts in multi-agent pathfinding," in *ICAPS*, vol. 25, 2015, pp. 47–51.

[22] L. S. M, *Planning algorithms*. Cambridge university press, 2006.

[23] J. Kottinger, "Kinodynamic conflict-based search," https://github.com/aria-systems-group/K-CBS, 2022.