Efficient Multi-Robot Motion Planning for Manifold-Constrained Manipulators by Randomized Scheduling and Informed Path Generation

Weihang Guo ⬭, Zachary Kingston ⬭, *Member, IEEE*, Kaiyu Hang ⬭, and Lydia E. Kavraki ⬭, *Fellow, IEEE*

Abstract—**Multi-robot motion planning for high degree-of-freedom manipulators in shared, constrained, and narrow spaces is a complex problem and essential for many scenarios such as construction, surgery, and more. Traditional coupled methods plan directly in the composite configuration space, which scales poorly; decoupled methods, on the other hand, plan separately for each robot but lack completeness. Hybrid methods that obtain paths from individual robots together require the enumeration of many paths before they can find valid composite solutions. This paper introduces Scheduling to Avoid Collisions (StAC), a hybrid approach that more effectively composes paths from individual robots by *scheduling* (adding stops and coordination motion along all paths) and generates paths that are likely to be feasible by using bidirectional feedback between the scheduler and motion planner for informed sampling. StAC uses 10 to 100 times fewer paths from the low-level planner than state-of-the-art hybrid baselines on challenging problems in manipulator cases.**

Index Terms—**Multi-robot systems, constrained motion planning, motion and path planning, collision avoidance.**

Fig. 1. We extend the classic planar doorway problem to high-DoF manipulators. The end effectors of the two arms are constrained to a manifold on the same plane. In this scenario, two robots must swap positions of their end effector by navigating a narrow passage, which forces one to wait. As shown in Section IV-A, the StAC algorithm solves this challenge in an average of 5 seconds, whereas the state-of-the-art hybrid method fails to find a solution within 60 seconds.

I. INTRODUCTION

MULTI-ROBOT systems are essential for solving tasks beyond the capabilities of a single robot [1]. Multi-Robot Motion Planning (MRMP) finds continuous, collision-free paths for multiple robots, considering collisions not just with obstacles but also between moving robots. In MRMP, the approaches can be categorized into three main types: decoupled, coupled, and hybrid methods. Decoupled methods, such as those based on velocity obstacles [2] or priority-based planning [3], are primarily applied to mobile and aerial robots. However, they are ill-suited for high-DoF manipulators operating in constrained,

cluttered environments, where the mapping from workspace to configuration space is high-dimensional and complex, and the configuration space contains many narrow passages. Consequently, they often face deadlocks and are inefficient in capturing the required coordination. This paper focuses on the constrained and cluttered scenarios, and we consider robotic arms with manifold constraints, which limit their motion to a surface, such as a planar work surface (see Fig. 1). In such complex cases, coupled and hybrid methods are more widely used due to their ability to handle the intricate coordination required.

Coupled methods plan directly in the composite configuration space by modeling the multi-robot system as a single entity. Algorithms such as dRRT [4] extend the principles of classic RRT [5], sampling composite configurations that jointly encode the states of all robots. dRRT often scales poorly, making it unsuitable for complex scenarios involving high-DoF manipulators in confined spaces (see Section IV). Hybrid approaches [6], [7] combine the strengths of coupled and decoupled methods by using a low-level planner to generate paths individually and a high-level planner to schedule them. The process typically involves two main components. The first component is *planning:* each robot plans a path from its start to its goal, avoiding obstacles and ignoring other robots. The second component is *scheduling:* based on these planned trajectories, the execution is scheduled by adjusting velocities or introducing pauses to ensure that no inter-robot collisions occur during motion.

A less-explored aspect of hybrid approaches is effective scheduling. Methods such as CBSMP [7] emphasize enumerating paths over exploring alternative schedules. CBSMP, for

Weihang Guo is with the Department of Computer Science, Rice University, Houston, TX 77005 USA (e-mail: wg25@rice.edu).

Zachary Kingston is with the Department of Computer Science, Rice University, Houston, TX 77005 USA, and also with the Department of Computer Science, Purdue University, West Lafayette, IN 47907-2107 USA (e-mail: zkingston@purdue.edu).

Kaiyu Hang and Lydia E. Kavraki are with the Department of Computer Science, Rice University, Houston, TX 77005 USA, and also with Ken Kennedy Institute, Rice University, Houston, TX 77005 USA (e-mail: kaiyu.hang@rice.edu; kavraki@rice.edu).
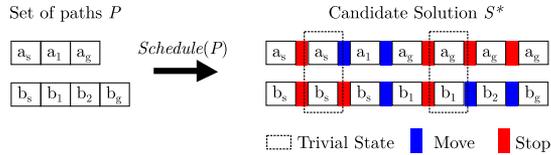
Fig. 2.   An illustration of SCHEDULE(P). Robots a and b are in the environment with their start a_s, b_s and goal a_g, b_g. Paths returned by the low-level planners are obtained from the PRM (see Remark 1). SCHEDULE(P) uses a set of individual robot paths to generate a candidate solution. The candidate solution uses the *same set* of vertices as the original paths, but may repeat certain configurations to represent *stops*. To transition from composite state i to $i + 1$, each robot will either move or stop. If all robots stop, then state $i + 1$ is *trivial* and will be removed.

example, checks collisions only under a fixed schedule (typically simultaneous motion) and, upon conflict, immediately regenerates paths. This neglects many viable schedules, such as allowing one robot to move first, which could resolve conflicts without re-planning the individual robot trajectory. In constrained settings like Fig. 1, where only one arm can pass through a narrow doorway at a time, simultaneous motion forces long detours that may be infeasible. By contrast, effective scheduling increases the likelihood of finding solutions and offers richer feedback to low-level planners.

To this end, we introduce the Scheduling to Avoid Collisions (StAC) algorithm, a probabilistically complete hybrid approach designed to emphasize and explore the importance of scheduling in MRMP. This method is specifically tailored to high-DoF robots operating in manifold-constrained, narrow, cluttered, and shared workspaces. StAC's high-level scheduler generates randomized schedules that coordinate the robot's motions to avoid collisions for paths generated by individual path planners. If the high-level planner cannot schedule a valid solution, collision information from all schedules is given to the low-level planners, who use this feedback to plan alternative paths. Our empirical results demonstrate that StAC requires 10 to 100 times fewer paths from the low-level planner to find a solution than CBSMP, significantly reducing planning times in highly constrained scenarios where state-of-the-art hybrid methods fail to solve even once.

II. PRELIMINARIES AND RELATED WORK

The Multi-Robot Motion Planning (MRMP) problem involves determining feasible continuous paths for a set of robots $A = \{a_1, \ldots, a_I\}$ operating in an environment W. The composite state space of all robot configurations is known as the configuration space $\mathbb{C}$, the Cartesian product of all individual robot configuration spaces $\mathbb{C}_1 \times \cdots \times \mathbb{C}_I$. Here, $\mathbb{C}_i$ represents the configuration space for robot $a_i \in A$.

We are interested in the case where each robot is *task constrained*, e.g., they must keep their end-effector on a planar work surface. We represent these constraints as *manifold constraints* and use notation from [8]: the constraint imposed on robot a_i is defined as the function $f_i : \mathbb{C}_i \to \mathbb{R}^k$, which results in the implicit submanifold $\mathbb{M}_i \subset \mathbb{C}_i$ where $f_i(q) = \mathbf{0}, q \in \mathbb{M}_i$. A review of methods for planning the motion of a single manifold-constrained robot is given in [9]. The composite manifold of

all constraint-satisfying configurations is given as $\mathbb{M} = \mathbb{M}_1 \times \cdots \times \mathbb{M}_I$.

The path of a_i is a continuous motion within a subset of $\mathbb{M}_i$ called the free space of a_i, denoted as $\mathbb{M}_{free,i}$. The composite constraint-satisfying free space $\mathbb{M}_{free} \subseteq \mathbb{M}$ consists of configurations in which the robots do not collide with obstacles or pairwise with each other, and satisfy constraints. Each robot a_i has an initial configuration $c_{s,i}$ and a set of goals $G_i \subset \mathbb{M}_{free,i}$. The objective is to find a continuous path with a schedule $\rho_i(s)$: $[0, 1] \to \mathbb{M}_{free,i}$ for each robot a_i that transitions it from $c_{s,i}$ to G_i without colliding with any obstacles $o \in W$ and all other robots $a_j \in A \backslash \{a_i\}$. Initially, this path is parameterized linearly over time, meaning $s(t) = \frac{t}{t_{final}}$ for $t \in [0, t_{final}]$, where t_{final} is the total time allocated for the robot to complete its motion from start to goal. To adjust the timing and coordination among robots, we introduce a scheduling function $\sigma_i(t) : [0, t_{final}] \to [0, 1]$ that modifies the linear mapping. The actual motion of the robot is then defined as $\rho_i(t) = p_i(\sigma_i(t))$, where $\sigma_i(t)$ adjusts the progression along the path p_i.

A. Coupled Methods

Sampling-based motion planners (e.g., RRT [5], PRM [10]) can solve MRMP by treating the multi-robot system as one composite system. Dedicated methods plan in factored representations, such as dRRT [4] and variants [11], which efficiently sample from individual robots' configuration spaces and search in the composite space. dRRT uses a prioritization rule to schedule the robots to connect two vertices in the tree. SSSP [12] iteratively builds local search spaces over individual robot roadmaps; here, all robots are scheduled to move simultaneously. The planner in [13] uses workspace topology and convex relaxation to coordinate more robots. However, these methods do not extend directly to high-DoF cases due to the costly workspace to configuration space transformations and the complexity of the composite space.

B. Decoupled Methods

Decoupled approaches address the exponential complexity of MRMP by decomposing the problem into many single-robot problems, e.g., by treating other robots as velocity obstacles and moving in simultaneously [2], [14], [15]. The velocity obstacles are computationally costly to determine in the configuration space of a high-degree-of-freedom manipulator robot. Other algorithms, such as [3], [16], [17], schedule robots in a prioritized order, where each robot plans in sequence and treats previously selected robots as dynamic obstacles. Resolving deadlocks in multi-robot motion planning using prioritized methods requires careful design of the prioritization strategy. Prior work [18] has shown that fixed-order prioritization is incomplete for multi-robot planning. Deadlock-free prioritized methods, such as [18], [19], [20], have only been demonstrated for mobile-robot settings, where the mapping between the workspace and configuration space is straightforward and solving the single-robot planning problem is trivial. Many decoupled algorithms are difficult to generalize to high-DoF cases due to unknown workspace-to-configuration space projections

and differing state representations for each manipulator. Many learning-based methods [16], [21] also require data and training and do not readily generalize.

C. Hybrid Methods

Hybrid methods address MRMP by combining high- and low-level planners. The low-level planner generates paths for individual robots (or subsets), while the high-level planner schedules execution, checks feasibility, and provides conflict feedback to guide re-planning. Graph-based methods such as CBS [6] and LaCAM [22] iteratively refine paths by adding constraints to avoid collisions, with CBS variants [23], [24] improving efficiency to handle hundreds of robots. However, CBS and MA-CBS operate on grid-based state representations, whereas our problem is defined in a continuous state space. MMD [25] incorporates inter-robot conflicts as conditions to the diffusion policy, which requires significant data to train.

CBSMP [7] extends CBS to continuous spaces by using roadmaps for low-level planning. Later work [13] outperforms the CBSMP in a planar robot setup, and their topology-guided strategy does not directly extend to a multi-arm setup. We therefore regard CBSMP as the state-of-the-art method for multi-arm planning in narrow passages and adopt it as our baseline. The high-level planner of CBSMP discretizes paths into uniform time segments, advancing all agents simultaneously. If collisions occur, the paths are discarded and replanned. However, many possible schedules exist to time a set of robot paths; for example, consider cars moving on the road, where traffic lights can coordinate the cars by asking them to stop. Unlike StAC, CBSMP only explores one possible scheduling function of the paths.

D. Conflict Resolution and Path Scheduling

Given a set of robot paths, execution can be coordinated by velocity tuning [26], [27], [28], which assigns each robot a velocity profile, or by priority-based search [3], which executes robots in a fixed order. Some works [22], [29] incorporate stopping actions during path search in *discrete* spaces, but these strategies do not directly extend to continuous settings. In contrast, our scheduling explicitly reorders motion timing across robots, enabling more flexible coordination. The work in [30] similarly reschedules around conflict areas, using priority ordering and reverting to a composite PRM in failure cases. Nevertheless, in high-DoF contexts, subproblems *still* involve planning for multiple manipulators, which remains challenging. The method in [31] precomputes collision pairs with continuous time intervals to efficiently find collision-free schedules, but it applies only to 2D roadmaps where neighbor searches are efficient. Finally, [32] uses a learning-based scheduling approach, which could also be integrated into the StAC framework to further improve performance.

III. SCHEDULING TO AVOID COLLISIONS

In this section, we introduce the core idea of Scheduling to Avoid Collisions (StAC) and elaborate on its details in the following subsections.

Algorithm 1: The StAC Algorithm.

Input: Set of robots A, Maximum reschedule attempts N_{RA}, Maximum replan attempts N_B

Output: Solution S

1:   **while** time available **do**
2:       Initial motion plans $P \leftarrow \emptyset$
3:       **for** each robot $a_i \in A$ **do**
4:           $p_i \leftarrow a_i.\text{QUERYPRM}()$
5:           $P \leftarrow P \cup \{p_i\}$
6:       **end for**
7:       $n \leftarrow 0$
8:       **while** $n \leq N_B$ **do**   ▷Section III-E
9:           Attempt $i \leftarrow 0$; Collision Record $record \leftarrow \emptyset$; $n \leftarrow n + 1$
10:          **while** $i \leq N_{RA}$ **do**   ▷Iterative schedule same set of paths
11:              Candidate Solution $S^* \leftarrow \text{SCHEDULE}(P)$ ▷Section III-B
12:              Collision Information $x \leftarrow \text{COLLISIONCHECK}(S^*)$
13:              **if** $x = \emptyset$ **then**
14:                  **return** $S \leftarrow S^*$ ▷Find the valid scheduling
15:              **end if**
16:              $\text{RECORDCOLLISION}(x, record)$; $i \leftarrow i + 1$ ▷Section III-C
17:          **end while**
18:          $P \leftarrow \emptyset$ ▷Cannot find a valid scheduling.
19:          **for** each robot $a_i \in A$ **do**
20:              $a_i.\text{UPDATE}(p_i, record[i])$ ▷Update the feedback to robots
21:              $p_i \leftarrow a_i.\text{QUERYPRMWITHEXPERIENCE}()$ ▷Section III-D
22:              $P \leftarrow P \cup \{p_i\}$
23:          **end for**
24:      **end while**
25: **end while**
26: **return** $S \leftarrow \emptyset$

A. Core Idea

StAC consists of the scheduler and the individual low-level motion planner $a_i \in A$ for a set of A robots. The pseudocode of StAC is given in Algorithm 1. Initially, each low-level planner i plans a path $p_i(s)$ from its start to goal in its own manifold-constrained configuration space $\mathbb{M}_i$ by calling $a_i.QueryPRM()$, which assumes the robot i is the only robot in the environment.

Remark 1 (Individual low-level motion planner): Each robot constructs its own projection-based manifold-constrained PRM [8], assuming it is the only robot in the environment. A path, p, is then obtained as a sequence of vertices in this PRM, with its **length**, $|p|$, defined by the number of vertices. Throughout this paper, when we refer to the **path vertex** or **path edge** of a robot, we specifically mean the corresponding vertex or edge in the PRM.

Given a set of paths P returned by the all robots' low-level planners, the scheduler computes *schedules* $\sigma_1(t), \ldots, \sigma_A(t)$ for each $p_i \in P$ such that no collisions occur between robots. This process, which we call *coordination space scheduling* (Section III-B). As illustrated in Fig. 2, SCHEDULE(P) first generates a candidate solution S^* by inserting random pauses along individual robot paths and assigning a random priority order for their progression to the next vertex (Section III-B). A collision check is then performed on S^*. If S^* is collision-free with respect to all obstacles and other robots, it becomes the solution. Otherwise, the scheduler records all collision edges between robots in the *Collision Record* (Algorithm 1). The scheduler then iteratively reschedules for a new candidate solution using the same set of paths until it reaches the maximum rescheduling attempts (N_{RA}) (Section III-C).

If a valid solution is not found after N_{RA} attempts, the collision counts of all edges stored in the Collision Record are provided to the robots as feedback (Algorithm 1). Each robot's low-level planner then stores this collision information in a *Collision History*. Based on this experience, each robot plans a different set of paths, P, using QUERYPRMWITHEXPERIENCE(), biased towards paths that will avoid collisions by favoring edges that have had fewer collisions with other robots in the past (Section III-D).

B. Scheduling in Coordination Space

The robot i's path p_i is a sequence of vertices, each corresponding to a robot configuration (see Remark 1) in $\mathbb{M}_{free,i}$. All vertices in path p_i form a set $\mathbb{P}_i \subseteq \mathbb{M}_{free,i}$. The scheduler takes the set of paths P from each robot's PRM as input. The goal of the scheduler is to find a continuous inter-robot collision-free paths schedule for all robots $\rho_i(t) : [0, t_{final}] \to \mathbb{P}_i$ for $i \in \{1 \ldots I\}$. Thus, given any time $t \in [0, t_{final}]$, $\rho_i(t)$ represents robot i's configuration $c_i \in \mathbb{P}_i$. By doing this, we constrain the range of individual scheduling functions to $\mathbb{P}_i$, which has a dimension of 1. The space of all scheduling functions for the composite system with a given set of robot paths P is called the *coordination space* of P. We used a sampling-based method to add *random* stops and schedule the paths. StAC samples stops between the vertices. We define $|p_i|$ equal to the number of vertices of path p_i.

Definition 1 (Candidate solution length): Given a set of robot paths P, we define the length of a candidate solution as $L := \sum_{p_i \in P} |p_i|$, which equals the total number of vertices across all robot paths.

We discretize the time into L steps with all robots starting at their initial state at time $t_1 = 0$ and reaching their goal state at $t_L = t_{final}$. From t_i to t_{i+1}, the robot will either *move* from its current vertex to the next vertex or *stop*. We sample $L - |p_i|$ stops for each robot to determine when it remains stationary. If all robots stop, then t_{i+1} is a trivial state and will be removed. We call the path with a scheduling function a candidate solution S^*. The length of S^*, $|S^*|$, equals the number of non-trivial time steps. We provide an example in Fig. 2.

C. Collision Record for Robot Feedback

For the given set of paths P, we attempt to find a valid schedule within N_{RA} times, as shown in Algorithm 1. We use a *Collision*
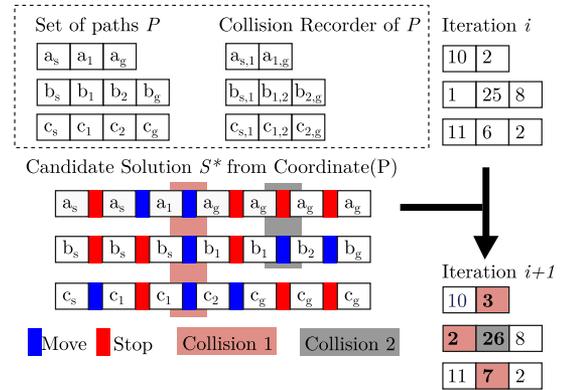

Fig. 3. An example of the Collision Record during scheduling three robots. The Collision Record values for iteration i are randomly initialized for illustration purposes. During each iteration, collision checking is performed on the candidate solution, and the counter for any edge involved in a collision is incremented by one. If a robot *stops*, the collision is not counted. For example, a collision between a_g and $b_{1,2}$ in the grey region only increments the counter for $b_{1,2}$ from 25 to 26, since robot a is stopped.

Record to log inter-robot collisions that happen during the N_{RA} attempts. As shown in Fig. 3, each path maintains a Collision Record with a length that matches the number of edges in the path. Whenever a collision occurs between edges, the counters for both edges are incremented by one. Unlike CBS-based planners, the Collision Record does not record the time t at which a collision occurs. It stores only geometric information. This design effectively decouples scheduling from planning.

D. Motion Planning With Experience

If no solution is found by SCHEDULE(P) after N_{RA} iterations, StAC sends the Collision Record back to each low-level planner. Each robot maintains a *Collision History* list to store all paths and their corresponding Collision Records. The low-level planner uses this history as a reference to guide the selection of new paths. The objective of each robot's low-level planner is to identify areas with a high likelihood of inter-robot collisions using the Collision History and to generate diverse paths to avoid these regions.

Specifically, the Collision History is initialized and expanded as follows. It is initialized with a *root* element containing a null path and an empty Collision Record. All Collision Records from the scheduler are stored in the Collision History. Each time QUERYPRMWITHEXPERIENCE() is called, the planner first selects an element from the Collision History based on a weighted sample. For an element e in the Collision History, the cost of the element is assigned by a user-customized function $C(e)$. Here, we use $C(e) = l \cdot c(e) + k \cdot s(e)$, where $c(e)$ is the path cost in configuration-space distance, $s(e)$ is the number of times this node has already been selected. Constants l and k adjust these contributions. Specifically, the sampler intends to select paths with smaller costs and fewer selections. $C(e)$ is always greater than 0.

If the root element is selected, the robot will not identify any high-collision area and return QUERYPRM(). If a *non-root* element is selected, a random-walk algorithm is used to identify the high likelihood of inter-robot collisions region, represented as an
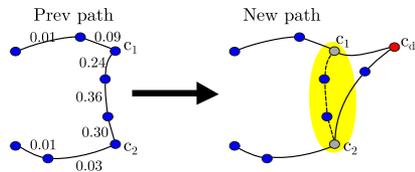
Fig. 4. Low-level planners will plan a new path using the previous path in the Collision Record. The number near each edge is the normalized Collision Record of the edge and is randomly initialized for illustration purposes. The yellow region is the collision region determined by the random walk algorithm with c_1 and c_2 as its end vertices. c_d is the detour configuration. The new path is shown in the solid line.
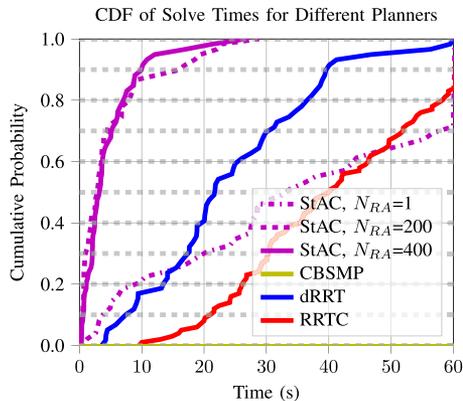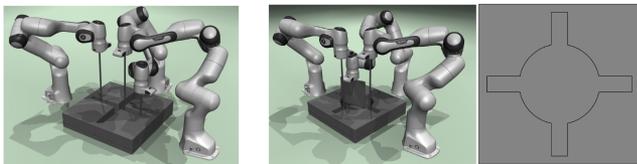

Fig. 5. CDF of solve time for the manipulator doorway setup with two 7-DOF Franka Panda arms shown in Fig. 1. Note that CBSMP cannot solve this problem even once, as indicated by the horizontal line at 0.

(a) Case 1: 3-manipulator setup with a cross maze **(b)** Case 2: 3-manipulator setup with circular arrangement. Maze top view is shown on the right.

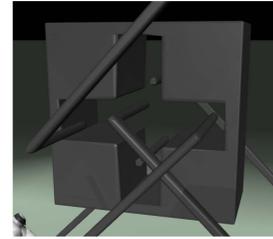Fig. 6. Two different three-manipulator setups.

Fig. 7. The multi-manipulator setups constrain the end effector to the yz-plane within the cross area. Each arm is staggered along the x-axis, so the ends of the S-shaped end effector will collide.

path to form $(c_s, \ldots, c_1, \ldots, c_d, \ldots, c_2, \ldots, c_g)$, returned by QUERYPRMWITHEXPERIENCE().

E. Re-Schedule

As stated in Section III-C, for a given set of paths P, we schedule the path maximum N_{RA} times and record all inter-robot collisions. The parameter N_{RA} allows users to trade off between scheduling and planning. When $N_{RA} = 1$, StAC resembles CBSMP, where paths are scheduled only once. However, it differs from CBSMP in the following ways. (1) robot priority is determined randomly (by adding stops) rather than pre-defined. (2) Robot paths are scheduled N_{RA} times rather than only once. (3) Time at which collisions occur is decoupled from the low-level planner, making the planner purely geometric. (4) Conflicts are treated as soft constraints (because of the random walk algorithm in Section III-D) instead of hard constraints as in CBS. Additionally, when the low-level planner calls QUERYPRMWITHEXPERIENCE(), its PRM is frozen for efficiency, meaning no new vertices are added. This may hurt the probabilistic completeness of our algorithm. To address this, the low-level planner rebuilds the PRM and clears the Collision History after it has called QUERYPRMWITHEXPERIENCE() N_B times.

F. Probabilistic Completeness

The proof of probabilistic completeness for the StAC algorithm has two parts. First, we show that given a set of paths P, we eventually find a collision-free solution by calling SCHEDULE(P) with infinitely many N_{RA} if such a solution exists. Second, given infinitely many attempts for QUERYPRMWITHEXPERIENCE(), the low-level planner of each robot will return all possible paths.

Lemma 1: Let P be a set of robot paths. If solution S exist from scheduling robot paths P, then the length $|S| \leq \sum_{p_i \in P} (|p_i| - 1)$

Proof: If a solution $\dot{S}$ without trivial states had length $> \sum_{p_i \in P} (|p_i| - 1)$, at least one robot moves per step. But the longest sequential solution has length $\sum_{p_i \in P} (|p_i| - 1)$, a contradiction. Hence, $|S| \leq \sum_{p_i \in P} (|p_i| - 1)$. ∎

Theorem 1: Given an infinite number of reschedule attempts $N_{RA} \to \infty$, SCHEDULE(P) will eventually try every possible valid scheduling of robot paths P. Consequently, if a solution S exists in P, it will be found.

interval along the element's path. The random-walk algorithm begins by selecting a random vertex from the element's Collision Record. From this vertex, the algorithm performs k steps, visiting the left or right vertex with probabilities proportional to the neighboring edge's collision counts. For example, the probability of moving left is computed as the number of left-edge collisions divided by the sum of left-edge and right-edge collisions. During the walk, the algorithm tracks the minimum and maximum value vertices visited. Those vertices are further mapped back to the selected element's PRM path, illustrated by the yellow region, (c_1, c_2), in Fig. 4. Because the walk is biased by the number of collisions, it preferentially selects regions with frequent inter-robot conflicts and ensures every interval retains a non-zero probability of being chosen.

The low-level planner reroutes around high-collision areas by randomly sampling a detour vertex c_d in the PRM, which must be included in the path. As shown in Fig. 4, it plans from c_1 to c_d and c_d to c_2, then merges these segments with the original

Problem	Method		# of Queries			Coord.	Succ.
	Planner	N_{RA}	Q1	Median	Q3	ratio	
Case 1 (Fig. 6a)	CBSMP	-	1	1	26	11.5%	79.3%
	dRRT	-	-	-	-	-	75.7%
	RRTC	-	-	-	-	-	28.9%
	StAC	1	1	1	29	20.3%	86.0%
		200	1	1	10	39.1%	**89.7%**
		400	1	1	10	37.5%	86.3%
Case 2 (Fig. 6b)	CBSMP	-	29	73	282	1.4%	42.0%
	dRRT	-	-	-	-	-	19.0%
	RRTC	-	-	-	-	-	22.2%
	StAC	1	32	174	270	15.4%	47.3%
		200	3	32	99	54.2%	**64.0%**
		400	2	32	78	65.6%	61.7%

Proof: Assume, for contradiction, that a valid scheduling S' of P is never returned by SCHEDULE(P), even as $N_{RA} \to \infty$. Let l_i be the length of robot i's path and $m = \sum_{p_i \in P} |p_i|$. There are $C(l_i, m)$ ways to schedule each path.

The probability of selecting S' in one iteration is $\prod_{i \in A} \frac{1}{C(l_i, m)}$, so the probability it is not selected is $1 - \prod_{i \in A} \frac{1}{C(l_i, m)}$. Thus, as $N_{RA} \to \infty$, the probability that S' is never selected approaches zero:

$$\lim_{N_{RA} \to \infty} P(\text{not } S') = \lim_{N_{RA} \to \infty} \left(1 - \prod_{i \in A} \frac{1}{C(l_i, m)} \right)^{N_{RA}}$$
$$= 0.$$

This contradicts the assumption, so SCHEDULE(P) eventually returns every valid scheduling of P. ∎

We now show that each robot has a non-zero chance of returning all different paths (including those with loops), to compensate for the monotonicity of scheduling.

Theorem 2: Let robot i's low-level planner be probabilistically complete. Then, as the number of iterations $n \to \infty$, the probability that the planner returns a path ϵ-close to any valid path p approaches 1.

Proof: Fix any valid path $p = \{c_s, c_1, \ldots, c_g\}$ and any $\epsilon > 0$. By probabilistic completeness of the low-level planner, the probability of sampling a detour point within ϵ of c_1 is nonzero. Repeating this argument inductively for $c_2, c_3, \ldots, c_g$, each step of extending the path within ϵ has nonzero probability.

Let q denote an ϵ-close approximation of p. The probability that q is not constructed in n iterations is bounded by $(1 - \delta)^n$, where $\delta > 0$ is the minimum probability of extending the path within ϵ at each step. As $n \to \infty$, $(1 - \delta)^n \to 0$.

Thus, for any valid path p, the probability that the planner produces an ϵ-close path tends to 1 as $n \to \infty$. ∎

Theorem 3: Given an infinite number of iterations $t \to \infty$ and an infinite maximum number of reschedule attempts $N_{RA} \to \infty$, StAC will find a solution if one exists.

Proof: According to Theorem 2, the probability that a robot returns a path ϵ-close to any valid path, including those with loops, approaches 1 as $t \to \infty$. Consequently, StAC will, with probability 1, eventually obtain ϵ-close approximations of every possible combination of robot paths as $t \to \infty$.

As stated in Theorem 1, for each possible scheduling of paths, StAC will find a solution if one exists when $t \to \infty$. Therefore, given infinite iterations, the StAC is guaranteed to identify a solution if one exists. ∎
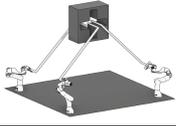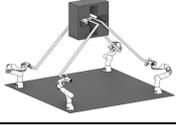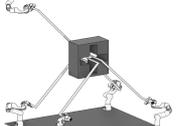
IV. EXPERIMENTAL RESULTS

We test StAC on three setups: (1) the extended 2D doorway problem with constrained Panda arms (Section IV-A), (2) three-manipulator coordination tasks (Section IV-B), and (3) a cross maze with 2–5 arms to evaluate scalability (Section IV-C). All algorithms are implemented in C++ in OMPL [33] with manifold constrained planning [8] and tested on a PC with an Intel i5-8365 1.6 GHz CPU and 8 GB of RAM. We utilized the MuJoCo physics engine [34] for collision detection in all experiments. We measure the end-to-end solving time after MuJoCo and the OMPL problem definition have been initialized. We used CBSMP [7], dRRT [4], and RRT-Connect [5] as our comparison baseline. Since the source code for CBSMP and dRRT is not available, we implemented them in OMPL and added support for planning under manifold constraints. All robots are subject to end-effector constraints. To efficiently generate samples on the manifold, we first sample the end-effector pose and compute the corresponding configurations using inverse kinematics. During interpolation, the algorithm projects intermediate configurations onto the constraint manifold based on [8]. We benchmarked different map sizes and the number of expansions before connecting to the goal for dRRT, selecting the parameters with the best performance. Additionally, we used the same set of PRM parameters for both CBSMP and StAC. Unless otherwise stated, we set $N_B = 10$ for StAC in all experiments.

A. 3D Doorway Setup

This setup involves a 3D doorway setup where two 7-DoF Franka Panda arms, each with an end stick, are tasked with swapping their end effector positions as shown in Fig. 1. The arms' end effectors are constrained to move only on the xy-plane within the maze. Each algorithm was run 60 times with a 60-second timeout. We evaluated the performance of dRRT, RRT-Connect, CBSMP, and StAC, and the cumulative distribution function (CDF) of the solve times is shown in Fig. 5. StAC demonstrates superior performance compared to other state-of-the-art algorithms, while CBSMP fails to solve the problem even once. This result further highlights the power of scheduling over path enumeration, as the robot has more possible configurations to avoid collisions. dRRT has the shorter average path length of 5.56, whereas StAC has a length of 7.03.

TABLE II

We Randomly Generated 8 Start-Goal Pairs Per Problem and Tested Each on Three Algorithms, Running Each Pair 10 Times With a 60s Timeout, Yielding 80 Results Per Algorithm. We Report the First Quartile (Q1), Median, and Third Quartile (Q3). Here, T is Total Solve Time, Q is the Number of Path Sets Queried by the High-Level Planner, and *Coord. Ratio* is the Ratio Between the Scheduling Time and the Total Time. *:As Mentioned in Section III-E, We Also Define a Maximum Number of Attempts Before Reconstructing the PRMs. By Default, This Maximum is 10, But for 400* it is Increased to 20.

Problem	Method		Planning Statistics						Coord.	Succ.
			Q1		Median		Q3		ratio	
	Planner	N_{RA}	T	Q	T	Q	T	Q		
	dRRT	-	0.016	-	0.042	-	0.068	-	-	**100.0%**
	CBSMP	-	0.015	1	0.099	14	0.160	11	10.3%	**100.0%**
	StAC	1	0.015	1	0.021	1	0.029	1	38.6%	**100.0%**
		200	**0.013**	1	**0.017**	1	0.023	1	36.5%	**100.0%**
		400	0.014	1	**0.017**	1	**0.022**	1	34.0%	**100.0%**
		400*	0.015	1	0.018	1	0.028	1	4.3%	**100.0%**
	dRRT	-	1.398	-	2.794	-	5.242	-	-	**100.0%**
	CBSMP	-	0.996	31	2.906	145	8.162	315	3.1%	94.4%
	StAC	1	0.086	1	0.737	8	2.459	22	35.1%	**100.0%**
		200	**0.057**	1	**0.085**	1	**0.177**	1	84.7%	**100.0%**
		400	0.061	1	0.092	1	0.192	1	87.2%	**100.0%**
		400*	0.058	1	0.098	1	0.249	1	91.3%	**100.0%**
	dRRT	-	7.926	-	37.143	-	T/O	-	-	67.1%
	CBSMP	-	0.524	31	13.009	516	T/O	2041	5.5%	68.9%
	StAC	1	0.561	6	5.078	60	T/O	412	58.6%	72.2%
		200	**0.075**	1	0.621	2	28.174	26	92.7%	82.2%
		400	0.098	1	1.326	3	**23.586**	20	97.6%	**84.4%**
		400*	0.083	1	**0.313**	1	T/O	17	94.6%	73.3%
	dRRT	-	T/O	-	T/O	-	T/O	-	-	-
	CBSMP	-	6.960	274	23.347	745	T/O	1813	4.4%	**65.6%**
	StAC	1	54.344	308	T/O	349	T/O	385	53.4%	25.6%
		200	26.911	11	T/O	22	T/O	28	97.7%	30.0%
		400	28.761	11	T/O	16	T/O	19	96.0%	33.3%
		400*	T/O	11	T/O	14	T/O	15	97.3%	20.0%

B. Three-Manipulator Setup

Three Franka Panda manipulators plan motions in clustered environments with two setups (Fig. 6): (1) all manipulators have stick-shaped end effectors constrained to planar translation, with one shorter than the others, and (2) two manipulators have stick-shaped end effectors constrained to the plane, and the third manipulator has a paddle-shaped end effector that can also rotate and translate in a central circular region. We test six start-goal pairs with a 60 s timeout for Case 1 and nine pairs with a 120 s timeout for Case 2, each repeated 50 times. CBSMP fails on one pair in Case 1 and two in Case 2, while StAC with $N_{RA} = 200$ achieves the highest success rate (Table I). For case 1, dRRT has the shortest average path length of 0.57, and CBSMP and StAC are comparable with 0.73 and 0.83. For case 2, dRRT has the shortest average path length of 0.76, and CBSMP and StAC have lengths of 1.10 and 1.53 respectively.

C. Multi-Manipulator Setups

To test scalability, we evaluate StAC on a cross maze with end effectors constrained to the yz-plane and staggered along the x-axis so only the s-shaped tips collide (Fig. 7, Table II). We generated eight start-goal pairs, each run ten times, yielding 80 trials per algorithm (dRRT, CBSMP, and StAC). Among the 8 sampled queries, some instances are easier than others. For

these easier instances, our coordinator queries each individual motion planner only once to obtain a valid path and finds a feasible schedule. This is why the solve time for the first quartile of the four-arm problem is 0.075 s. In contrast, CBSMP must perform extensive re-planning before resolving conflicts; for the same case, it issues 31 motion-planning queries before finding a solution. With 2-4 arms, StAC finds schedules within about ten queries, but beyond 30 DoF, scheduling becomes the bottleneck.

D. Insight and Lessons

Our experiments show that incorporating feedback and path scheduling significantly improves multi-robot motion planning success rates and computational efficiency. By leveraging path scheduling, robots exploit paths better. StAC queries fewer paths to find the solution. CBSMP outperforms our method when the number of arms increases to five in Section IV-C. We attribute this difference to how the two approaches scale in their scheduling components. CBSMP can struggle in tight environments, such as the 3D Doorway (Section IV-A) and Three-Manipulator problems (Section IV-B), because resolving deadlocks often requires substantial expansion of the constraint tree. In contrast, our algorithm handles deadlocks more efficiently through repeated rescheduling and the use of soft constraints. However, in our method, the number of sampled

stop actions grows linearly with the total path length across all robots, resulting in increased scheduling overhead as the team size grows. CBSMP's scheduling effort, by comparison, is largely independent of the number of robots, since it evaluates only a single continuous-time execution without incorporating explicit per-robot stop actions.

V. Conclusion

This paper presented the Scheduling to Avoid Collisions (StAC) algorithm for multi-robot motion planning, demonstrating its effectiveness in manipulator environments with manifold constraints. With iterative path rescheduling and an informed path generation mechanism, StAC efficiently reduces the complexity of finding collision-free solutions in shared, clustered workspaces with manifold constraints. StAC balances the search between low-level planners and the high-level scheduler. This novel high-level scheduling allows StAC to query significantly fewer paths from the low-level planner to find a valid solution. Future work includes automating the selection of rescheduling attempts, exploiting both CPU and GPU parallelism, and extending the method to optimal path planning.

Acknowledgment

References

[1] A. Khamis, A. Hussein, and A. Elmogy, *Multi-robot Task Allocation: A Review of the State-of-the-Art*. Cham, Switzerland: Springer International Publishing, 2015, pp. 31–51, doi: 10.1007/978-3-319-18299-5_2.

[2] J. Van Den Berg, J. Snape, S. J. Guy, and D. Manocha, "Reciprocal collision avoidance with acceleration-velocity obstacles," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2011, pp. 3475–3482.

[3] J. P. Van Den Berg and M. H. Overmars, "Prioritized motion planning for multiple robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2005, pp. 430–435.

[4] 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," *Int. J. Robot. Res.*, vol. 35, pp. 501–513, 2013. [Online]. Available: https://api.semanticscholar.org/CorpusID:152780

[5] J. Kuffner and S. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proc. ICRA. Millennium Conf. IEEE Int. Conf. Robot. Automat. Symposia Proc.*, 2000, vol. 2, pp. 995–1001.

[6] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artif. Intell.*, vol. 219, pp. 40–66, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0004370214001386

[7] J. I. Solis Vidana, J. Motes, R. Sandstrom, and N. Amato, "Representation-optimal multi-robot motion planning using conflict-based search," *IEEE Trans. Robot. Autom.*, vol. 6, no. 3, pp. 4608–4615, Jul. 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9387143/

[8] Z. Kingston, M. Moll, and L. E. Kavraki, "Exploring implicit spaces for constrained sampling-based planning," *Int. J. Robot. Res.*, vol. 38, no. 10–11, pp. 1151–1178, Sep. 2019.

[9] Z. Kingston, M. Moll, and L. E. Kavraki, "Sampling-based methods for motion planning with constraints," *Annu. Rev. Control, Robot., Auton. Syst.*, vol. 1, no. 1, pp. 159–185, 2018.

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

[11] R. Shome, K. Solovey, A. Dobson, D. Halperin, and K. E. Bekris, "dRRT*: Scalable and informed asymptotically-optimal multi-robot motion planning," *Auton. Robots*, vol. 44, no. 3, pp. 443–467, 2020, doi: 10.1007/s10514-019-09832-9.

[12] K. Okumura and X. Défago, "Quick multi-robot motion planning by combining sampling and search," in *Proc. 32nd Int. Joint Conf. Artif. Intell.*, Aug. 2023, pp. 252–261.

[13] C. McBeth, J. Motes, D. Uwacu, M. Morales, and N. M. Amato, "Scalable multi-robot motion planning for congested environments with topological guidance," *IEEE Robot. Automat. Lett.*, vol. 8, no. 11, pp. 6867–6874, Nov. 2023.

[14] S. H. Arul and D. Manocha, "V-RVO: Decentralized multi-agent collision avoidance using voronoi diagrams and reciprocal velocity obstacles," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 8097–8104.

[15] T. Pan, C. K. Verginis, A. M. Wells, L. E. Kavraki, and D. V. Dimarogonas, "Augmenting control policies with motion planning for robust and safe multi-robot navigation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 6975–6981.

[16] W. Li, H. Chen, B. Jin, W. Tan, H. Zha, and X. Wang, "Multi-agent path finding with prioritized communication learning," in *Proc. Int. Conf. Robot. Automat.*, 2022, pp. 10695–10701.

[17] A. Orthey, S. Akbar, and M. Toussaint, "Multilevel motion planning: A fiber bundle formulation," *Int. J. Robot. Res.*, vol. 43, no. 1, pp. 3–33, 2024.

[18] H. Ma, D. Harabor, P. J. Stuckey, J. Li, and S. Koenig, "Searching with consistent prioritization for multi-agent path finding," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, no. 1, 2019, pp. 7643–7650.

[19] M. Čáp, P. Novák, A. Kleiner, and M. Seleck, "Prioritized planning algorithms for trajectory coordination of multiple mobile robots," *IEEE Trans. Automat. Sci. Eng.*, vol. 12, no. 3, pp. 835–849, Jul. 2015.

[20] X. Zhang, G. Xiong, Y. Wang, S. Teng, and L. Chen, "D-PBS: Dueling priority-based search for multiple nonholonomic robots motion planning in congested environments," *IEEE Robot. Automat. Lett.*, vol. 9, no. 7, pp. 6288–6295, Jul. 2024.

[21] Q. Li, W. Lin, Z. Liu, and A. Prorok, "Message-aware graph attention networks for large-scale multi-robot path planning," *IEEE Robot. Automat. Lett.*, vol. 6, no. 3, pp. 5533–5540, Jul. 2021.

[22] K. Okumura, "LaCAM: Search-based algorithm for quick multi-agent pathfinding," in *Proc. AAAI Conf. Artif. Intell.*, vol. 37, no. 10, Jun. 2023, pp. 11655–11662. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/26377

[23] M. Barer, G. Sharon, R. Stern, and A. Felner, "Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem," in *Proc. Int. Symp. Combinatorial Search*, vol. 5, no. 1, 2014, pp. 19–27.

[24] J. Lim and P. Tsiotras, "CBS-budget (CBSB): A complete and bounded suboptimal search for multi-agent path finding," *Artif. Intell.*, vol. 346, 2025, Art. no. 104349.

[25] Y. Shaoul, I. Mishani, S. Vats, J. Li, and M. Likhachev, "Multi-robot motion planning with diffusion models," in *Proc. 13th Int. Conf. Learn. Representations*, 2025. [Online]. Available: https://openreview.net/forum?id=AUCYptvAf3

[26] G. Sanchez and J.-C. Latombe, "Using a PRM planner to compare centralized and decoupled planning for multi-robot systems," in *Proc. 2002 IEEE Int. Conf. Robot. Automat.*, vol. 2, 2002, pp. 2112–2119.

[27] K. Kant and S. W. Zucker, "Toward efficient trajectory planning: The path-velocity decomposition," *Int. J. Robot. Res.*, vol. 5, pp. 72–89, 1986. [Online]. Available: https://api.semanticscholar.org/CorpusID:120059732

[28] T. Simeon, S. Leroy, and J.-P. Lauumond, "Path coordination for multiple mobile robots: A resolution-complete algorithm," *IEEE Trans. Robot. Automat.*, vol. 18, no. 1, pp. 42–49, Feb. 2002.

[29] A. Wiktor, D. Scobee, S. Messenger, and C. Clark, "Decentralized and complete multi-robot motion planning in confined spaces," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2014, pp. 1168–1175.

[30] I. Solis, J. Motes, M. Qin, M. Morales, and N. M. Amato, "Adaptive robot coordination: A subproblem-based approach for hybrid multi-robot motion planning," *IEEE Robot. Automat. Lett.*, vol. 9, no. 8, pp. 7238–7245, Aug. 2024.

[31] K. Kasaura, M. Nishimura, and R. Yonetani, "Prioritized safe interval path planning for multi-agent pathfinding with continuous time on 2D roadmaps," *IEEE Robot. Automat. Lett.*, vol. 7, no. 4, pp. 10494–10501, Oct. 2022.

[32] K. Okumura, R. Yonetani, M. Nishimura, and A. Kanezaki, "CTRMs: Learning to construct cooperative timed roadmaps for multi-agent path planning in continuous spaces," in *Proc. 21st Int. Conf. Auton. Agents Multiagent Syst.*, 2022, pp. 972–981.

[33] I. A. Şucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robot. Automat. Mag.*, vol. 19, no. 4, pp. 72–82, Dec. 2012. [Online]. Available: https://ompl.kavrakilab.org

[34] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 5026–5033.