Space-Time Graphs of Convex Sets for Multi-Robot Motion Planning

Jingtao Tang, Zining Mao, Lufan Yang and Hang Ma

***Abstract*— We address the Multi-Robot Motion Planning (MRMP) problem of computing collision-free trajectories for multiple robots in shared continuous environments. While existing frameworks effectively decompose MRMP into single-robot subproblems, spatiotemporal motion planning with dynamic obstacles remains challenging, particularly in cluttered or narrow-corridor settings. We propose Space-Time Graphs of Convex Sets (ST-GCS), a novel planner that systematically covers the collision-free space-time domain with convex sets instead of relying on random sampling. By extending Graphs of Convex Sets (GCS) into the time dimension, ST-GCS formulates time-optimal trajectories in a unified convex optimization that naturally accommodates velocity bounds and flexible arrival times. We also propose Exact Convex Decomposition (ECD) to "reserve" trajectories as spatiotemporal obstacles, maintaining a collision-free space-time graph of convex sets for subsequent planning. Integrated into two prioritized-planning frameworks, ST-GCS consistently achieves higher success rates and better solution quality than state-of-the-art sampling-based planners—often at orders-of-magnitude faster runtimes—underscoring its benefits for MRMP in challenging settings. Project page: https://sites.google.com/view/stgcs.**

I. Introduction

We study Multi-Robot Motion Planning (MRMP), where the problem is to compute collision-free trajectories that move multiple robots from given start to goal states in space-time, while avoiding collisions both with the environment and with each other. Recent developments in Multi-Agent Path Finding (MAPF) on discrete graphs have produced powerful frameworks that decouple MRMP into single-robot trajectory computations. However, when these single-robot planners must navigate "dynamic obstacles" (i.e., the trajectories of other robots), the resulting spatiotemporal motion planning problem remains challenging and underexplored.

Sampling-based planners, such as Rapidly-Exploring Random Trees (RRT) [1] and Probabilistic Roadmaps (PRM) [2], are popular for their simplicity and theoretical completeness. However, spatiotemporal motion planning introduces additional challenges that significantly degrade their effectiveness: Dynamic obstacles can open or close narrow corridors, and random sampling may fail to capture these brief windows of safe transit. Moreover, many sampling-based planners either discretize time coarsely or incrementally adjust time bounds, limiting both effectiveness and efficiency in seeking time-optimal solutions within an arbitrarily large time dimension. Additionally, repeated collision checks required for state expansions become prohibitively expensive when the time dimension is included. When applied to MRMP in a coupled manner, sampling-based planners face even greater challenges due to the curse of dimensionality, which severely restricts scalability as the concatenated state space grows rapidly with the number of robots.

[1]The authors are with the School of Computing Science, Simon Fraser University, Burnaby, BC V5A1S6, Canada. {jingtao_tang, zining_mao, lufan_yang, hangma}@sfu.ca.

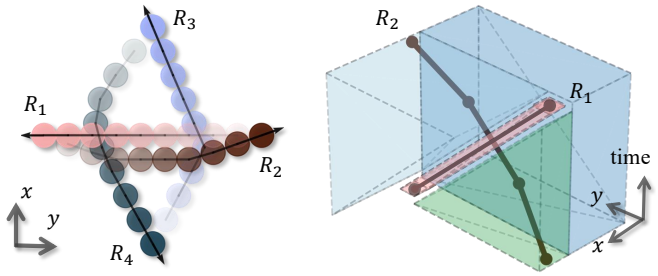


Fig. 1. Demonstration of the proposed PBS+ST-GCS for MRMP, where R_1 and R_3 exchange positions with R_2 and R_4, respectively. Left: Solution trajectories visualized in 2D spatial coordinates, with higher transparency indicating states at later time stamps. Right: Solution trajectories of R_1 and R_2 visualized in 3D space-time coordinates, where R_2 treats R_1 as a space-time obstacle and generates a trajectory through space-time collision-free convex sets (colored polyhedra) that exclude the trajectory of R_1.

In this paper, we propose Space-Time Graphs of Convex Sets (ST-GCS), a novel time-optimal motion planner that significantly improves MRMP solving. ST-GCS offers a fundamentally different, deterministic approach by systematically covering the entire collision-free space-time region with convex sets, inherently capturing spatiotemporal bottlenecks and avoiding the pitfalls of random sampling. ST-GCS extends Graphs of Convex Sets (GCS) [3]—originally designed for static, single-robot motion planning—into the spatiotemporal and multi-robot context. The key idea is to solve a generalized shortest-path problem on a graph whose vertices are convex sets, determining which sets form the path and the state within each set, which jointly optimizes a chosen objective function.

Algorithmic Contributions: (1) We present the key idea of ST-GCS by demonstrating how to augment a spatial decomposition (i.e., a graph of convex sets) with an arbitrarily large time dimension for time-optimal trajectories. By enforcing constraints on time flow and velocity bounds within a unified convex optimization, we obtain piecewise-linear, time-optimal trajectories without specialized time discretization. (2) We propose two MRMP methods by integrating ST-GCS into Random-Prioritized Planning (RP) [4], which randomly explores priority orders and plans each robot sequentially according to each priority order, and Priority-Based Search (PBS) [5], which searches over priority orders. In both frameworks, ST-GCS serves as the low-level trajectory planner once higher-priority trajectories are determined. (3) To

make ST-GCS tractable when higher-priority trajectories are already planned, we introduce the Exact Convex Decomposition (ECD) algorithm to partition the space-time convex sets so that subsequent planning does not conflict with these "reserved" trajectories. Fig. 1 illustrates how a robot's trajectory is generated by applying ST-GCS on convex sets that have been updated to incorporate reserved trajectories as obstacle regions.

Empirical Findings: We evaluate our MRMP methods against baselines that integrate a state-of-the-art sampling-based planner in a 2D mobile robot scenario. Our results demonstrate how restricting the collision-free region to space-time convex sets grants ST-GCS a unique advantage, especially in "narrow corridors" or crowded settings. In contrast, sampling-based methods require extensive exploration and often struggle to capture spatiotemporal safe transient windows, even when they are modified to sample directly in collision-free convex sets. Our methods consistently achieve higher success rates and better solution quality with orders of magnitude faster runtimes, highlighting the benefits of a deterministic convex-optimization approach to MRMP.

II. Related Work

We survey relevant research on MAPF, MRMP, and GCS.

Search-Based MAPF: MAPF [6], [7] is a prominent approach for multi-robot path planning on graphs (e.g., 2D grids [8] or state lattices [9]) typically assuming discrete time steps. Modern search-based MAPF methods have offered powerful bi-level frameworks that decompose a multi-robot planning problem into the high-level coordination (e.g., PP [4], Conflict-Based Search [10], or PBS [5]) that resolves collisions among individual trajectories and low-level single-robot trajectory computations (e.g., space-time A* [11]) that respects spatiotemporal constraints posed by the high level for collision resolution. Although some MAPF methods have been adapted to continuous-time robot actions [12]–[14], their solution quality remains constrained by the chosen discrete graph representation and limited motion primitives.

Sampling-Based MRMP: Sampling-based MRMP methods offer greater representational flexibility by requiring only a collision checker for the environment. Coupled sampling-based MRMP approaches [8], [15] plan in the joint state space of all robots but rely on synchronized robot actions to facilitate collision checks; they thus do not offer time optimality in general. Several recent spatiotemporal motion planners have been combined with PP [4] for MRMP. For example, Time-Based RRT [16] augments RRT [1] with a time dimension but assumes a fixed arrival time at the goal state. Temporal PRM [17] extends PRM [2] using safe time intervals [18] but relies on constant velocity magnitude, thus compromising solution quality. Space-Time RRT (ST-RRT*) [19] incorporates bidirectional tree search [20] into RRT* [21] and uses a specialized conditional sampler that progressively tightens the goal arrival time bound whenever a better feasible solution is found. Although ST-RRT* is asymptotically optimal with sufficient sampling, it shares a common limitation with other sampling-based planners: "narrow corridors" in the space-time state space remain difficult to sample and connect. Consequently, such methods can be effective in relatively open environments but may struggle in cluttered or heavily constrained instances.

GCS Applications: Techniques for constructing collision-free convex sets [22]–[24] have enabled GCS-based solutions to various robotics tasks, including single-UAV path planning in cluttered environments [25], non-Euclidean motion planning on Riemannian manifolds for mobile manipulators [26], and temporal-logic motion planning in high-dimensional systems [27]. Additionally, several search-based methods [28], [29] have been developed to improve the efficiency of GCS solving. Although [30] allows planning in the joint configuration space of two robotic arms with synchronous actions and [31] uses GCS solution to guide nonconvex trajectory optimization for dynamic environments, applying GCS to dynamic environments and multi-robot settings with asynchronous robot actions remains under-explored.

III. Space-Time Graphs of Convex Sets (ST-GCS)

In this section, we first present the GCS formulation for single-robot motion planning around static obstacles (Sec. III-A). We then introduce ST-GCS, which augments GCS with an explicit time dimension for time-optimal spatiotemporal motion planning (Sec. III-B).

A. GCS: Motion Planning in Time-Invariant State Spaces

We consider a single-robot motion planning problem in a d-dimensional state space whose collision-free region is decomposed into a given collection of convex sets (e.g., via different preprocessing techniques [22]–[24]). This decomposition is represented by a connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where each vertex $v \in \mathcal{V}$ corresponds to a convex set $\mathcal{X}_v = \{\mathbf{x} \in \mathbb{R}^d \mid A_v\mathbf{x} \preceq b_v\}$ and each edge $e = (u, v) \in \mathcal{E}$ indicates $\mathcal{X}_u \cap \mathcal{X}_v \neq \emptyset$. Since $\mathcal{G}$ is connected, a simple (acyclic) *path* $\pi = (v_1, v_2, ..., v_{|\pi|})$ exists between any two vertices v_1 and $v_{|\pi|}$, using the set of edges $\mathcal{E}(\pi) = \{(v_{i-1}, v_i)\}_{i=2}^{|\pi|} \subseteq \mathcal{E}$.

Following [30], we formulate a bilinear program for the single-robot motion planning problem over $\mathcal{G}$, which is then transformed into a mixed-integer convex program for solving. We introduce (1) a set of binary variables $\Phi = \{\phi_e\}_{e\in\mathcal{E}}$ that parameterizes any path π_Φ, with $\phi_e = 1$ if and only if $e \in \mathcal{E}(\pi_\Phi)$; and (2) two vectors of continuous variables $\mathbf{x}_v, \mathbf{y}_v \in \mathcal{X}_v$ that represent the robot's initial and terminal states within each convex set $\mathcal{X}_v$ for all $v \in \mathcal{V}$. For simplicity, we describe only the bilinear formulation below:

$$\min_{\Phi,\mathbf{x},\mathbf{y}} \quad \sum_{e\in\mathcal{E}(\pi_\Phi)} f(e) + \sum_{v\in\pi_\Phi} g(v) \tag{1}$$

$$\textbf{s.t.} \quad \mathcal{E}(\pi_\Phi) \subseteq \mathcal{E}, \tag{2}$$

$$\mathbf{x}_v, \mathbf{y}_v \in \mathcal{X}_v, \qquad \forall v \in \mathcal{V} \tag{3}$$

$$\mathbf{x}_v = \mathbf{y}_u, \qquad \forall e = (u, v) \in \mathcal{E} \tag{4}$$

$$\mathbf{x}_{v_{\text{start}}} = \mathbf{x}_{\text{start}}, \mathbf{y}_{v_{\text{goal}}} = \mathbf{x}_{\text{goal}}. \tag{5}$$

where $f(e)$ and $g(v)$ in Eqn. (1) specify additive costs over edges and vertices along the parametrized path π_Φ. Given a feasible solution that fixes the values of π_Φ, $\mathbf{x}$, and $\mathbf{y}$,

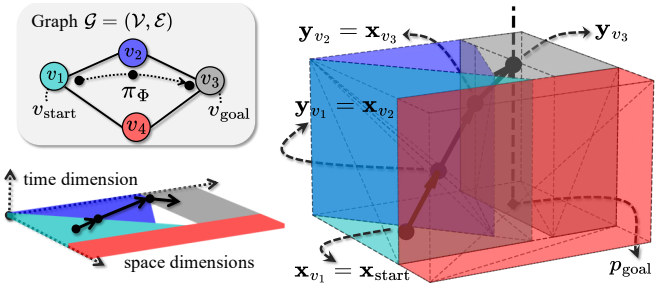


Fig. 2. Piecewise linear trajectories from $\mathbf{x}_{\text{start}}$ to p_{goal} through collision-free 2D space convex sets (lower-left) and 3D space-time convex sets (right).

we can reconstruct a continuous, collision-free *trajectory* τ by chaining the segments $(\mathbf{x}_v, \mathbf{y}_v)$ for each successive vertex in π_Φ. Specifically, Constraints Eqn. (2) enforces that π_Φ is a simple path that uses edges in $\mathcal{E}$, which can be achieved by introducing auxiliary flow conservation and degree constraints, commonly seen in linear program formulations for routing problems [32], rendering the above program nonlinear yet convex (see [3] for more details). Constraints Eqn. (3) enforce that the two states $\mathbf{x}_v, \mathbf{y}_v$ regarding each vertex $v \in \mathcal{V}$ must reside within the corresponding collision-free convex set $\mathcal{X}_v$, ensuring that the trajectory segment $(\mathbf{x}_v, \mathbf{y}_v)$ is also collision-free. Constraints Eqn. (4) enforce that the terminal state of u always coincides with the initial state of v for any edge (u, v) of π_Φ, ensuring that the reconstructed trajectory is continuous. Constraints Eqn. (5) ensure that τ starts from $\mathbf{x}_{\text{start}}$ and ends at $\mathbf{x}_{\text{goal}}$. Note that the two vertices (convex sets) $v_{\text{start}}, v_{\text{goal}} \in \mathcal{V}$ are determined by iterating through all vertices $v \in \mathcal{V}$ to check whether $\mathbf{x}_{\text{start}} \in \mathcal{X}_v$ or $\mathbf{x}_{\text{goal}} \in \mathcal{X}_v$, respectively.[1]

The above formulation aligns with standard MRMP conventions, focusing on kinematic feasibility and omitting differential and kinodynamic constraints. If needed, these constraints can be incorporated by augmenting the state space and adding linear or other convex constraints to capture, for example, bounded accelerations or nonholonomic motion.

B. ST-GCS: Time-Optimal Spatiotemporal Motion Planning

Although GCS effectively handles single-robot motion planning in a time-invariant d-dimensional state space, spatiotemporal motion planning with dynamic obstacles requires additional machinery to handle dynamic avoidance, variable arrival times, velocity bounds, and time optimality in a unified optimization. We thus propose ST-GCS, which operates on a graph of collision-free space-time convex sets.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ represent a space-time decomposition, where each vertex $v \in \mathcal{V}$ corresponds to a space-time convex set $\mathcal{X}_v \subset \mathbb{R}^{d+1}$ that is free of both static and dynamic obstacles. We construct these sets by (1) extruding each given spatial convex set (free of static obstacles) from time 0 to an arbitrarily large global time limit $t_{\max}$ and (2) applying ECD (see Sec. IV-A) to remove space-time regions intersecting other robots' trajectories (treated as dynamic obstacles), potentially subdividing the extruded sets further. An edge $(u, v) \in \mathcal{E}$ indicates $\mathcal{X}_u \cap \mathcal{X}_v \neq \emptyset$.

Let $\mathbf{x}.p$ and $\mathbf{x}.t$ respectively denote the spatial and temporal components of a space-time state $\mathbf{x}$. Within each space-time convex set $\mathcal{X}_v$, the local trajectory segment $\mathbf{x}_v \to \mathbf{y}_v$ is traversed at a uniform velocity $\mathbf{v} = \dfrac{\mathbf{y}_v.p - \mathbf{x}_v.p}{\mathbf{y}_v.t - \mathbf{x}_v.t}$, potentially different across sets. We now present the following ST-GCS formulation for spatiotemporal motion planning, which explicitly specifies a time-minimizing objective in Eqn. (6) and linear time and velocity constraints in Eqn. (8-9):

$$\min_{\Phi, \mathbf{x}, \mathbf{y}} \quad \sum_{v \in \pi_\Phi} (\mathbf{y}_v.t - \mathbf{x}_v.t) \tag{6}$$

$$\textbf{s.t.} \quad \text{Constraints in Eqn. (2-4)} \tag{7}$$

$$\mathbf{y}_v.t - \mathbf{x}_v.t > 0, \qquad \forall v \in \pi_\Phi \tag{8}$$

$$\mathbf{v}_{\min} \preceq \frac{\mathbf{y}_v.p - \mathbf{x}_v.p}{\mathbf{y}_v.t - \mathbf{x}_v.t} \preceq \mathbf{v}_{\max}, \qquad \forall v \in \pi_\Phi \tag{9}$$

$$\mathbf{x}_{v_{\text{start}}} = \mathbf{x}_{\text{start}}, \mathbf{y}_v.p = p_{\text{goal}} \qquad \text{with } v \in \mathcal{V}_{\text{goal}}. \tag{10}$$

where Constraints Eqn. (8) prevents time reversal from each state to a subsequent state [2]. Constraints Eqn. (9) impose given velocity bounds in each spatial dimension. Unlike Eqn. (5) in the static GCS formulation, Constraint Eqn. (10) only enforces that the spatial component matches the given goal position p_{goal}, leaving the arrival time unconstrained. Note that, while v_{start} is determined in the same way as in GCS, each goal vertex $v \in \mathcal{V}_{\text{goal}}$ can be identified by checking whether $\mathcal{X}_v$ contains any states with the goal position (i.e., $\mathcal{X}_v \cap \{(p_{\text{goal}}, t) \,|\, 0 \leq t \leq t_{\max}\} \neq \emptyset$) and, if so, whether any arrival time at the goal position within $\mathcal{X}_v$ can be extended to $t_{\max}$ (i.e., $\{(p_{\text{goal}}, t) \,|\, t^* \leq t \leq t_{\max}\} \subseteq \bigcup_{u \in \mathcal{V}} \mathcal{X}_u$, where $t^* = \min\{t \mid (p_{\text{goal}}, t) \in \mathcal{X}_v\}$) to ensure the robot can stay there indefinitely.

As shown in Fig. 2, given a feasible solution that fixes the values of π_Φ, $\mathbf{x}$, and $\mathbf{y}$, we can reconstruct a space-time piecewise linear trajectory, and any space-time state along the dashed line can serve as a valid goal if the robot can indefinitely stay at p_{goal} the arrival. In summary, ST-GCS fuses the core convex path parametrization of GCS with explicit time and velocity constraints, enabling time-optimal [3], piecewise-linear trajectories in a space-time domain that can include dynamic obstacles (handled via ECD).

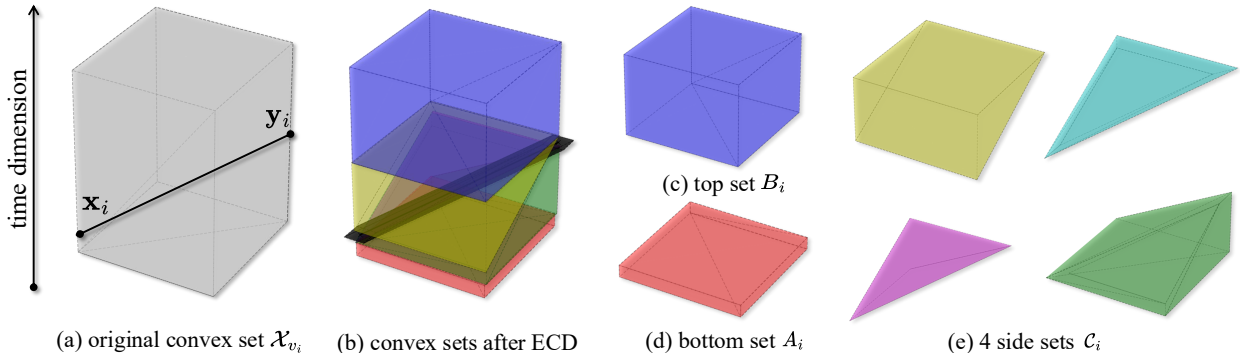
IV. ST-GCS For Multi-Robot Motion Planning

In this section, we consider MRMP with n robots, where each robot i is given a start state $\mathbf{x}^{(i)}_{\text{start}}$ and a goal position $p^{(i)}_{\text{goal}}$. The problem is to compute n space-time collision-free trajectories, $\mathcal{T} = \{\tau_i\}_{i=1}^n$, for the robots. Our approach relies on the Exact Convex Decomposition (ECD) algorithm (Sec. IV-A), which "reserves" a given piecewise linear trajectory on a graph $\mathcal{G}$ of space-time convex sets, thereby producing an updated graph $\mathcal{G}'$ whose convex sets are collision-free with respect to the reserved trajectory.

[1] In case of multiple v_{start} (or v_{goal}), a hyper vertex is created connecting itself to each v_{start} (or v_{goal}) [33] with slight changes to constraints Eqn. (5).

[2] In practice, we impose Eqn. (8) as $\mathbf{y}_v.t - \mathbf{x}_v.t \geq \epsilon$ with an inclusive lower bound of a small positive number ϵ.

[3] More rigorously, the time optimality is defined over the solution space of all piecewise linear trajectories parameterized by $(\Phi, \mathbf{x}, \mathbf{y})$.



(a) original convex set $\mathcal{X}_{v_i}$ (b) convex sets after ECD (c) top set B_i (d) bottom set A_i (e) 4 side sets $\mathcal{C}_i$

Fig. 3. The proposed ECD algorithm decomposes (a) convex set $\mathcal{X}_{v_i}$ for a vertex $v_i \in \mathcal{V}$ and a trajectory segment $\mathbf{x}_i \to \mathbf{y}_i$ into (c-e) 6 convex sets. (b) The trajectory segment is enlarged to a parallelepiped (highlighted in black) with its apothem being the safe radius between robots.

Consequently, once some robots' trajectories are planned, they can be treated as dynamic obstacles for the remaining robots by applying ECD and then running ST-GCS on $\mathcal{G}'$. Following standard MAPF practices, we solve MRMP via two prioritized planning frameworks (Sec. IV-B), i.e., Random-Prioritized Planning (RP) and Priority-Based Search (PBS), where robots plan one at a time while avoiding collisions with higher-priority robots' trajectories, which are incorporated as dynamic obstacles through ECD.

A. Exact Convex Decomposition (ECD)

ECD takes as input a sequence of vertex-segment tuples $\{(v_i, \mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^k$, where each trajectory segment $\mathbf{x}_i \to \mathbf{y}_i$ lies entirely in a single convex set $\mathcal{X}_{v_i}$ of the given graph $\mathcal{G}$. We will discuss how to obtain such a sequence from any given piecewise linear trajectory resulting from an ST-GCS solution at the end of this subsection. Here, we assume each robot has a d-dimensional hypercube occupancy centered at each state $\mathbf{x}$ and exemplify ECD in a 3D space-time state space, but the same principle can be extended to other convex-shaped occupancies and to higher-dimensional state spaces with an additional time dimension.

Pseudocode (Alg. 1): ECD iterates over the k vertex-segment tuples [Line 1]. To simplify notation, let $\boxdot(p, r)$ denote the 2D square centered at p with apothem r. For each tuple $(v_i, \mathbf{x}_i, \mathbf{y}_i)$, ECD first constructs a *parallelepiped* L_i with its top and bottom faces being squares $\boxdot(\mathbf{x}_i.p, r)$ and $\boxdot(\mathbf{y}_i.p, r)$, respectively (see Fig. 3(b)). ECD then creates two convex sets A_i and B_i [Lines 2-3], bounding L_i from above and below (see Fig. 3(b)). Specifically, B_0 excludes the cuboid formed by extruding $\boxdot(\mathbf{x}_0.p, r)$ from time 0 to $\mathbf{x}_0.t$, and A_k excludes the cuboid formed by extruding $\boxdot(\mathbf{x}_k.p, r)$ from time $\mathbf{x}_k.t$ to $t_{\max}$. This is crucial as it ensures the robot can safely stay at $\mathbf{x}_0.p$ during $t \in [0, \mathbf{x}_0.t)$ and at $\mathbf{x}_k.p$ during $t \in [\mathbf{x}_k.t, t_{\max})$. As shown in Fig. 3(e), after removing $A_i \cup B_i$ from $\mathcal{X}_{v_i}$, ECD partitions the remaining center region into four convex sets by intersecting it with each halfspace $\text{outside}(L_i, s)$ [Line 7], where s is each side face of L_i, considered as a separating plane, and $\text{outside}(L_i, s)$ is the halfspace that does not contain L_i (i.e., $\text{outside}(L_i, s) \cap L_i = \emptyset$). ECD then removes $\text{outside}(L_i, s)$ from the remaining region before processing the next halfspace [Line 8]. After partitioning $\mathcal{X}_{v_i}$ into these new convex sets, ECD adds a vertex corresponding to each new set to $\mathcal{G}$ [Line 9]. After processing all k tuples, ECD updates the edges of $\mathcal{G}$ by checking intersections among new neighboring sets [Line 10]. Fig. 3 shows an example for an intermediate tuple $(v_i, \mathbf{x}_i, \mathbf{y}_i)$ with $1 < i < k$.

Algorithm 1: ECD for Trajectory Reservation

Input: vertex-segment tuples $\{(v_i, \mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^k$, graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ of convex sets, safe radius r

1 **for** $i = 1, 2, ..., k$ **do**
2 $\quad A_i \leftarrow \{\mathbf{x} \in \mathcal{X}_{v_i} \mid \mathbf{x}.t \le \mathbf{x}_i.t\}$ ▷ Fig. 3(d)
3 $\quad B_i \leftarrow \{\mathbf{x} \in \mathcal{X}_{v_i} \mid \mathbf{x}.t \ge \mathbf{y}_i.t\}$ ▷ Fig. 3(c)
4 $\quad \mathcal{C}_i \leftarrow \{\}$, $\mathcal{X}_{v_i} \leftarrow \mathcal{X}_{v_i} \setminus (A_i \cup B_i)$
5 $\quad L_i \leftarrow$ the parallelepiped defined by $\mathbf{x}_i, \mathbf{y}_i, r$
6 $\quad$ **for** $s \in \{$*side faces of parallelepiped* $L_i\}$ **do**
7 $\quad\quad \mathcal{C}_i \leftarrow \mathcal{C}_i \cup \{\mathcal{X}_{v_i} \cap \text{outside}(L_i, s)\}$ ▷ Fig. 3(e)
8 $\quad\quad \mathcal{X}_{v_i} \leftarrow \mathcal{X}_{v_i} \setminus \text{outside}(L_i, s)$
9 $\quad$ add a vertex for every set in $\mathcal{C}_i \cup \{A_i, B_i\}$ to $\mathcal{G}$
10 update edges in $\mathcal{G}$ by checking set intersections

Vertex-Segment Sequence Construction: We describe how to construct a sequence $S = \{(v_i, \mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^k$ of vertex-segment tuples from a piecewise linear trajectory τ on $\mathcal{G}$. This construction is particularly relevant in our prioritized planning frameworks (Sec. IV-B) and for dynamic obstacles, where ECD reserves trajectories for collision avoidance. Three cases arise: (1) If τ is reconstructed from an ST-GCS solution computed on the same graph $\mathcal{G}$, each segment of τ naturally maps to a unique convex set $\mathcal{X}_v$ of $\mathcal{G}$, and potentially to the adjacent convex sets of $\mathcal{X}_v$ if the segment lies in the boundaries of $\mathcal{X}_v$. (2) If a segment of τ intersects more than one convex set of $\mathcal{G}$, we subdivide it at each boundary so that each sub-segment lies in exactly one convex set. (3) If a convex set $\mathcal{X}_v$ contains multiple segments $\mathbf{x}_j \to \mathbf{y}_j$ (non-overlapping in time), we slice $\mathcal{X}_v$ by the planes $t = \mathbf{x}_j.t$ and $t = \mathbf{y}_j.t$, ensuring each resulting subset contains at most one segment. We adopt a unified procedure to construct the sequence S, considering the above three cases. For each linear segment in τ, we check whether it

Algorithm 2: RP+ST-GCS for MRMP

Input: start and goal pairs$\{(\mathbf{x}_{\text{start}}^{(i)}, p_{\text{goal}}^{(i)})\}_{i=1}^{n}$, graph $\mathcal{G}$ of convex sets

1 **while** *not reaching the terminal condition* **do**
2 $\mathcal{G}' \leftarrow$ a copy of $\mathcal{G}$
3 **for** $i \in$ *random unused permutation of* $\{i\}_{i=1}^{n}$ **do**
4 $\tau_i \leftarrow \text{solve}(\mathcal{G}', \mathbf{x}_{\text{start}}^{(i)}, p_{\text{goal}}^{(i)})$
5 **if** *solving reports failure* **then**
6 **break**
7 $\mathcal{G}' \leftarrow$ reserve τ_i on $\mathcal{G}'$ via ECD
8 **if** *ST-GCS solving succeed for all robots* **then**
9 **return** $\{\tau_i\}_{i=1}^{n}$
10 **return** "*fail to find a solution*"

Algorithm 3: PBS+ST-GCS for MRMP

Input: start and goal pairs$\{(\mathbf{x}_{\text{start}}^{(i)}, p_{\text{goal}}^{(i)})\}_{i=1}^{n}$, graph $\mathcal{G}$ of convex sets

1 create a root node N_{root} w/ $\prec_{N_{\text{root}}} \leftarrow \emptyset$
2 $N_{\text{root}}.\mathcal{T} \leftarrow \{\text{solve}(\mathcal{G}, \mathbf{x}_{\text{start}}^{(i)}, p_{\text{goal}}^{(i)})\}_{i=1}^{n}$
3 Stack $\leftarrow \{N_{\text{root}}\}$
4 **while** *Stack* $\neq \emptyset$ **do**
5 $N \leftarrow$ Stack.pop()
6 **if** *no collisions in* $N.\mathcal{T}$ **then**
7 **return** $N.\mathcal{T}$
8 $\tau_i, \tau_j \leftarrow$ first pairwise collision in $N.\mathcal{T}$
9 **for** $(i,j) \in \{(i,j),(j,i)\}$ **do**
10 create node N' w/ $\prec_{N'} \leftarrow \prec_N \cup \{i \prec j\}$
11 **if** `UpdateNode`$(N', \mathcal{G}, j)$ **then**
12 Stack.push(N')
13 **return** "*fail to find a solution*"
14 **Function** `UpdateNode`$(N, \mathcal{G}, i)$**:**
15 $L \leftarrow \{i\} \cup \{j \,|\, 1 \le j \le n, i \prec_N j\}$
16 **for** $j \in topologicalSort(L, \prec_N)$ **do**
17 **if** $\exists k \prec_N j$ ***s.t.*** $N.\tau_k$ *collides w/* $N.\tau_j$ **then**
18 $\mathcal{G}' \leftarrow$ reserve $\{\tau_k | k \prec_N j\}$ on $\mathcal{G}$ via ECD
19 $N'.\tau_j \leftarrow \text{solve}(\mathcal{G}', \mathbf{x}_{\text{start}}^{(j)}, p_{\text{goal}}^{(j)})$
20 **if** *solving reports failure* **then**
21 **return False** ▷ fails to update N
22 **return True** ▷ succeeds to update N

intersects with each convex set of $\mathcal{G}$[4]. If intersecting, we collect the vertex v_i and the two endpoints $\mathbf{x}_i, \mathbf{y}_i$ of the intersecting sub-segment into sequence S.

B. Prioritized Planning Frameworks

We now introduce two prioritized planning frameworks for MRMP that use ST-GCS as the single-robot planner and ECD as a subroutine for trajectory reservation.

Random-Prioritized Planning (RP): RP (Alg. 2) explores random total priority orders (i.e., permutations of the robots). For each order, the robots plan sequentially [Line 3]. Each robot i plans its trajectory τ_i by solving ST-GCS on the current graph $\mathcal{G}'$ [Line 4], which is then updated with the planned τ_i reserved via ECD [Line 7]. RP returns the first feasible solution once all robots successfully plan their trajectories [Line 9]. Otherwise, it attempts the next order, until a terminal condition (e.g., runtime limit) is met.

Priority-Based Search (PBS): PBS (Alg. 3) systematically explores priority orders to resolve collisions by searching a priority tree, where each node N contains a unique priority set $\prec_N$ of ordered pairs of robots and a set $N.\mathcal{T}$ of n trajectories that respect the prioritized planning scheme specified by $\prec_N$. PBS initializes the root node with an empty priority set and potentially colliding trajectories [Lines 1-2]. When expanding a node N, PBS checks $N.\mathcal{T}$ for collisions [Line 6]. If none are found, then it returns $N.\mathcal{T}$ as a solution [Line 7]. Otherwise, it identifies a colliding pair τ_i and τ_j [Line 8] and generates two child nodes N_1 and N_2, adding the pair $i \prec j$ (i has a higher priority than j) to $\prec_{N_1}$ and $j \prec i$ to $\prec_{N_2}$ [Lines 9-10]. For each child node N', PBS invokes `UpdateNode` to replan the trajectories for a list of robots [Line 15] to ensure that all trajectories in $N'.\mathcal{T}$ respect $\prec_{N'}$. With all the high-priority trajectories reserved on $\mathcal{G}$ via ECD [Line 18], each replanning calls ST-GCS for a lower-priority robot j [Line 19]. If `UpdateNode` succeeds, PBS pushes the child node N' to the stack top [Lines 11-12]. PBS returns failure if no valid solution can be found after visiting all possible nodes in the priority tree [Line 13].

[4]Our implementation leverages bounding boxes for the convex sets to do coarse collision checking before exact collision checking for efficiency.

ST-GCS Solving: Solving ST-GCS to optimality can be expensive, especially after many ECD updates that enlarge the graph. Therefore, as a low-level planner for RP and PBS, we employ the convex-restriction and path-restriction heuristic approach from [30] rather than seeking global optimality [3]. In short, this approach first relaxes binary edge variables to fractional values, then heuristically reconstructs a graph path π_Φ by interpreting each fractional ϕ_e as the probability of using edge e. Fixing π_Φ yields a final convex program for ST-GCS that can be solved at a relatively low computational cost. Re-running this procedure multiple times with different random seeds can further improve solution quality, returning the best trajectory found.

V. Experiments

This section presents our experimental results on an *Apple*® M4 CPU machine with 16GB RAM. We evaluate **RP+ST-GCS** and **PBS+ST-GCS** against three baseline methods in 2D mobile robot domains. All methods are implemented in Python, and our ST-GCS program is solved using the *Drake* [34] library with the Mosek[5] solver. The source code and numerical results are publicly available on https://github.com/reso1/stgcs. More detailed visualizations and simulation videos of our approach can be found at https://sites.google.com/view/stgcs.

[5]https://www.mosek.com/

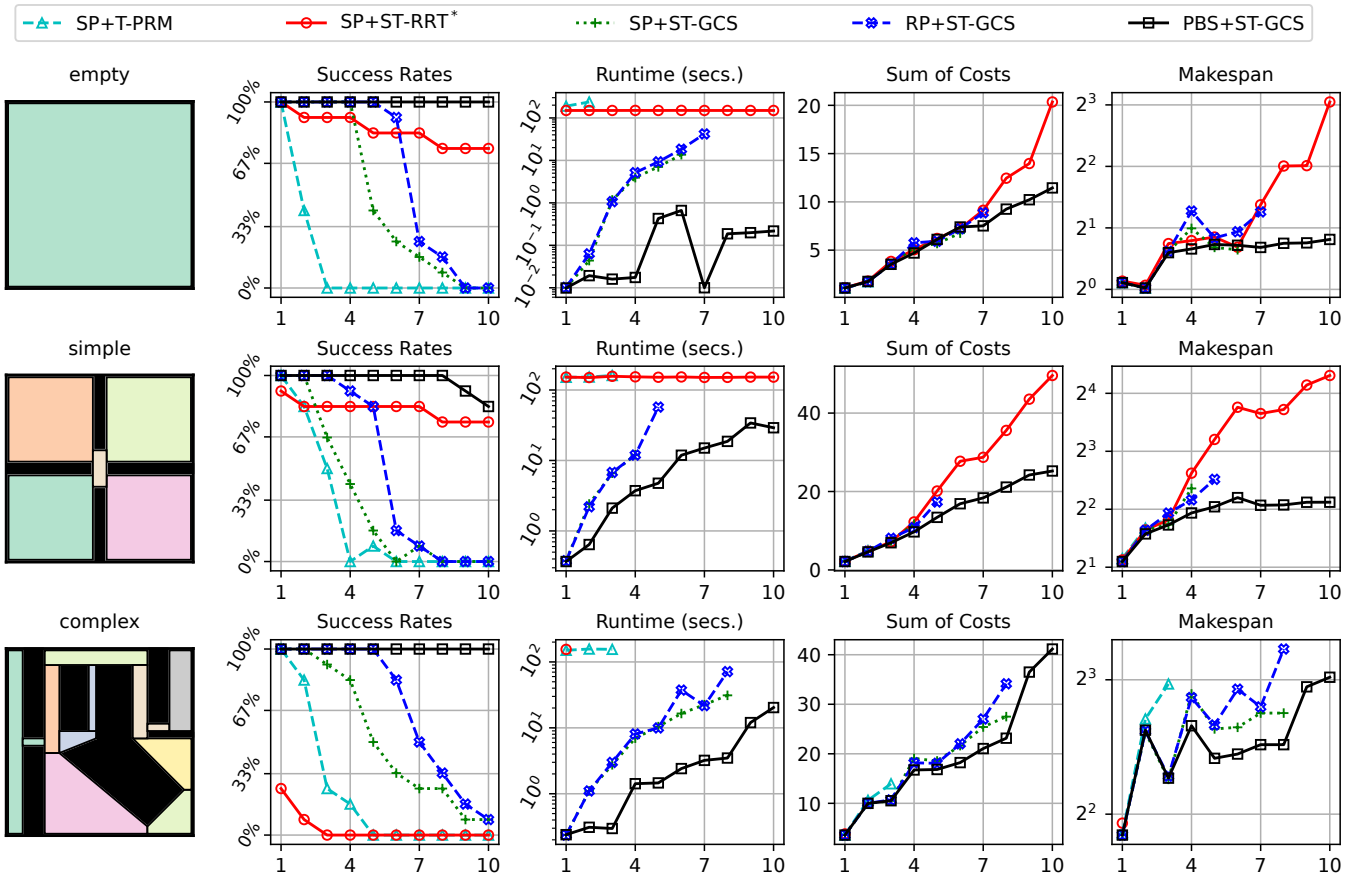


Fig. 4. Results for all MRMP methods on three maps. Each subplot shows the number of robots (x-axis) versus a specific metric (y-axis).

A. Experiment Setup

Instances: We use three benchmark 2D maps (Fig. 4): *empty*, *simple*, and *complex* [30]. Their spatial collision-free convex sets are given; we extrude each along time $t \in [0, 50]$ to create space-time convex sets. For *simple*, we additionally introduce four dynamic disk obstacles moving at constant velocities, then apply ECD to reserve their trajectories. For each map and each $n \in [1, 10]$ of robots, we generate 12 random instances by sampling start states and goal positions within the space-time convex sets.

Baselines: We compare against three sequential planning (SP) baselines—i.e., prioritized planning with the fixed priority order by robot index—each with a different low-level single-robot planner. (1) (Adapted) **SP+T-PRM** [17]: We adapt T-PRM to support collision checks on roadmap edges when finding shortest paths (missed by the original implementation) and restrict its random sampling to the given spatial convex sets. (2) (Adapted) **SP+ST-RRT*** [19]: We restrict the random sampling of ST-RRT* random sampling to the given spatial convex sets. (3) **SP+ST-GCS**.

Parameters: For *empty*, we set a maximum velocity limit of 0.5 for each space dimension; For *empty* and *complex*, we set a maximum velocity limit of 1.0 for each spatial dimension. We set a runtime limit of 150 seconds for all methods. SP+T-PRM allocates $150/n$ seconds for each robot's PRM construction (omitting its shortest-path finding runtime), and SP+ST-RRT* allocates $150/n$ seconds for each single-robot ST-RRT* planning. For ST-GCS solving on any graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ using the heuristic approach (see Sec. IV-B), we limit the number of graph paths sampled to $1e^3 \times \log|\mathcal{E}|$.

B. Results and Analysis

Fig. 4 reports four metrics: **Success Rate** (out of 12 random instances), **Runtime** (in seconds, on a **log-scaled** y-axis), **SoC** (the sum of time costs of all trajectories), and **Makespan** (the maximum time costs of all trajectories, on a **log2-scaled** y-axis). The last three metrics are averaged only over the intersection of instances solved by each method, ensuring a fair comparison. If a method solves fewer than 3 of 12 instances for a given n, it is excluded from that average to avoid empty intersections.

Success Rates and Runtimes: PBS+ST-GCS solves all instances on *empty* and *complex*, with average runtimes consistently under 1 second and 10 seconds on average, respectively—often orders of magnitude faster than sampling-based methods. Adding dynamic obstacles on *simple* significantly enlarges the initial space-time graph (via ECD), which can slow ST-GCS solving; nevertheless, PBS+ST-GCS still achieves the highest success rates overall. RP+ST-GCS randomly explores different priority orders until timeout generally attains higher success rates than SP+ST-GCS which uses only a fixed priority order.

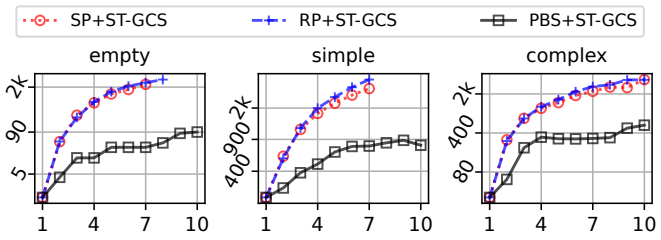


Fig. 5. Average number of graph edges in SP+ST-GCS, RP+ST-GCS, and PBS+ST-GCS when each returns a feasible solution.

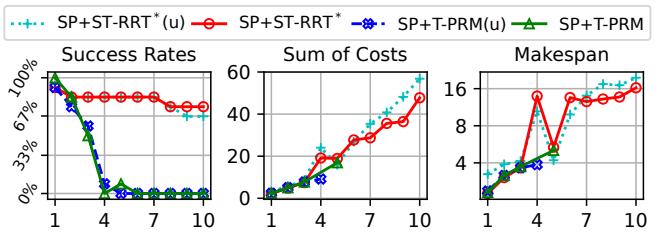


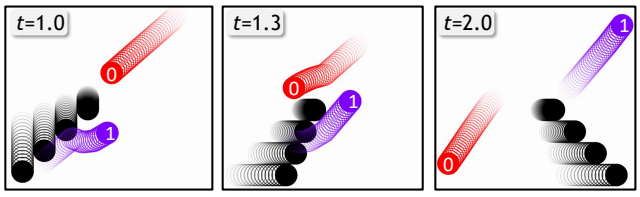
Fig. 6. Comparing the two sampling-based approaches with their sampling procedures performed on map bounding box and space collision-free sets.

Among sampling-based methods, SP+ST-RRT* outperforms SP+T-PRM on *empty* and *simple*, likely due to the inherent faster tree-based exploration rooted at the starts and goals. However, on *complex* with many spatial corridors, the more global random exploration of T-PRM yields higher success rates than ST-RRT*. Still, both fail when $n > 2$ on *complex*. Comparing all SP-based methods, SP+ST-GCS typically runs faster than the two sampling-based methods and achieves higher success rates in settings with more spatial corridors. However, its success rates are lower for larger n, due to the heuristic solver failing more frequently, as analyzed in the ablation study below.

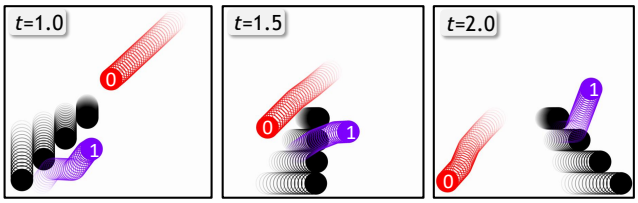
Solution Quality: PBS+ST-GCS consistently yields the best solution quality across all maps. Its SoC scales almost linearly with n, and its makespan barely increases as n grows. Other ST-GCS variants also produce better solutions than the sampling-based methods in many cases.

Ablation—Effectiveness of PBS for ST-GCS: Our raw data indicates that large space-time graphs (over 1000 edges) resulting from numerous dynamic obstacles can impede the heuristic solver under limited path-sampling budgets. Fig. 5 demonstrates that PBS mitigates this issue by reserving only conflicting trajectories deemed high-priority, controlling the graph size growth more effectively than SP+ST-GCS and RP+ST-GCS (which both rely on total priority orders with sequential ECD, thus expanding the graph faster as n grows).

Ablation—Constrained Sampling: Fig. 6 compares T-PRM and ST-RRT* with random sampling constrained to the spatial collision-free convex sets versus unconstrained sampling (labeled **T-PRM(u)** and **ST-RRT*(u)**) over the entire bounding box. Constrained sampling yields modest improvements in success rates (though T-PRM still fails for higher n) and better solution quality for ST-RRT*. This confirms that focusing sampling on collision-free regions can be beneficial, albeit insufficient to match ST-GCS performance.

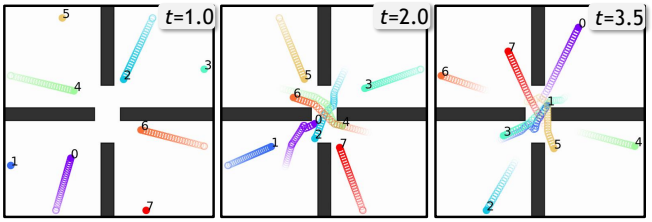


(a) PBS+ST-GCS: SoC = 4.80, makespan = 2.40, runtime = 8.34s

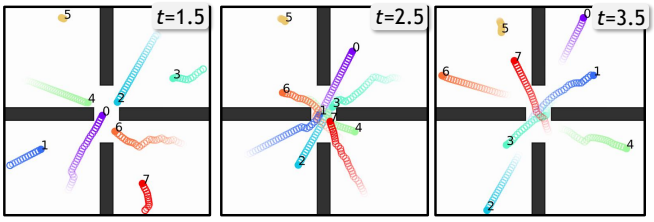


(b) SP+ST-RRT*: SoC = 5.23, makespan = 3.01, runtime = 150s

Fig. 7. Two robots exchange positions with four dynamic obstacles (black).



(a) PBS+ST-GCS: SoC = 26.4, makespan = 4.94, runtime = 55s



(b) SP+ST-RRT*: SoC = 29.3, makespan = 6.88, runtime = 150s

Fig. 8. Eight robots exchange positions with one another on *simple*.

C. Case Study

We highlight two examples demonstrating the advantages of PBS+ST-GCS over SP+ST-RRT*. Fig. 7 demonstrates that PBS+ST-GCS identifies a safe shortcut through dynamic obstacles, producing a time-optimal MRMP solution. Fig. 8 shows that PBS+ST-GCS enables robots to traverse the congested central region simultaneously with minimal waiting or detours, while SP+ST-RRT* generates zigzagging trajectories and significant delays (e.g., robot 5 is delayed until other robots nearly reach their goals).

VI. Conclusions & Future Work

We presented ST-GCS, a time-optimal deterministic approach that addresses the inherent limitations of sampling-based methods in spatiotemporal settings. By extending the GCS formulation to a space-time domain, ST-GCS systematically covers spatiotemporal bottlenecks using collision-free convex sets. Our ECD algorithm further enables straightforward reservation of piecewise linear trajectories of dynamic obstacles for collision avoidance. We integrated ST-GCS and ECD into prioritized planning frameworks for MRMP.

We demonstrated through extensive experiments that our approach consistently outperforms state-of-the-art sampling-based methods in both success rates and solution quality, often with orders-of-magnitude faster runtimes, especially in challenging scenarios such as narrow corridors and crowded environments. Future work includes investigating robust pruning strategies for managing the ST-GCS graph size when numerous trajectories are reserved via ECD, generalizing to high-dimensional nonlinear configuration spaces and more complex obstacle models, and extending ST-GCS with richer motion models with nonholonomic and dynamic constraints.

ACKNOWLEDGEMENT

This work was supported by the NSERC under grant number RGPIN2020-06540 and a CFI JELF award. We thank the anonymous reviewers for their constructive feedback that helped us improve this paper.

REFERENCES

[1] S. LaValle, "Rapidly-exploring random trees: A new tool for path planning," *Research Report 9811*, 1998.

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

[3] T. Marcucci, J. Umenberger, P. Parrilo, and R. Tedrake, "Shortest paths in graphs of convex sets," *SIAM Journal on Optimization*, vol. 34, no. 1, pp. 507–532, 2024.

[4] M. Erdmann and T. Lozano-Perez, "On multiple moving objects," *Algorithmica*, vol. 2, pp. 477–521, 1987.

[5] H. Ma, D. Harabor, P. J. Stuckey, J. Li, and S. Koenig, "Searching with consistent prioritization for multi-agent path finding," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 7643–7650.

[6] R. Stern, N. Sturtevant, A. Felner, S. Koenig, H. Ma, T. Walker, J. Li, D. Atzmon, L. Cohen, T. Kumar *et al.*, "Multi-agent pathfinding: Definitions, variants, and benchmarks," in *Proceedings of the International Symposium on Combinatorial Search*, vol. 10, no. 1, 2019, pp. 151–158.

[7] J. Tan, Y. Luo, J. Li, and H. Ma, "Reevaluation of large neighborhood search for mapf: Findings and opportunities," 2025. [Online]. Available: https://arxiv.org/abs/2407.09451

[8] 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," *The International Journal of Robotics Research*, vol. 35, no. 5, pp. 501–513, 2016.

[9] J. Yan and J. Li, "Multi-agent motion planning with bézier curve optimization under kinodynamic constraints," *IEEE Robotics and Automation Letters*, 2024.

[10] 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.

[11] D. Silver, "Cooperative pathfinding," in *Proceedings of the aaai conference on artificial intelligence and interactive digital entertainment*, vol. 1, no. 1, 2005, pp. 117–122.

[12] L. Cohen, T. Uras, T. Kumar, and S. Koenig, "Optimal and bounded-suboptimal multi-agent motion planning," in *International Symposium on Combinatorial Search*, 2019, pp. 44–51.

[13] A. Andreychuk, K. Yakovlev, P. Surynek, D. Atzmon, and R. Stern, "Multi-agent pathfinding with continuous time," *Artificial Intelligence*, vol. 305, p. 103662, 2022.

[14] J. Kottinger, S. Almagor, and M. Lahijanian, "Conflict-based search for multi-robot motion planning with kinodynamic constraints," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 13 494–13 499.

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

[16] A. Sintov and A. Shapiro, "Time-based rrt algorithm for rendezvous planning of two dynamic systems," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 6745–6750.

[17] M. Hüppi, L. Bartolomei, R. Mascaro, and M. Chli, "T-prm: Temporal probabilistic roadmap for path planning in dynamic environments," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 10 320–10 327.

[18] M. Phillips and M. Likhachev, "Sipp: Safe interval path planning for dynamic environments," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 5628–5635.

[19] F. Grothe, V. N. Hartmann, A. Orthey, and M. Toussaint, "St-rrt*: Asymptotically-optimal bidirectional motion planning through space-time," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 3314–3320.

[20] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 2. IEEE, 2000, pp. 995–1001.

[21] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[22] R. Deits and R. Tedrake, "Computing large convex regions of obstacle-free space through semidefinite programming," in *Algorithmic Foundations of Robotics XI: Selected Contributions of the Eleventh International Workshop on the Algorithmic Foundations of Robotics*. Springer, 2015, pp. 109–124.

[23] P. Werner, A. Amice, T. Marcucci, D. Rus, and R. Tedrake, "Approximating robot configuration spaces with few convex sets using clique covers of visibility graphs," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 10 359–10 365.

[24] H. Dai, A. Amice, P. Werner, A. Zhang, and R. Tedrake, "Certified polyhedral decompositions of collision-free configuration space," *The International Journal of Robotics Research*, vol. 43, no. 9, pp. 1322–1341, 2024.

[25] T. Marcucci, P. Nobel, R. Tedrake, and S. Boyd, "Fast path planning through large collections of safe boxes," *IEEE Transactions on Robotics*, 2024.

[26] T. Cohn, M. Petersen, M. Simchowitz, and R. Tedrake, "Non-euclidean motion planning with graphs of geodesically convex sets," *The International Journal of Robotics Research*, p. 02783649241302419, 2023.

[27] V. Kurtz and H. Lin, "Temporal logic motion planning with convex optimization via graphs of convex sets," *IEEE Transactions on Robotics*, 2023.

[28] S. Y. C. Chia, R. H. Jiang, B. P. Graesdal, L. P. Kaelbling, and R. Tedrake, "Gcs*: Forward heuristic search on implicit graphs of convex sets," *arXiv preprint arXiv:2407.08848*, 2024.

[29] R. Natarajan, C. Liu, H. Choset, and M. Likhachev, "Implicit graph search for planning on graphs of convex sets," *Robotics: Science and Systems (RSS)*, 2024.

[30] T. Marcucci, M. Petersen, D. von Wrangel, and R. Tedrake, "Motion planning around obstacles with convex optimization," *Science robotics*, vol. 8, no. 84, p. eadf7843, 2023.

[31] D. von Wrangel and R. Tedrake, "Using graphs of convex sets to guide nonconvex trajectory optimization," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 9863–9870.

[32] C. E. Miller, A. W. Tucker, and R. A. Zemlin, "Integer programming formulation of traveling salesman problems," *Journal of the ACM (JACM)*, vol. 7, no. 4, pp. 326–329, 1960.

[33] D. Bertsimas and J. N. Tsitsiklis, *Introduction to linear optimization*. Athena scientific Belmont, MA, 1997, vol. 6.

[34] R. Tedrake and the Drake Development Team, "Drake: Model-based design and verification for robotics," 2019. [Online]. Available: https://drake.mit.edu