Unified Branch-and-Bound Search for the Steiner Traveling Salesman Problem on Graphs of Convex Sets

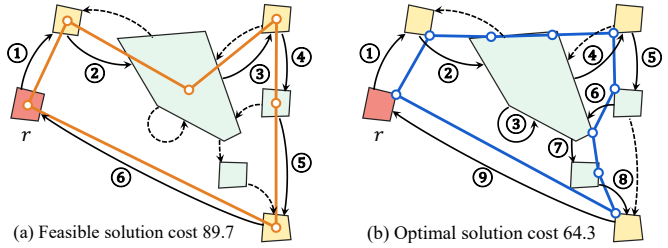
Jingtao Tang and Hang Ma

***Abstract*—We formalize the Steiner Traveling Salesman Problem (Steiner-TSP) on Graphs of Convex Sets (GCS), which seeks a minimum-cost closed trajectory through required convex sets while allowing optional transit vertices and revisits. To explore the resulting infinite solution space, we propose a unified branch-and-bound search over rooted walk prefixes. Additive lower-bound-graph costs bound committed prefixes, while a cut-separated connected-flow relaxation lower-bounds the residual cost of visiting every remaining target and returning to the root. Under a uniform positive-cost assumption, best-first traversal terminates after finitely many expansions on every feasible instance without an initial incumbent, whereas depth-first traversal does so once a finite incumbent is available. For a user-specified factor $\epsilon \geq 1$, a global lower bound certifies that either strategy's incumbent cost is at most ϵ times the global optimum. We further demonstrate joint sensing-mode, visitation-order, and continuous-trajectory selection for a mobile-manipulator inspection task, including action precedences expressed in linear temporal logic over finite traces (LTL_f). Both traversal strategies find feasible solutions on all benchmark instances within $30\,\text{s}$ with mean certified optimality gaps of 28.1% and 29.7%, respectively, whereas two recent baselines succeed on only about half of the instances.**

I. Introduction

Planning a minimum-cost trajectory visiting multiple task regions couples discrete route selection with continuous trajectory optimization. Graphs of Convex Sets (GCS) [1] encode candidate routes as discrete graph choices and their associated trajectories as continuous variables subject to convex costs and constraints. Although trajectory optimization is convex for a fixed route, jointly optimizing the discrete route and continuous trajectory is NP-hard in general [2]. GCS has enabled applications including contact-rich manipulation [3], [4] and multi-robot motion planning [5]–[7]. Building on the single-query shortest-path formulation, recent work has considered multi-query shortest paths [8], repeated-vertex shortest walks [9], and temporal-logic planning via product GCSs [10], [11] or augmented GCSs [12]. For multi-region routing, the Traveling Salesman Problem (TSP) on GCS seeks a minimum-cost closed trajectory that visits every GCS vertex while jointly choosing the visitation order and continuous trajectory variables [1], [13], [14]. Existing methods address these coupled decisions through a unified Mixed-Integer Convex Program (MICP) [1], hierarchical tour and path search [13], or an augmented GCS with exponentially many target-subset layers [14]. In many routing tasks, however, only selected GCS vertices correspond to required task regions.

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



(a) Feasible solution cost 89.7 (b) Optimal solution cost 64.3

Figure 1. Steiner-TSP on a GCS seeks a minimum-cost closed trajectory from root r (red) visiting all target convex sets (yellow). Black dashed arrows denote directed GCS edges (solid if used). The orange solution (a) is feasible without revisits, whereas the blue solution (b) exploits vertex revisits and is globally optimal, with circled numbers identifying the underlying GCS edges.

We study the *Steiner-TSP on GCS*, a required-subset generalization of the TSP on GCS. Only a designated subset of vertices must be visited, while non-required vertices remain available for transit and vertices may be revisited, consistent with discrete Steiner-TSP semantics [15]. A prior inspection-planning demonstration adapted a high-level restricted-TSP [16] formulation to require visits to selected GCS vertices [13]. Here, we formalize the general problem and study its solution structure. Compared with finding a shortest path in GCS [2], the Steiner-TSP on GCS must additionally choose the visitation order of the required vertices, the walks between successive visits, and the return walk to the root. Depending on the GCS topology, visiting all required vertices and returning to the root may require vertex revisits [13]; even when unnecessary for connectivity, revisits may reduce trajectory cost [9]. Candidate solutions therefore cannot be restricted a priori to walks without revisits. Once cyclic subwalks are admitted, the number of repetitions is unbounded, so even a finite GCS can induce infinitely many finite candidate walks. Efficiently searching this infinite family is therefore a central algorithmic challenge in solving the Steiner-TSP on GCS.

Recent work has explored graph search as an alternative for GCS optimization. For shortest-path problems, some methods interleave discrete expansion with convex optimization of partial or finite-horizon trajectories [4], [7], [17], whereas another line grows a vertex subset and repeatedly solves restricted GCS relaxations to obtain bounds [18]. Applying graph search to the Steiner-TSP on GCS raises a central design question: when should continuous trajectory optimization be performed? Optimizing every walk prefix can be expensive. Deferring full trajectory optimization until a complete target-covering closed walk is found, however, requires efficient lower bounds on the committed-prefix cost and the cost of visiting the remaining targets and returning to the root.

Our unified branch-and-bound search traverses the rooted-walk prefix tree in Fig. 1, maintaining one frontier of live prefixes (Sec. III). The tree represents every finite rooted walk admitted by the problem, including cycles and vertex revisits. To order and prune the frontier without optimizing each prefix, we combine an additive committed-prefix cost bound (Sec. IV-B) with a cut-separated connected-flow relaxation lower-bounding the cost of visiting every remaining target and returning to the root (Sec. IV-C). Their admissible sum reserves full trajectory optimization for complete target-covering closed walks (Sec. III-B). Under the uniform positive-cost assumption, best-first traversal terminates after finitely many expansions on every feasible instance without an initial incumbent; depth-first terminates after finitely many additional expansions once a finite incumbent exists. When either strategy terminates with a finite incumbent, the global lower bound certifies that its cost is at most ϵ times the global optimum for a user-specified factor $\epsilon \geq 1$ (Sec. III-C). Both variants return finite incumbents on all 180 benchmark instances within the 30 s budget (Sec. V). A mobile-manipulator case study demonstrates joint sensing-mode, visitation-order, GCS-walk, and continuous-trajectory selection with LTL$_f$ action precedences (Sec. VI).

II. Steiner-TSP on GCS

A GCS is a finite directed graph $G = (V, E, \mathcal{X})$. Each vertex $v \in V$ is associated with a nonempty compact convex set $\mathcal{X}_v$ and a positive convex cost $c_v : \mathcal{X}_v \to \mathbb{R}_{>0}$. Each edge $e = (u, v) \in E$ represents an allowed transition, with a closed convex feasible set $\mathcal{X}_e \subseteq \mathcal{X}_u \times \mathcal{X}_v$ and a nonnegative convex cost $c_e : \mathcal{X}_e \to \mathbb{R}_{\geq 0}$. We assume a uniform per-vertex minimum cost $\eta > 0$ such that $c_v(\mathbf{x}) \geq \eta$ for every $v \in V$ and $\mathbf{x} \in \mathcal{X}_v$. Notably, we allow a self-loop $(v, v) \in E$ for each $v \in V$. A *walk* $\pi = (v_1, \ldots, v_k)$ in G is a finite sequence of vertices such that $(v_i, v_{i+1}) \in E$ for all $i = 1, \ldots, k-1$, where vertices may repeat to support more general applications or solutions with smaller costs [9] (see also Fig. 1). Unlike a *path* that disallows vertex revisits, a walk may be necessary to form a closed route when G is incomplete. A *trajectory* $\tau = (\mathbf{x}_1, \ldots, \mathbf{x}_k)$ is *conditioned* on the walk π if $\mathbf{x}_i \in \mathcal{X}_{v_i}$ for $i = 1, \ldots, k$ and $(\mathbf{x}_i, \mathbf{x}_{i+1}) \in \mathcal{X}_{(v_i, v_{i+1})}$ for $i = 1, \ldots, k-1$. Each occurrence of a repeated vertex has its own continuous trajectory variable. For a trajectory conditioned on a closed walk, however, we require $\mathbf{x}_k = \mathbf{x}_1$ and call the trajectory *closed*. By convention, the terminal occurrence incurs no additional vertex cost. The trajectory cost is therefore

$$c(\tau) := \sum_{i=1}^{k-1} \left[c_{v_i}(\mathbf{x}_i) + c_{(v_i, v_{i+1})}(\mathbf{x}_i, \mathbf{x}_{i+1}) \right]. \tag{1}$$

Given a target set $V_t \subseteq V$ containing at least two vertices and a designated root $r \in V_t$, the *Steiner-TSP on GCS* is

$$\begin{aligned}
\min_{\pi, \tau} \quad & c(\tau) && && \text{(2a)}\\
\textbf{s.t.} \quad & (v_{i-1}, v_i) \in E, && i = 2, \ldots, k, && \text{(2b)}\\
& v_1 = v_k = r \text{ and } \mathbf{x}_1 = \mathbf{x}_k, && && \text{(2c)}\\
& V_t \subseteq \{v_1, \ldots, v_{k-1}\}, && && \text{(2d)}\\
& \tau \text{ is conditioned on } \pi. && && \text{(2e)}
\end{aligned}$$

A feasible solution consists of a closed trajectory conditioned on a closed walk that visits every target vertex at least once. This formulation generalizes the TSP on GCS [1], [13], which is recovered when $V_t = V$. Its required-subset semantics also coincide with the labeled-target setting of [14], and the Steiner terminology emphasizes that non-target GCS vertices remain available for transit without being required.

The Steiner-TSP on GCS couples the discrete walk π with the continuous trajectory τ. A *convex restriction* fixes π in Eqn. (2) and minimizes $c(\tau)$ over the continuous variables consistent with that walk. MICP [1] optimizes π and τ jointly within its formulation, whereas GHOST uses a two-level search [13]: a high-level restricted-TSP search explores target tours, while a low-level search constructs their realizing GCS walks and evaluates complete target-covering closed walks by convex restriction. Both baselines allow vertices to recur across different inter-target segments. MICP requires each segment to be vertex-simple, while GHOST prevents vertex revisits while searching toward the next target and advances in the selected tour once that target becomes adjacent.

III. Unified Branch-and-Bound Search

The Steiner-TSP on GCS has an intrinsic solution space with coupled discrete and continuous components. Its discrete component consists of finite rooted GCS walks, while each target-covering return to the root indexes a walk-conditioned convex restriction over the continuous trajectory variables; this restriction may be infeasible. As vertices may be revisited, this family of walks is generally infinite, and unrolling it yields a finitely branching prefix tree determined by the problem. Our unified branch-and-bound search systematically explores this tree by maintaining a single set of live walk prefixes and using the bounds developed in Sec. IV to prioritize and prune them.

A. Search Space

Formally, the discrete tree contains a search node n for each prefix walk $n.\pi = (v_1 = r, v_2, \ldots, v_k)$. This full-prefix identity is essential because two prefixes reaching the same frontier vertex v_k can induce different feasible boundary conditions and optimal conditioned trajectories even when followed by the same suffix [4], [7]. Let $V(n.\pi) := \{v_1, \ldots, v_k\}$ denote the set of vertices visited by $n.\pi$. For any successor w of v_k, let $n.\pi \oplus w := (v_1, \ldots, v_k, w)$ denote the prefix walk obtained by appending w. A node n is a solution node if $V_t \subseteq V(n.\pi)$ and $v_k = r$. This is only a discrete condition: the indexed convex restriction may be infeasible; when feasible, its optimal value is the minimum cost among trajectories conditioned on $n.\pi$. Every node n, including a solution node, has a child n_c with $n_c.\pi = n.\pi \oplus w$ for each edge $(v_k, w) \in E$. Consequently, the tree contains every finite rooted GCS walk, including walks that continue after an earlier target-covering return to r; together with the convex restrictions indexed by its solution nodes, it represents every feasible pair (π, τ) of Eqn. (2).

The tree is defined independently of how it is explored. Alg. 1 materializes it lazily by maintaining a set of live

search nodes, called the *Frontier*. A traversal strategy selects which live node to explore next. Each node n additionally stores a cost-to-come lower bound $\hat{g}(n)$ and a cost-to-go heuristic $h(n)$. We call h *admissible* with respect to $\hat{g}$ if, for every feasible solution (π', τ) whose walk extends $n.\pi$, $h(n) \leq c(\tau) - \hat{g}(n)$. Sec. III-B combines these quantities into the node lower bound used for Frontier ordering and pruning.

B. Search Procedure

We first define the node lower bound and traversal strategy used by the proposed Alg. 1. For a search node n and its child n_c induced by a successor w, the shared relaxation developed in Sec. IV supplies an incremental lower-bound cost $\ell(n, w)$, yielding $\hat{g}(n_c) = \hat{g}(n) + \ell(n, w)$. Since each extension commits one vertex cost, the uniform positive-cost assumption in Sec. II gives $\ell(n, w) \geq \eta$. Together with the admissible cost-to-go heuristic, this defines the node lower bound

$$f(n) := \hat{g}(n) + h(n), \tag{3}$$

which lower-bounds the cost of every feasible solution whose walk extends $n.\pi$. The traversal strategy controls only which live node is selected. We describe two canonical choices: best-first maintains a min-priority queue ordered by $f(n)$, whereas depth-first uses a last-in-first-out stack and locally orders siblings by $f(n)$. All traversal strategies prune any node satisfying $f(n) \geq \bar{c}$ and, once $\bar{c} < +\infty$, use the global stopping condition $\bar{c} \leq \epsilon \min_{n \in \text{Frontier}} f(n)$ for $\epsilon \geq 1$.

Alg. 1 accepts an optional initial incumbent to prune in the initial search stage. Our implementation constructs one by solving the convex restriction for a nearest-neighbor closed walk that repeatedly follows a shortest graph path to the closest unvisited target and finally returns to r. If its restriction is infeasible, no initial incumbent is supplied and $\bar{c} = +\infty$.

Alg. 1 initializes Frontier with one two-vertex walk (r, v) for every outgoing edge of r (lines 1–5). Each iteration checks the global stopping condition, selects a node according to the traversal strategy, and discards it if its bound cannot improve the incumbent (lines 6–11). An unpruned solution node is evaluated by convex restriction and then, like every unpruned node, expanded along all outgoing edges; only children whose cost-to-come and node bounds are below $\bar{c}$ are inserted (lines 12–20). If Frontier is exhausted or the stopping condition holds, it returns the incumbent if one exists (line 21).

Algorithm 1: Unified Branch-and-Bound Search

Input: GCS $G = (V, E, \mathcal{X})$, targets V_t, root r, optional initial incumbent τ^* with cost $\bar{c}$
Param: traversal strategy, suboptimality factor $\epsilon \geq 1$
Output: updated incumbent τ^* and its cost $\bar{c}$

1 Frontier $\leftarrow \emptyset$
2 **foreach** $(r, v) \in E$ **do**
3 $n \leftarrow \textsc{Node}((r, v), \hat{g} = 0)$
4 **if** $f(n) < \bar{c}$ **then**
5 Frontier.$\text{push}(n)$
6 **while** Frontier $\neq \emptyset$ **do**
7 **if** $\bar{c} < +\infty$ *and* $\bar{c} \leq \epsilon \min_{n \in \text{Frontier}} f(n)$ **then**
8 **break** ▷ ϵ-optimal incumbent found
9 $n \leftarrow$ Frontier.$\text{pop}()$
10 **if** $f(n) \geq \bar{c}$ **then**
11 **continue**
12 **if** $v_k = r$ *and* $V_t \subseteq V(n.\pi)$ **then**
13 $\tau \leftarrow \textsc{ConvexRestriction}(G, n.\pi)$
14 **if** τ *is feasible and* $c(\tau) < \bar{c}$ **then**
15 $(\tau^*, \bar{c}) \leftarrow (\tau, c(\tau))$
16 **foreach** $(v_k, w) \in E$ **do**
17 **if** $\hat{g}(n) + \ell(n, w) < \bar{c}$ **then**
18 $n_c \leftarrow \textsc{Node}(n.\pi \oplus w, \hat{g} = \hat{g}(n) + \ell(n, w))$
19 **if** $f(n_c) < \bar{c}$ **then**
20 Frontier.$\text{push}(n_c)$
21 **return** $(\tau^*, \bar{c})$

C. Theoretical Properties

We establish finite termination for best-first traversal (Theorem 1) and then show that it also holds for depth-first traversal once a finite incumbent becomes available (Remark 1).

Theorem 1 (Finite Best-First Search)**.** *Alg. 1 with best-first traversal terminates after finitely many node expansions for a feasible instance.*

Proof. Let $d(n) := |n.\pi| - 2$ be the number of extensions after an initial two-vertex node. Suppose first that initialization leaves $\bar{c} = +\infty$, and choose any feasible solution $(\bar{\pi}, \bar{\tau})$ with cost $C := c(\bar{\tau})$. Every node whose prefix lies on $\bar{\pi}$ satisfies $f(n) \leq C$ by the lower-bound property of f. Before an incumbent is found, $\bar{c} = +\infty$, so none of these prefixes is pruned. The proposed relaxation has nonnegative costs, so $h(n) \geq 0$. Since every extension increases $\hat{g}$ by at least η, any node with $f(n) \leq C$ satisfies $d(n)\eta \leq \hat{g}(n) \leq f(n) \leq C$. Since G is finite, only finitely many nodes have such bounded depth. Because best-first traversal always selects a minimum-key live node, induction over the finitely many prefixes of $\bar{\pi}$ shows that it reaches and evaluates the solution node indexed by $\bar{\pi}$ after finitely many expansions, unless another branch supplies a finite incumbent earlier.

Let $\bar{c}_0 < +\infty$ be the first finite incumbent cost, whether supplied initially or found during search. Only finitely many nodes have been generated when it becomes available. Every node inserted afterward satisfies $\hat{g}(n) < \bar{c} \leq \bar{c}_0$ and hence $d(n) < \bar{c}_0/\eta$. Finite branching thus allows only finitely many additional nodes to be generated for finite termination. ■

Remark 1 (Depth-First Traversal)**.** *The finite-depth argument in the proof of Theorem 1 is independent of the traversal strategy. Thus, depth-first traversal also terminates after finitely many additional expansions once a finite incumbent is found during search or supplied initially. Before then, pure last-in-first-out traversal may starve a live sibling on the infinite tree.*

Let $c^\star$ denote the optimal value of Eqn. (2). The following Theorem 2 show that the same lower-bound property yields an ϵ-optimality certificate for any traversal strategy.

Theorem 2 (ϵ-optimality)**.** *For any traversal strategy, if Alg. 1 terminates with a finite incumbent of cost $\bar{c}$ by satisfying the global stopping condition or exhausting Frontier, then $\bar{c} \leq \epsilon c^\star$.*

Proof. At termination, let

$$L := \min\left\{\bar{c}, \min_{n\in\text{Frontier}} f(n)\right\},$$

where the Frontier minimum is omitted if Frontier is empty. Consider any feasible solution (π, τ) of cost C. Exhaustive successor generation preserves the branch corresponding to π until its complete walk is evaluated, one of its prefixes remains in Frontier, or one of its prefixes is discarded by an incumbent-bound test. In the first case, the walk-conditioned convex restriction gives $\bar{c} \leq C$; in the second, $\min_{n\in\text{Frontier}} f(n) \leq C$; and in the third, the lower bound responsible for discarding the prefix gives $\bar{c} \leq C$. Therefore $L \leq C$ for every feasible solution, and taking the infimum over all feasible solutions gives $L \leq c^\star$. If the global stopping condition holds, then $\bar{c} \leq \epsilon L \leq \epsilon c^\star$. If Frontier is exhausted, $L = \bar{c} \leq c^\star$. Since the incumbent is feasible, $c^\star \leq \bar{c}$ and thus $\bar{c} = c^\star$. ■

IV. Lower-Bound Graph for Search Bounds

Efficient branch-and-bound requires inexpensive cost-to-come and cost-to-go bounds at every prefix, but future extensions can change the continuous trajectory along a prefix and hence its realized cost [4], [7]. Prefix-conditioned convex restrictions preserve continuous consistency [4], [7], [8], [17], while repeated fractional GCS relaxations over expanding vertex subsets offer another online bounding strategy [18]; both require continuous optimization during search, and our ablation shows that solving a prefix restriction at every generated node sharply limits search progress under the fixed runtime budget (Table II). To avoid this repeated online optimization, we precompute local convex-restriction costs for short GCS walks and compose them in a reusable discrete relaxation that enforces feasibility within each piece while relaxing consistency between adjacent pieces. This representation supplies a cost-to-come lower bound $\hat{g}(n)$ for the committed prefix and an admissible cost-to-go heuristic $h(n)$ for visiting the remaining targets and returning to the root, without a GCS trajectory solve at every generated node. For every feasible solution (π', τ) to Eqn. (2) whose walk π' extends $n.\pi$, these bounds satisfy $\hat{g}(n) + h(n) \leq c(\tau)$.

A. Lower-Bound Graph (LBG) Relaxation

The triplet-based LBG is one such representation and has previously been used for GCS search bounds [13], [17]; a closely related triplet relaxation is used in [7]. We adopt the LBG as the shared representation underlying both $\hat{g}$ and h. Given a GCS $G = (V, E, \mathcal{X})$, we construct its LBG $L = (E, T)$ as follows. Every length-two GCS walk (u, v, w) defines a triplet $t = (u, v, w) \in T$, directed in L from edge $e_1 = (u, v) \in E$ to edge $e_2 = (v, w) \in E$. We assign t the optimal value of the local convex program

$$\ell_t := \min_{\mathbf{x}_u, \mathbf{x}_v, \mathbf{x}_w} \left[c_v(\mathbf{x}_v) + c_{e_2}(\mathbf{x}_v, \mathbf{x}_w)\right], \tag{4}$$

where $\mathbf{x}_u \in \mathcal{X}_u$, $\mathbf{x}_v \in \mathcal{X}_v$, and $\mathbf{x}_w \in \mathcal{X}_w$ are constrained by $(\mathbf{x}_u, \mathbf{x}_v) \in \mathcal{X}_{e_1}$ and $(\mathbf{x}_v, \mathbf{x}_w) \in \mathcal{X}_{e_2}$. If the program in Eqn. (4) is infeasible, we set $\ell_t = +\infty$. The objective of Eqn. (4) includes exactly the costs of the middle vertex v and its outgoing edge e_2, as assigned in Eqn. (1), while the constraints enforce local feasibility with both neighboring edges. Thus, ℓ_t lower-bounds the trajectory cost committed at vertex v by any feasible trajectory whose walk contains t. The uniform positive-cost assumption in Sec. II gives $\ell_t \geq \eta$ for every finite-cost triplet $t \in T$.

Every GCS walk $\pi = (v_1, \ldots, v_k)$ therefore induces an LBG walk through the consecutive edges (v_i, v_{i+1}). Because each ℓ_t is computed independently, adjacent triplets need not share consistent GCS variables, so the induced LBG-walk cost lower-bounds the cost of any feasible trajectory conditioned on π. Consider any feasible solution to the Steiner-TSP on GCS, with trajectory τ conditioned on the GCS walk $\pi = (v_1 = r, v_2, \ldots, v_k = r)$. Closing the corresponding LBG walk gives

$$\hat{c}(\pi) := \sum_{i=1}^{k-2} \ell_{(v_i, v_{i+1}, v_{i+2})} + \ell_{(v_{k-1}, r, v_2)} \leq c(\tau), \tag{5}$$

where $\hat{c}(\pi)$ denotes the cost of the closed LBG walk induced by π. This lower bound is the closed-walk case of Theorem 1 in [13], extending the open-path result in Theorem 3 of [17].

B. Cost-to-Come Lower Bound

Consider a search node n with $n.\pi = (v_1 = r, v_2, \ldots, v_k)$. We define its cost-to-come lower bound $\hat{g}(n)$ as the cost of the corresponding fixed LBG walk (Lemma 3):

$$\hat{g}(n) := \sum_{i=1}^{k-2} \ell_{(v_i, v_{i+1}, v_{i+2})}. \tag{6}$$

For the initial prefix $n.\pi = (r, v_2)$, the sum is empty, so $\hat{g}(n) = 0$. Appending a successor w extends the LBG walk by $t = (v_{k-1}, v_k, w) \in T$ and produces a child n_c satisfying

$$\hat{g}(n_c) = \hat{g}(n) + \ell(n, w) = \hat{g}(n) + \ell_t, \tag{7}$$

where the incremental lower bound in Sec. III-B is instantiated as $\ell(n, w) := \ell_t$. Because $\ell_t \geq \eta$, each extension provides the positive progress required by the proof of Theorem 1.

Lemma 3. *For any search node n and feasible solution (π', τ) whose walk extends $n.\pi$, we have $\hat{g}(n) \leq \hat{c}(\pi') \leq c(\tau)$.*

Proof. The first inequality holds because $\hat{g}(n)$ is a partial sum of the nonnegative LBG-walk cost $\hat{c}(\pi')$, and the second one comes from the LBG lower-bound property [7], [13], [17]. ■

C. Cut-Separated Connected-Flow Cost-to-Go

We propose an admissible heuristic $h(n)$ that lower-bounds the cost of visiting every unvisited target and closing the walk. The heuristic is evaluated online at every search node by a cut-separated connected-flow Linear Program (LP). Consider LBG $L = (E, T)$. With a slight abuse of notation, T denotes only

the finite-cost triplets throughout this subsection. For a search node n with $n.\pi = (v_1 = r, v_2, \ldots, v_k)$, let

$$e_{\mathrm{src}} := (v_{k-1}, v_k) \in E \text{ and } V_t(n) := V_t \setminus V(n.\pi).$$

For each $v \in V_t(n)$, define its incoming-edge set

$$E_v := \{(u, v) \in E\}.$$

Because v is unvisited by $n.\pi$, $e_{\mathrm{src}} \notin E_v$. Because each GCS edge is an LBG vertex, a GCS walk reaches v exactly when its induced LBG walk visits a vertex in E_v. For any feasible solution (π', τ) to Eqn. (2) with $n.\pi$ as a prefix of π', the residual LBG walk induced by π' therefore starts at e_{src}, visits a vertex in E_v for every $v \in V_t(n)$, and returns through the initial edge (r, v_2) to close the walk.

To enforce closure of the LBG walk through the final triplet (u, r, v_2), let e_{cl} denote an artificial sink vertex. Let $E^\sharp := E \cup \{e_{\mathrm{cl}}\}$, and let $T^\sharp$ augment T with, for every closing triplet $(u, r, v_2) \in T$, a terminal transition from (u, r) to e_{cl} with cost $\ell_{(u,r,v_2)}$. Define the admissible source-side sets by

$$\mathcal{S}(n) := \bigcup_{v \in V_t(n)} \{S \subseteq E \setminus E_v : e_{\mathrm{src}} \in S\}.$$

For each $S \in \mathcal{S}(n)$, define its outgoing finite-triplet cut by

$$\delta^+(S) := \{(u, v, w) \in T : (u, v) \in S,\ (v, w) \notin S\}.$$

For a nonnegative flow ξ on $T^\sharp$, define $\operatorname{div} \xi \in \mathbb{R}^{E^\sharp}$ componentwise, where $(\operatorname{div} \xi)_e$ is the total outgoing minus incoming flow at edge $e \in E^\sharp$. For each $e \in E^\sharp$, let $\mathbf{1}_e \in \{0, 1\}^{E^\sharp}$ denote the unit vector at e. Let $\boldsymbol{\ell}^\sharp$ contain the corresponding costs on $T^\sharp$. We define $h(n)$ as the optimal value of

$$\min_{\xi \in \mathbb{R}^{T^\sharp}_{\geq 0}} \quad (\boldsymbol{\ell}^\sharp)^\top \xi \tag{8a}$$

$$\mathbf{s.t.} \quad \operatorname{div} \xi = \mathbf{1}_{e_{\mathrm{src}}} - \mathbf{1}_{e_{\mathrm{cl}}}, \tag{8b}$$

$$\sum_{t \in \delta^+(S)} \xi_t \geq 1, \quad \forall S \in \mathcal{S}(n). \tag{8c}$$

Eqn. (8b) routes one net unit of cost-bearing flow from e_{src} to e_{cl} through a valid closing terminal transition, while permitting additional circulations. Eqn. (8c) requires at least one unit across every directed cut separating e_{src} from each remaining target's incoming-edge set. Hence, a disconnected target-covering circulation cannot by itself certify target visitation. When $V_t(n) = \emptyset$, the LP reduces to a minimum-cost flow from e_{src} to e_{cl}. Whenever the LP is infeasible, $h(n) = +\infty$.

By the max-flow/min-cut theorem, these cut constraints are equivalent to sending one auxiliary unit of flow from e_{src} to each E_v under capacities ξ. This construction adapts the multi-commodity-flow formulation for the classic Steiner-TSP [19] by replacing each required vertex with its incoming-edge set E_v. We solve this exponential cut formulation by row generation using a persistent restricted master. After each solve, we use ξ as arc capacities, add any connectivity cuts violated by an e_{src}-to-E_v minimum cut, and reoptimize. Lemma 4 establishes that h is an admissible cost-to-go heuristic.

Lemma 4 (Admissibility)**.** *For any search node n and feasible solution (π', τ) whose walk extends $n.\pi$, we have*

$$\hat{g}(n) + h(n) \leq \hat{c}(\pi') \leq c(\tau). \tag{9}$$

Thus, h is admissible with respect to $\hat{g}$ by definition, and $f(n)$ in Eqn. (3) *lower-bounds the cost of every feasible solution whose walk extends $n.\pi$.*

Proof. The multiplicities of the finite triplets in the residual LBG walk of π', together with its closing terminal transition, define an integral flow ξ satisfying Eqn. (8b). For each $S \in \mathcal{S}(n)$, choose a remaining target v such that $S \cap E_v = \emptyset$. The residual walk starts in S and reaches an edge in E_v outside S. It must therefore cross $\delta^+(S)$ at least once, so every connectivity cut is satisfied. The resulting objective is $\hat{c}(\pi') - \hat{g}(n)$. Optimality gives $h(n) \leq \hat{c}(\pi') - \hat{g}(n)$, and the second inequality follows from Eqn. (5). ■

V. Numerical Results

This section presents our numerical results for Steiner-TSP on GCS solvers. We implement the proposed search and its components in *Python* and evaluate the resulting solver on a machine with an *Apple*® M4 processor and 16 GB of memory. The GCS-related trajectory optimization uses the *Drake* [20] library configured with the *Gurobi* solver [21]. Gurobi also solves the persistent restricted master for the cost-to-go bound, while directed minimum-cut computations identify violated connectivity rows. Source code, numerical results, and additional visualizations will be released upon publication. More detailed visualizations and simulations are available at https://sites.google.com/view/steiner-tsp-gcs.

A. Instances

We evaluate the Steiner-TSP on GCS across the three task domains shown in Fig. 2. All instances use the same minimum-time trajectory model [22]. Each visit to a GCS vertex is parameterized by degree-5 Bézier curves for the d-dimensional configuration and scalar time. Accordingly, each variable $\mathbf{x}_i \in \mathbb{R}^{6(d+1)}$ in Eqn. (2) comprises six d-dimensional configuration control points and six scalar time control points. We impose C^2 continuity across internal GCS transitions, constrain all configuration control points to the selected GCS region, and enforce componentwise velocity bounds on each segment. We impose neither acceleration nor torque limits. All edge costs are zero, and each vertex cost equals its segment duration, with a minimum duration of $\eta = 0.1\,\mathrm{s}$.

Each domain contains 60 instances generated from seeds $0, \ldots, 11$ and five controlled sizes. For *rand*, point configurations move through convex hulls of six points sampled within 1.5×1.5 squares on a unit-spaced grid, yielding a size-dependent workspace. Each seed grows a connected cover to undirected intersection-edge thresholds of $10, 20, \ldots, 50$, with every polygon treated as a target ($V = V_t$). For *maze*, point configurations move through a random $10 \times 10 \times 10$ unit-voxel recursive-division maze [23] with one-voxel-thick walls. We partition its free voxels into a minimum-cardinality set of

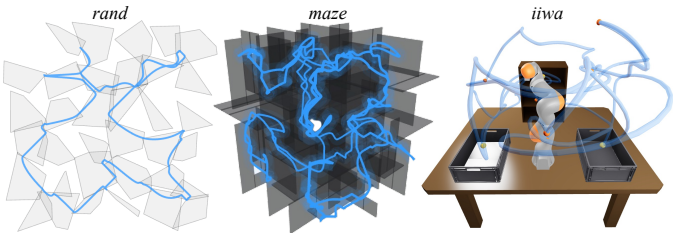


Figure 2. Steiner-TSP on GCS solution trajectories (blue) in *rand* with a 2-D GCS of gray polygons, *maze* with a 3-D GCS exactly partitioning the free voxels outside black walls, and *iiwa* with a 7-D GCS comprising IRIS-NP [25] regions seeded by representative collision-free configurations.

disjoint axis-aligned cuboids, a 3-D analogue of the minimum-brick decomposition in [24], where cuboids sharing a face define GCS adjacencies. We select $10, 20, \ldots, 50$ cuboids distributed across each maze GCS as targets. For *iiwa*, we use a precomputed collision-free 15-region joint-space GCS for a fixed-base 7-DoF KUKA LBR iiwa 14 in a shelves-and-bins scene, with IRIS-NP regions [25] bounded by the robot's joint-position limits. Each instance adds $3, 6, 9, 12$, or 15 exclusive target boxes with halfwidth at most $0.01\,\mathrm{rad}$, each adjacent only to its owner region. In all three domains, we build two oppositely directed edges if the configuration convex sets of a GCS vertex pair intersect. Table I summarizes the resulting graphs, with Degree computed using undirected adjacency.

B. Solvers

We compare six solver variants: two traversal strategies for the proposed search, two bound ablations, and two existing solvers. We instantiate the proposed search with the best-first (BF) and depth-first (DF) traversal strategies, denoted **Ours (BF)** and **Ours (DF)**, respectively. Both use the proposed additive LBG prefix bound $\hat{g}$ (Sec. IV-B) and cut-separated connected-flow cost-to-go bound h (Sec. IV-C). The ablation study isolates the two proposed bounds. **Alt-G** retains h but replaces $\hat{g}$ with the optimal committed-prefix cost obtained from a prefix convex restriction at every generated node, whereas **Alt-H** retains $\hat{g}$ but replaces h with an LBG heuristic that connects the current prefix to the remaining targets, spans them with a metric-closure MST, and returns to the root. Both Alt-G and Alt-H use the best-first traversal strategy. For the performance comparison, we include **GHOST** [13] and **MICP** [1], as described in Sec. II. We provide the active edges of a greedy closed walk as a partial Mixed-Integer Programming (MIP) start for MICP. The walk repeatedly connects the current vertex to the closest unvisited target by an unweighted shortest path on the input GCS. All solvers receive a runtime budget $T = 30\,\mathrm{s}$.[1]

C. Metrics

We evaluate terminal quality, anytime performance, and search effort under runtime budget T, using as each instance's *reference cost* the lowest incumbent at T across all six solver variants. **Cost Regret** is the terminal incumbent's fractional excess over this reference, whereas **Gap** is the fractional difference between the terminal incumbent and the solver-reported terminal lower bound, capped at 100%. Both are averaged only over runs with a finite incumbent, with a missing finite lower bound assigned a 100% gap; because GHOST and MICP restrict the walk family (Sec. II), their gaps do not certify the unrestricted problem. **Feasible** counts runs with a finite incumbent at T.

Following [26], the Time-Normalized Primal Integral (**NPI**) averages over $[0, T]$ the incumbent's excess over the reference cost, normalized by the current incumbent and capped at one. Periods without an incumbent contribute one, so lower NPI reflects earlier feasibility and better anytime solution quality.

For search effort, **Closed Nodes** counts popped and processed frontier nodes, whereas **Evaluated Walks** counts complete target-covering closed walks passed to a convex restriction. Closed Nodes is omitted for GHOST and MICP, and Evaluated Walks is omitted for MICP; for the four search-based variants, Evaluated Walks includes the greedy initial walk but excludes Alt-G's prefix restrictions, which are not complete walks. Both terminal counters may include the final operation begun before T and completed afterward.

[1]For the search-based solvers, LBG precomputation is performed offline and excluded from the runtime budget. For MICP, model construction and partial MIP-start generation are also excluded.

Table I
STEINER-TSP ON GCS INSTANCE COMPLEXITY. THE $|V|$, $|E|$, DEGREE, AND $|V_t|$ COLUMNS REPORT THE MINIMUM/MEAN/MAXIMUM. THE d COLUMN REPORTS THE CONFIGURATION-SPACE DIMENSION, AND LBG COLUMN REPORTS THE MEAN PRECOMPUTATION TIME IN SECONDS.

Domain	$\lvert V\rvert$	$\lvert E\rvert$	Degree	$\lvert V_t\rvert$	d	LBG
rand	8/23.6/41	20/60.6/104	1/2.51/7	8/23.6/41	2	0.07s
maze	63/63/63	124/125.2/130	1/1.99/4	10/30/50	3	0.14s
iiwa	18/24/30	104/116.0/128	1/4.93/13	3/9/15	7	4.57s

D. Result Analysis

Table II shows that both proposed variants find incumbents on all 180 instances, compared with 99 for GHOST and 98 for MICP, while their lower NPI indicates earlier near-reference solutions. Ours (BF) performs best across the three quality metrics on *rand* and *iiwa*, whereas Ours (DF) achieves the best gap and NPI on *maze* and ties GHOST in displayed regret.

GHOST and MICP both exhibit complementary bottlenecks. GHOST remains competitive when successful, with mean regret at most 3.9%, but none of its 81 failed runs evaluates a complete walk: failures exhaust T in the initial restricted-TSP solve for *maze*, in the low-level GCS-walk search for *iiwa*, and at either level for larger *rand* instances. MICP finds incumbents on all *maze* instances but none on *iiwa*, and all its reported gaps reach the 100% cap. This contrast plausibly reflects formulation difficulty rather than raw GCS size: MICP creates one GCS copy per target and initializes only selected edge indicators, while a representative 3-target *iiwa* model has 4.28 million nonzeros, 5.4 times that of a 10-target *maze* model. This is consistent with easier initialization on nearly

Table II
ABLATION AND PERFORMANCE COMPARISON RESULTS.

	Solver	Cost Regret↓	Gap↓	Feasible↑	NPI↓	Closed Nodes	Eval. Walks
rand	**GHOST [13]**	2.1%	50.4%	43/60	0.386	–	294
	MICP [1]	24.5%	100.0%	38/60	0.603	–	–
	Ours (BF)	**0.0%**	**18.3%**	**60/60**	**0.011**	8.6k	666
	Ours (DF)	1.8%	21.7%	**60/60**	0.023	1.2k	429
	Alt-G	28.0%	33.0%	**60/60**	0.205	736	1
	Alt-H	26.8%	54.7%	**60/60**	0.197	749k	56
maze	**GHOST [13]**	**0.0%**	3.4%	12/60	0.802	–	23
	MICP [1]	2.2%	100.0%	**60/60**	0.163	–	–
	Ours (BF)	0.7%	3.6%	**60/60**	0.014	53.5k	168
	Ours (DF)	**0.0%**	**3.1%**	**60/60**	**0.005**	1.3k	173
	Alt-G	2.5%	5.2%	**60/60**	0.027	453	1
	Alt-H	2.5%	52.7%	**60/60**	0.027	871k	1
iiwa	**GHOST [13]**	3.9%	64.6%	44/60	0.383	–	37
	MICP [1]	–	–	0/60	1.000	–	–
	Ours (BF)	**0.2%**	**62.4%**	**60/60**	**0.042**	830	54
	Ours (DF)	4.0%	64.4%	**60/60**	0.062	98	30
	Alt-G	10.6%	64.0%	**60/60**	0.111	43	1
	Alt-H	6.2%	65.1%	**60/60**	0.083	314k	37

tree-structured six-facet maze regions than on *rand*, where all vertices are targets, or on 7-D *iiwa* regions with 48–237 facets.

The ablations confirm $\hat{g}$ and h are complementary. Alt-H is inexpensive but loose because its pairwise LBG connections need not form one ordered residual walk that visits every remaining target and returns to the root, leaving many prefixes competitive and closing 314k–871k nodes while evaluating only 1–56 complete walks. Alt-G instead solves a prefix convex restriction for every generated child before pruning; these solves are not captured by either effort counter and limit the search to 43–736 closed nodes without evaluating a complete walk beyond the greedy initialization. Combining the inexpensive additive $\hat{g}$ with the connectivity-aware h avoids both bottlenecks and focuses the frontier on complete walks.

The two traversal strategies also have complementary strengths. Ours (BF) globally prioritizes prefixes by $\hat{g}+h$ and achieves lower NPI on *rand* and *iiwa*, whereas Ours (DF) uses the same bound for pruning and sibling ordering and performs better on the nearly tree-structured *maze*.

VI. MOBILE-MANIPULATOR INSPECTION CASE STUDY

We demonstrate Ours (BF) using a mobile manipulator comprising the same 7-DoF arm as in the *iiwa* domain, a wrist-mounted camera, and a 3-DoF holonomic planar base with pose $(x_b, y_b, \theta_b) \in \mathbb{R}^2 \times S^1$. As shown in Fig. 3, the robot departs from a charging dock, navigates the workspace to acquire eight inspection images, and returns to the dock. We precompute the GCS $G = (V, E, \mathcal{X})$ as coupled navigation and inspection subgraphs, totaling 92 vertices and 208 directed edges. The navigation subgraph contains collision-free regions in the 3-DoF base configuration space, lifted to the full 10-DoF robot configuration space by fixing the manipulator in a retracted posture. The inspection subgraph contains convex regions generated by IRIS-NP [25] in the 7-DoF manipulator

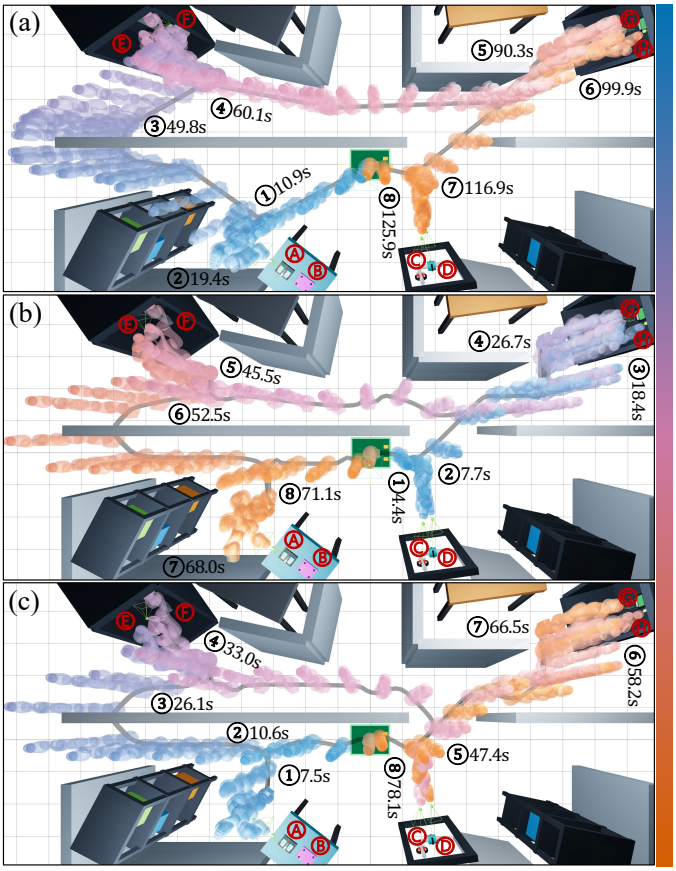


Figure 3. Mobile-manipulator inspection: (a) PRM-based generalized-TSP baseline, (b) Ours (BF), and (c) Ours (BF) with LTL_f [27] action precedences. Translucent arm poses encode normalized active-motion progress from blue (early) to orange (late), as shown by the right color bar. Red circled A–H mark tasks; black circled 1–8 show execution order and simulation timestamps.

configuration space, with the base fixed at one of the inspection base poses represented in the navigation subgraph. The charging-dock vertex serves as the root $r \in V$, and each inspection task $i = 1, \ldots, 8$ is represented by a nonempty service-vertex subset $V_i \subseteq V \setminus \{r\}$, where each $v \in V_i$ encodes one feasible combination of base pose and manipulator region for acquiring image i. The GCS construction took $0.4\,\text{s}$ and its LBG precomputation took $6.0\,\text{s}$.

We formulate this inspection task as a Steiner-TSP on the precomputed GCS G by generalizing the coverage constraint in Eqn. (2d) so that the rooted closed walk must visit at least one vertex in each V_i. Across the eight tasks, the alternatives encoded by $V_1, \ldots, V_8$ yield 96 sensing-mode assignments; if every V_i is a singleton, the formulation reduces to Eqn. (2) with $V_t = \{r\} \cup \bigcup_i V_i$. The search tracks which tasks have been covered and represents each uncovered V_i in the connected-flow cost-to-go relaxation by the union $\bigcup_{v \in V_i} E_v$; all other machinery is unchanged, and the search jointly selects the modes, order, walk, and trajectory. We adopt the same benchmark trajectory model in Sec. V-A.

Ours (BF) finds an initial $91.4\,\text{s}$ greedy solution in $0.35\,\text{s}$ with a 19.03% gap, and improves it to $78.4\,\text{s}$ in $1.72\,\text{s}$ with a

5.64% gap after 642 closed search nodes (Fig. 3(b)). For comparison, we construct a PRM [28] for the 3-DoF base (in $2.0\,\mathrm{s}$) with the manipulator fixed in a retracted posture, and augment it with sampled feasible 7-DoF manipulator configurations at the same inspection base poses in the GCS using OMPL [29]. We then solve the resulting generalized-TSP [30] for the same eight inspection tasks. It returns a feasible $133.1\,\mathrm{s}$ solution in $31.6\,\mathrm{s}$ without an optimality guarantee (Fig. 3(a)).

Fig. 3(c) further demonstrates the versatility of Ours (BF) by augmenting the inspection task with a linear temporal logic over finite traces (LTL_f) specification [27]. It additionally enforces $\{\mathrm{A},\mathrm{B}\} \prec \{\mathrm{E},\mathrm{F}\} \prec \mathrm{D} \prec \{\mathrm{G},\mathrm{H}\} \prec \mathrm{C}$ for the task, where $\prec$ denotes a precedence relation. Following prior works over product GCSs [10], [11], we compile the specification into a deterministic finite automaton and explicitly track its state in each search node. Ours (BF) finds an initial $87.4\,\mathrm{s}$ greedy incumbent in $0.4\,\mathrm{s}$ and, after closing 17 search nodes, improves it to $82.3\,\mathrm{s}$ in about $1.3\,\mathrm{s}$ with a 10.12% gap.

VII. Conclusion

We formalized Steiner-TSP on GCS and proposed a unified branch-and-bound search over rooted walk prefixes. With uniformly positive GCS vertex costs, best-first traversal terminates after finitely many expansions on every feasible instance without an initial incumbent, whereas depth-first traversal does so once a finite incumbent is available. A global lower bound provides an ϵ-optimality certificate for either strategy. Future work will apply the proposed search to physical robots for energy-aware service and maintenance, study online replanning via incremental space-time GCS [7] updates with ultrafast convex-set construction [31], and develop scalable multi-robot planning through coordinated robot-local product GCSs.

References

[1] T. Marcucci, "Graphs of convex sets with applications to optimal control and motion planning," Ph.D. dissertation, Massachusetts Institute of Technology, 2024.

[2] 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.

[3] B. P. Graesdal, S. Y. C. Chia, T. Marcucci, S. Morozov, A. Amice, P. A. Parrilo, and R. Tedrake, "Towards tight convex relaxations for contact-rich manipulation," in *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024.

[4] 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.

[5] J. Tang, Z. Mao, L. Yang, and H. Ma, "Space-time graphs of convex sets for multi-robot motion planning," in *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2025, pp. 8683–8690.

[6] S. Zhao, A. G. Philip, S. Rathinam, H. Choset, and Z. Ren, "CB-GCS: Conflict-based search on the graph of convex sets for multi-agent motion planning," in *2025 IEEE 21st International Conference on Automation Science and Engineering (CASE)*. IEEE, 2025, pp. 2208–2214.

[7] J. Tang, Z. Mao, L. Yang, and H. Ma, "Search-based spatiotemporal and multi-robot motion planning on graphs of space-time convex sets," *arXiv preprint arXiv:2607.00444*, 2026.

[8] S. Morozov, T. Marcucci, A. Amice, B. P. Graesdal, R. Bosworth, P. A. Parrilo, and R. Tedrake, "Multi-query shortest-path problem in graphs of convex sets," in *International Workshop on the Algorithmic Foundations of Robotics*. Springer, 2024, pp. 67–86.

[9] S. Morozov, T. Marcucci, B. P. Graesdal, A. Amice, P. A. Parrilo, and R. Tedrake, "Mixed discrete and continuous planning using shortest walks in graphs of convex sets," *arXiv preprint arXiv:2507.10878*, 2025.

[10] V. Kurtz and H. Lin, "Temporal logic motion planning with convex optimization via graphs of convex sets," *IEEE Transactions on Robotics*, vol. 39, no. 5, pp. 3791–3804, 2023.

[11] Z. Wei, X. Luo, and C. Liu, "Hierarchical temporal logic task and motion planning for multi-robot systems," in *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2025.

[12] S. You, G. Luna, and T. H. Summers, "A framework for motion planning with temporal logic precedence specifications via augmented graphs of convex sets," *arXiv preprint arXiv:2606.00842*, 2026.

[13] J. Tang and H. Ma, "Ghost: Solving the traveling salesman problem on graphs of convex sets," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 40, no. 43, 2026, pp. 36 421–36 428.

[14] G. Luna and T. Summers, "Augmented graphs of convex sets and the traveling salesman problem," *arXiv preprint arXiv:2604.06406*, 2026.

[15] J. Rodríguez-Pereira, E. Fernández, G. Laporte, E. Benavent, and A. Martínez-Sykora, "The steiner traveling salesman problem and its extensions," *European Journal of Operational Research*, vol. 278, no. 2, pp. 615–628, 2019.

[16] H. W. Hamacher and M. Queyranne, "K best solutions to combinatorial optimization problems," *Annals of Operations Research*, vol. 4, no. 1, pp. 123–143, 1985.

[17] 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.

[18] K. Sundar and S. Rathinam, "A* for bounding shortest paths in the graphs of convex sets," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 35, no. 1, 2025, pp. 260–268.

[19] A. N. Letchford, S. D. Nasiri, and D. O. Theis, "Compact formulations of the Steiner Traveling Salesman Problem and related problems," *European Journal of Operational Research*, vol. 228, no. 1, pp. 83–92, 2013.

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

[21] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2026. [Online]. Available: https://www.gurobi.com

[22] 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.

[23] J. Buck, "Maze generation: Recursive division," https://weblog.jamisbuck.org/2011/1/12/maze-generation-recursive-division-algorithm, Jan. 2011, blog post, accessed 2026-03-09.

[24] J. Lu, B. Zeng, J. Tang, T. L. Lam, and J. Wen, "TMSTC*: A path planning algorithm for minimizing turns in multi-robot coverage," *IEEE Robotics and Automation Letters*, vol. 8, no. 8, pp. 5275–5282, 2023.

[25] M. Petersen and R. Tedrake, "Growing convex collision-free regions in configuration space using nonlinear programming," *arXiv preprint arXiv:2303.14737*, 2023.

[26] T. Berthold, "Measuring the impact of primal heuristics," *Operations Research Letters*, vol. 41, no. 6, pp. 611–614, 2013.

[27] G. De Giacomo and M. Y. Vardi, "Linear temporal logic and linear dynamic logic on finite traces," in *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, 2013, pp. 854–860.

[28] 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.

[29] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, 2012.

[30] M. Saha, T. Roughgarden, J.-C. Latombe, and G. Sánchez-Ante, "Planning tours of robotic arms among partitioned goals," *The International Journal of Robotics Research*, vol. 25, no. 3, pp. 207–223, 2006.

[31] P. Werner, R. Cheng, T. Stewart, R. Tedrake, and D. Rus, "Superfast configuration-space convex set computation on GPUs for online motion planning," in *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2025.