Scalable Inspection Planning via Flow-based Mixed Integer Linear Programming

Adir Morgan*, Kiril Solovey, and Oren Salzman

Technion–Israel Institute of Technology, Haifa, Israel
samorgan@campus.technion.ac.il, kirilsol@technion.ac.il,
osalzman@cs.technion.ac.il

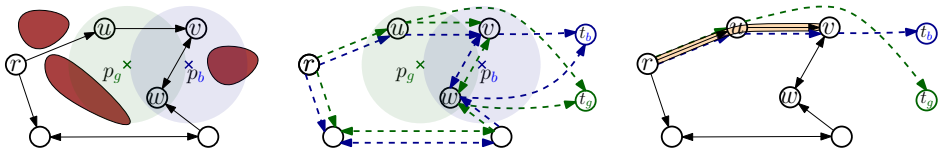
Abstract. *Inspection planning* is concerned with computing the shortest robot path to inspect a given set of points of interest (POIs) using the robot's sensors. This problem arises in a wide range of applications from manufacturing to medical robotics. To alleviate the problem's complexity, recent methods rely on sampling-based methods to obtain a more manageable (discrete) *graph inspection planning* (GIP) problem. Unfortunately, GIP still remains highly difficult to solve at scale as it requires simultaneously satisfying POI-coverage and path-connectivity constraints, giving rise to a challenging optimization problem, particularly at scales encountered in real-world scenarios. In this work, we present highly scalable Mixed Integer Linear Programming (MILP) solutions for GIP that significantly advance the state-of-the-art in both runtime and solution quality. Our key insight is a reformulation of the problem's core constraints as a network flow, which enables effective MILP models and a specialized Branch-and-Cut solver that exploits the combinatorial structure of flows. We evaluate our approach on medical and infrastructure benchmarks alongside large-scale synthetic instances. Across all scenarios, our method produces substantially tighter lower bounds than existing formulations, reducing optimality gaps by 30–50% on large instances. Furthermore, our solver demonstrates unprecedented scalability: it provides non-trivial solutions for problems with up to *15,000 vertices* and thousands of POIs, where prior state-of-the-art methods typically exhaust memory or fail to provide any meaningful optimality guarantees.

1 Introduction & Related Work

In inspection planning (IP), a robot equipped with an onboard sensor is tasked with computing a path in a known environment to inspect a set of points of interest (POIs) while avoiding obstacles and minimizing path cost. Applications of IP arise in a wide range of domains, including construction [8], manufacturing [5], and medical robotics [9, 10].

State-of-the-art approaches to the IP problem [19, 20] use sampling-based motion planning to discretize the robot's continuous configuration space into a roadmap represented as a graph. The vertices and edges of this graph correspond

* Corresponding author.



(a) (b) (c)

Fig. 1: Graph inspection planning, with network flow reduction for inspecting tour generation: (a) A given inspection planning graph, where the POI p_b can be inspected from vertices v and w, and the POI p_g can be inspected from u and w (depicted by colored circles surrounding each POI). (b) Pseudo-terminals t_b and t_g are introduced, and a multi-commodity network flow problem is formulated (dashed colored edges), where providing positive flow from the root r to each pseudo-terminal will correspond to a path from the root to visit an inspecting configuration for the associated POI. (c) A specific flow solution (orange edges), minimizing the cumulative path cost to inspect all POIs. Additional constraints will handle path returning to root.

to valid robot configurations and feasible local motions, respectively. Edge costs encode the cost of local motion, and directed edges may be used to capture asymmetric motions arising from kinodynamic constraints. Each vertex is additionally associated with a *visibility set* that specifies which POIs can be inspected from that configuration. The IP task then reduces to finding a minimum-cost tour on this graph that inspects all POIs, a problem known as *graph inspection planning* (GIP). GIP presents a significant computational challenge, as it entails the simultaneous selection of a subset of vertices that jointly cover all POIs, together with the computation of a minimum-length tour connecting them. Thus, GIP generalizes both the *set cover* problem and the *traveling salesman problem* (TSP) [26, 30], both of which are NP-hard, and even hard to approximate within constant factors [17, 37].

Fu et al. [19] proposed a search-based approach that reformulates GIP as a shortest-path problem on a new graph G_{IP}, whose vertex set captures all original roadmap vertices and all possible POI subsets. Consequently, the size of G_{IP} is exponential in the number of POIs. While this approach provides theoretical guarantees on solution quality, it does not manage to handle real-world instances involving thousands of POIs within a reasonable time frame, thus limiting its practical applicability. More recently, Mizutani et al. [34] tackled GIP using dynamic programming and mixed-integer linear programming (MILP) approaches. Among the two, the most scalable is their MILP formulation, solved using the Branch-and-Bound (BnB) paradigm (see Sec. 5). While this formulation can handle substantially larger GIP instances than earlier methods, its performance degrades as the graph size and the number of POIs grow, leading to loose lower bounds and limited near-optimality certification, even for modest graph sizes (relative to those commonly considered in motion planning [36]). This motivates alternative formulations that retain strong solution-quality guarantees while enabling scalability.

Contribution. We develop MILP-based GIP solvers that scale to large instances while providing tight bounds on solution quality. Our approach is driven by a flow-based interpretation of GIP, illustrated in Fig. 1, which demonstrates how

coverage and connectivity constraints can be expressed through network-flow structures within MILP formulations. Toward this goal, we start by formally defining the GIP problem (Sec. 2) and then reformulate it using a group-covering perspective (Sec. 3). This perspective situates GIP within a well-studied class of combinatorial optimization problems and yields structural insights that guide the design of effective MILP solvers. Using this perspective, we present a baseline MILP formulation that enforces coverage and local structure (Sec. 4.1), yet lacks global connectivity enforcement. We next propose three different approaches to ensure global connectivity (Sec. 4.2), each offering a trade-off between formulation compactness and solver strength.

Our most scalable approach out of the three relies on network-flow cutset constraints, that are too numerous to be explicitly evaluated simultaneously. Instead, we consider a lazy constraint-generation approach with the Branch-and-Cut framework [33, 35]. To construct this solver, we introduce in Sec. 5 tailored algorithmic building blocks, that enable on-demand constraint generation and the construction of progressively-improving feasible solutions to GIP. We conclude (Sec. 6) with an extensive experimental evaluation on real-world and simulated instances spanning a wide range of problem scales, demonstrating the scalability of the proposed approach and the trade-offs between different formulations.

2 Problem Definition

Let $G = (V, E, c)$ be a weighted directed graph that abstracts the robot's motion, where vertices correspond to robot configurations and edges correspond to dynamically feasible, collision-free motions with cost $c(e)$. Let $\mathcal{P}$ denote a set of points of interest (POIs) in the workspace. The robot is equipped with an exteroceptive sensor (e.g. camera, lidar), inducing a *coverage function* $\chi : V \to 2^{\mathcal{P}}$ that specifies the POIs inspected when the robot is positioned at a given vertex.

Given a designated root vertex $r \in V$, a *tour* τ is a closed path starting and ending at r, traversing edges in E. The cost of a tour is the sum of its edge costs, denoted $c(\tau)$, and its coverage is the union of POIs observed along the tour, denoted $\chi(\tau)$. Let $\mathcal{T}(G, r)$ denote the set of all such tours.

Definition 1 (Graph inspection planning). *Given* $\langle G = (V, E, c), r, \mathcal{P}, \chi \rangle$, *find a minimum-cost tour from* r *that inspects all POIs, i.e.,*

$$\arg \min_{\tau \in \mathcal{T}(G,r)} c(\tau) \quad \textit{subject to} \quad \chi(\tau) = \mathcal{P}.$$

3 Graph Inspection Planning via Group Covering

To set the foundations for the design of scalable MILP formulations for GIP, we reinterpret the problem through a group-covering (GC) lens, revealing similarities to group variants of TSP [4] and steiner tree (ST) [25]. In GC problems [22, 28], alongside a weighted graph $G = (V, E, c)$ we are also given a

set $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$ of $k \geq 1$ subsets (or "groups") of vertices $S_i \subseteq V$. The goal is to visit the vertices of V, according to some path constraints, such that at least one vertex in each set S_i is visited. To express GIP as a GC problem, recall that in GIP each vertex $v \in V$ covers a set of POIs $\chi(v) \subseteq \mathcal{P}$. We define the associated family of groups by *inverting* χ: for each $p \in \mathcal{P}$, define a set $S_p := \{v \in V \mid p \in \chi(v)\}$, and let $\mathcal{S} := \{S_p\}_{p \in \mathcal{P}}$. Here, S_p consists of all vertices from which POI p can be inspected, and inspecting p in a tour corresponds to visiting at least one vertex in S_p. Thus, GIP can be expressed as follows.

Definition 2 (Group-covering form of GIP). *Given $\langle G = (V, E, c), r, \mathcal{S} \rangle$, find a minimum-cost tour from r that covers all groups in $\mathcal{S}$.*

$$\underset{\tau \in \mathcal{T}(G,r)}{\arg\min}\ c(\tau) \quad \textit{subject to} \quad \forall\, S_i \in \mathcal{S} :\ V(\tau) \cap S_i \neq \emptyset.$$

This representation reveals a connection to two well-studied GC problems:

Definition 3 (Group-TSP [28]). *Given $\langle G = (V, E, c), r, \mathcal{S} \rangle$, let $\mathcal{T}_1(G, r) \subseteq \mathcal{T}(G, r)$ denote tours from r that visit any vertex in $V \setminus \{r\}$ at most once, find a minimum-cost tour in $\mathcal{T}_1(G, r)$ that covers $\mathcal{S}$.*

$$\underset{\tau \in \mathcal{T}_1(G,r)}{\arg\min}\ c(\tau) \quad \textit{subject to} \quad \forall\, S_i \in \mathcal{S} :\ V(\tau) \cap S_i \neq \emptyset.$$

Definition 4 (Group-ST [22]). *Given $\langle G = (V, E, c), r, \mathcal{S} \rangle$, find a minimum-cost tree $T \in \mathrm{Trees}(G, r)$, where $\mathrm{Trees}(G, r)$ is the set of all subtrees of G rooted in r, that covers all groups in $\mathcal{S}$.*

$$\underset{T \in \mathrm{Trees}(G,r)}{\arg\min}\ c(T) \quad \textit{subject to} \quad \forall\, S_i \in \mathcal{S} :\ V(T) \cap S_i \neq \emptyset.$$

Despite their apparent similarity, the path-simplicity constraint in Group-TSP introduces additional combinatorial complexity, distinguishing it from GIP, where vertex revisits are allowed. In contrast, Group-ST is structurally closer to GIP, as it emphasizes connectivity between the root and representative vertices of each group rather than tour simplicity. However, existing Group-ST approaches typically rely on reductions to classical Steiner Tree [21], which enlarge the problem and limit scalability. Additionally, their conversion to tours can yield poor inspection paths, especially in directed graphs. These observations motivate MILP formulations for GIP that, inspired by Group-ST, enforce root-to-group connectivity rather than explicit tour ordering. We discuss further relation between the problems in the extended version of our paper.

4 MILP Formulations for Graph Inspection Planning

Following the group-covering perspective (Sec. 3), we propose MILP formulations for GIP that enforce both group coverage and tour structure. We start with a baseline formulation that enforces coverage and local connectivity, and then introduce additional constraints to ensure global connectivity.

We denote the number of POIs by $|\mathcal{P}| = |\mathcal{S}| = k$, and set $K = \{1, \ldots, k\}$. Furthermore, $N^+(v) := \{u \in V : (v,u) \in E\}$ and $N^-(v) := \{u \in V : (u,v) \in E\}$ denote the out-neighborhood and in-neighborhood of vertex v, respectively.

4.1 Baseline MILP Formulation

The following is our baseline MILP formulation of GIP[1]. The inspection tour τ is specified through binary decision variables x_{uv} associated with each edge $(u,v) \in E$, i.e., $x_{uv} = 1$ indicates that τ traverses edge (u,v).

$$\min_{x_{uv}, \forall (u,v) \in E} \quad \sum_{(u,v) \in E} c(u,v) \cdot x_{uv} \tag{1a}$$

$$\text{s.t.} \quad \sum_{v \in N^+(r)} x_{rv} \geq 1, \tag{1b}$$

$$\sum_{v \in S_i} \sum_{u \in N^-(v)} x_{uv} \geq 1, \quad \forall S_i \in \mathcal{S}, \tag{1c}$$

$$\sum_{u \in N^-(v)} x_{uv} = \sum_{u \in N^+(v)} x_{vu}, \quad \forall v \in V, \tag{1d}$$

$$x_{uv} \in \{0,1\}, \quad \forall (u,v) \in E. \tag{1e}$$

Objective (1a) minimizes the total cost of the selected edges, and constraints (1b) and (1c) ensure that the corresponding tour will start from r and cover every group $S_i \in \mathcal{S}$, respectively.[2] Finally, constraints (1d) enforce that every vertex in the subgraph induced by the selected edges has equal in-degree and out-degree. As a result, the selected edges decompose into one or more edge-disjoint directed cycles [7]. Additional subtour-elimination constraints (SEC) are therefore required to ensure that all selected edges belong to a single connected tour containing the root. Before introducing our SECs, we briefly review the role of LP-relaxations in Branch-and-Bound (BnB) [29], and their interpretation for GIP.

Linear-programming (LP) relaxation. In BnB, the MILP's feasible region is recursively partitioned into a sequence of increasingly restricted subproblems. For each subproblem, an *LP relaxation* of the MILP, obtained by omitting integrality constraints (i.e., turning $x_{uv} \in \{0,1\}$ into $x_{uv} \in [0,1]$), is solved by an LP-solver. The objective value of this relaxed problem provides a lower bound for any feasible integer solution within the corresponding subproblem.

Flow-based interpretation. In GIP, the LP relaxation admits a natural flow interpretation, where each variable $x_{uv} \in [0,1]$ is viewed as assigning a fractional

[1] Although some instances involve only integer variables, we consistently use the term MILP rather than ILP, as our solution methodology does not distinguish between the two.

[2] This formulation permits revisiting vertices but explicitly forbids revisiting edges. However, the latter is not limiting: Mizutani et al. [34] (Lemma 1) prove that any optimal GIP tour traverses each directed edge at most once. Additionally, while outside the scope of this work, we note that this formulation can be extended to partial-coverage GIP [19, 34]. See App. B of the extended version of our paper.

amount of flow to edge (u, v). Feasible LP solutions therefore correspond to routing fractional flow that satisfies group-coverage, motivating our flow-based GIP formulations.

Integrality gap. In addition to lower bounds obtained from LP relaxations, a BnB solver maintains feasible *integer* solutions that provide upper bounds. Convergence is achieved as these bounds progressively tighten. The effectiveness of a MILP formulation is therefore governed by its *integrality gap*, defined as the difference between the LP optimum and the best integer-feasible solution. Large integrality gaps lead to weak bounds and slow convergence in BnB, highlighting the importance of strong subtour-elimination constraints [27].

4.2 Subtour Elimination Constraints (SEC) for GIP

Motivated by the flow-based interpretation of the LP relaxation, we introduce three families of subtour-elimination constraints (SEC), where the binary edge-selection variables x_{uv} serve as capacities for a directed network, while additional continuous flow-variables-based mechanisms are used to enforce global connectivity constraints.

Single-commodity flow (SCF). We consider a single flow originating from the root which is propagated along and consumed by any selected edges, thereby enforcing connectivity. This is inspired by similar approaches for TSP solvers [23]. Specifically, we introduce a nonnegative continuous flow variable $f_{uv} \geq 0$ for each directed edge $(u, v) \in E$, together with the following constraints:

$$f_{uv} \leq M \cdot x_{uv}, \quad \forall (u, v) \in E, \tag{2a}$$

$$\sum_{u \in N^-(v)} f_{uv} - \sum_{u \in N^+(v)} f_{vu} = \sum_{u \in N^+(v)} x_{vu}, \quad \forall v \in V \setminus \{r\}. \tag{2b}$$

Constraint (2a) ensures flow is permitted only on selected edges, where M is chosen sufficiently large to accommodate any required flow. Following Lemma 2 of Mizutani et al. [34], which (tightly) bounds the length of an optimal tour by $2 \cdot (|V| - 1)$, we set $M = 2 \cdot (|V| - 1)$. Constraint (2b) forms a flow-consumption rule, which eliminates subtours by requiring each traversal of a selected edge to consume one unit of flow. This is expressed from a vertex-centric perspective: A visit to a vertex v is induced by selecting an incoming edge x_{uv} for some $u \in N^-(v)$. Each such visit consumes one unit of flow, implying that the total incoming flow at v exceeds the total outgoing flow by exactly one unit per visit.

Lemma 1 (SCF constraints eliminate subtours). *An optimal solution for the MILP formulation defined by constraints* (1) *and* (2) *yields a single tour containing* r*.*

Proof. (sketch) By contradiction, assume that the optimal solution contains a directed subtour that does not visit the root r. Denote by $C \subseteq V$ the vertices visited by this subtour. Summing constraint (2b) over all the vertices in C, the left-hand side telescopes to zero, since every unit of flow that leaves a cycle-vertex enters another cycle-vertex, and the net-flow out of C is zero. In contrast,

the right-hand side sums to the number of edges in the cycle, which is strictly positive. This yields a contradiction, showing that such a cycle cannot satisfy the flow constraints. The only vertex excluded from constraint (2b) is the root, which is therefore allowed to exhibit a net flow imbalance, effectively acting as a flow source. Consequently, any feasible cycle must include the root. □

The SCF formulation augments the baseline with constraints (2), introducing $2|E|$ additional continuous variables and $O(|E| + |V|)$ constraints, and is the most compact MILP formulation we consider. However, for the LP-relaxed problem, the Big-M constraints in (2a) allow substantial flow on weakly selected edges, enabling the LP relaxation to satisfy connectivity constraints without committing to specific edges. This results in a large integrality gap, motivating tighter flow formulations that avoid Big-M constants, which we introduce next.

Mizutani et al. [34] proposed an alternative subtour-elimination scheme based on continuous *charge* variables, which enforces connectivity by maintaining a global charge balance with the root acting as a sink. Although asymptotically as compact as SCF, this formulation induces a different LP relaxation. We empirically compare their behavior within BnB in Sec. 6.

Multi-commodity flow (MCF). To overcome the large integrality gap that may be induced by constraint (2), we consider the following SEC mechanism, inspired by related routing and network-design problems [12, 38]. The key idea, depicted in Fig. 1, is to associate a distinct flow commodity with each group $i \in K$, and to require that one unit of flow be delivered from the root to a vertex covering that group. Formally, we introduce continuous flow variables $f^i_{uv} \in [0, 1]$ for each group $i \in K$ and edge $(u, v) \in E$. Each group i requires one unit of flow to originate at the root r.

$$\sum_{v \in N^+(r)} f^i_{rv} \geq 1, \qquad \forall i \in K, \tag{3a}$$

$$\sum_{v \in S_i} \sum_{u \in N^-(v)} f^i_{uv} - \sum_{v \in S_i} \sum_{u \in N^+(v)} f^i_{vu} \geq 1, \quad \forall i \in K, \tag{3b}$$

$$\sum_{u \in N^-(v)} f^i_{uv} = \sum_{u \in N^+(v)} f^i_{vu}, \qquad \forall i \in K,\ v \in V \setminus (S_i \cup \{r\}), \tag{3c}$$

$$f^i_{uv} \leq x_{uv}, \qquad \forall i \in K,\ (u, v) \in E. \tag{3d}$$

In the MCF formulation, each group requires one unit of flow from the root to a covering vertex, ensuring that any cycle carrying flow is connected to the root by flow conservation (3c). Cycles without commodity flow are irrelevant and can be removed without increasing cost, yielding a single structure rooted at the start vertex. By bounding flow directly with edge-selection variables (3d), MCF avoids Big-M constraints and produces a much tighter LP relaxation than SCF. However, this strength comes at a high cost: the formulation adds $2k|E|$ continuous variables, leading to tens of millions of variables in realistic instances [19, 20, 34], which makes it impractical beyond small scales (See App. D.2 of the extended version of our paper).

Consequently, we next move on to suggest a formulation that retains this connectivity strength without introducing an explicit flow commodity for each

group. While reducing the number of variables, it will introduce an exponential number of constraints. To avoid generating them explicitly, which will render the approach infeasible, we will take a lazy approach and generate them on demand.

Group-cutset formulation. We present a subtour-elimination mechanism based on explicit connectivity constraints, inspired by cutset-based approaches for TSP [3], which explicitly enforces root-to-group connectivity. This formulation aims to achieve the structural strength of root-to-group connectivity enforcement found in multi-commodity flow SEC, while avoiding its prohibitive memory requirements, thereby enabling application to large-scale GIP instances. We achieve this by introducing an exponential number of vertex-cut constraints that are *lazily evaluated* within a Branch-and-Cut framework. The effectiveness of this approach relies on the fact that these *group-cutset* constraints can be efficiently validated using network-flow based algorithms, as detailed in Sec. 5.

Formally, for any group $S_i \in \mathcal{S}$, we examine all partitions of the vertex set V into two disjoint subsets $R \subset V$ and $V \setminus R$ such that the root $r \in R$ and $R \cap S_i = \emptyset$. Such a partition defines a vertex cut between R and its complement. If a tour were to remain entirely within R, it would be impossible to visit any vertex in S_i, and thus the corresponding POI p_i could not be inspected. Therefore, any feasible GIP tour must include at least one edge crossing this vertex cut. Thus, we add constraints requiring the selected edges to cross every such cut R, enforcing connectivity between the root and each group S_i. Formally, let $\mathcal{R}$ denote the family of vertex subsets that contain r but exclude at least one group, i.e., $\mathcal{R} := \{R \subseteq V \mid r \in R \text{ and } \exists S_i \in \mathcal{S} \text{ such that } R \cap S_i = \emptyset\}$. For any $R \subseteq V$, we define its outgoing edge set as $\delta^+(R) := \{(u, v) \in E \mid u \in R,\ v \notin R\}$. The baseline formulation (1) is augmented with the following family of *group-cutset constraints*, requiring that for any $R \in \mathcal{R}$, at least one selected edge leaves R:

$$\sum_{(u,v)\in\delta^+(R)} x_{uv} \geq 1, \qquad \forall R \in \mathcal{R}. \tag{4}$$

The following lemma states that this constraint indeed eliminates subtours.

Lemma 2 (Group-cutset constraints eliminate subtours). *Assume $c(e) > 0$, for all $e \in E$.*[3] *The MILP formulation defined by constraints* (1) *and* (4) *yields a single tour containing r.*

Proof. Let $\{x^*\}$ be an optimal integral solution and set $F := \{e \in E \mid x^*_e = 1\}$. We show that F forms a single tour containing the root. Let $C_r \subseteq V$ denote the strongly-connected component of r in the subgraph induced by F, and let $F_r \subseteq F$ be the edges whose both ends are in C_r. Note that $|C_r| \geq 2$ due to constraint (1b). Define a new solution $\{\hat{x}\}$ by setting $\hat{x}_e = 1$ if $e \in F_r$ and $\hat{x}_e = 0$ otherwise and note that it can be easily shown that $\{\hat{x}\}$ is feasible.

If $F \neq F_r$, then $\{\hat{x}\}$ removes at least one selected edge. Since all edge costs are strictly positive, this strictly decreases the objective value, contradicting the

[3] If zero-cost edges are allowed, i.e., $c(e) \geq 0$, an optimal solution may contain additional disconnected components of zero total cost. Such components can be removed without affecting feasibility or optimality.

optimality of $\{x^*\}$. Therefore, $F = F_r$, and all selected edges lie in the connected component of r. □

As $\mathcal{R}$ grows exponentially with the graph size, explicitly enumerating the group-cutset formulation would render the solver intractable. However, given a candidate solution, constraints (4) can be efficiently *verified*, and for infeasible candidates, violated constraints can be identified using efficient flow-based algorithms. This property allows us to apply the *Branch-and-Cut* (BnC) approach, enabling a *lazy* evaluation of the constraints (4), hence avoiding explicitly enumerating them. As a result, the group-cutset formulation is particularly suited for large-scale instances, where explicit multi-commodity flow formulations are computationally infeasible and compact formulations yield poor lower bounds. To use BnC, we need to introduce several additional algorithmic building blocks which we now describe.

5 Branch-and-Cut Solver for Group-Cutset formulation

The Branch-and-Cut (BnC) framework extends Branch-and-Bound by incorporating *cutting planes*, i.e., additional constraints that are generated dynamically during the search. In the context of the group-cutset model introduced in Sec. 4.2, the BnC framework allows for a *lazy-constraint* evaluation. Specifically, the solver starts from a static partial formulation, and extends the formulation only with constraints that are relevant to the regions of the solution space explored by the solver, thus avoiding the overhead of enforcing constraints that are never active. Such dynamic constraint enforcement is performed by a *separation oracle*, which either identifies violated constraints or certifies that none exist. This allows BnC to operate on a compact formulation while retaining the strength of a much richer, potentially exponential constraint set. We present an oracle tailored to the group-cutset formulation in Sec. 5.1. A complementary component of the BnC framework is a *primal heuristic*, which transforms fractional LP relaxations into integer-feasible solutions that serve as upper bounds in the search process. While primal heuristics are standard in BnB-based MILP solvers, their problem-specific design is especially important in lazy-constraint formulations. In such settings, generic heuristics employed by solvers such as Gurobi [24] may produce solutions that satisfy the current formulation but violate constraints introduced later in the search. We therefore introduce a problem-specific primal heuristic that explicitly exploits the structure of GIP to provide valid solutions (Sec. 5.2).

5.1 Group-Cutset Separation Oracles

During the BnC search, partial problem formulations are LP-relaxed and solved, providing solution candidates $\{x^c\}$. The separation oracle, given such candidate, determines whether one of the group-cutset constraints (4) is violated, i.e., there exists a set $R \in \mathcal{R}$ as defined in Sec. 4.2, such that $\sum_{(u,v)\in\delta^+(R)} x^c_{uv} < 1$. If

such a set exists, the corresponding constraint is returned as a separating cut. We introduce two complementary separation oracles, along with a third hybrid oracle that combines their respective strengths.

Connectivity-based separation oracle. We first consider a fast separation oracle for verifying integral candidate solutions $\{x^c\}$. We construct a subgraph $G^c = (V, E^c)$ containing only the edges $(u, v) \in E$ for which $x^c_{uv} = 1$. We then compute the strongly connected component (SCC) $R^c \subseteq V$ of the root r in G^c, i.e., R^c contains any vertex $v \in V$ such that there exists a directed path from r to v, and v to r. For each group $S_i \in \mathcal{S}$, we check whether $R^c \cap S_i$ is empty. If so, the set R^c defines a violated group-cutset constraint, which is returned as a certificate. Otherwise, the oracle certifies that $\{x^c\}$ satisfies all group-cutset constraints. As the complexity of the SCC operation is linear in the number of edges [13] this oracle runs in $O(|E| + |\mathcal{S}| \cdot |V|)$ time and guarantees that no invalid integral solution is ever accepted by the solver. Although this oracle is computationally efficient, it generates only a single violated constraint per integer candidate solution, which may make the BnC solver struggle to develop meaningful problem representation.

Flow-based separation oracle. To strengthen constraint generation, we introduce a separation oracle applicable to both fractional and integral solutions, based on solving an s–t max-flow problem (Alg. 1). The key idea is to interpret the fractional values x^c_{uv} as capacities on directed edges and test whether the resulting network can route at least one unit of flow from the root r to a vertex covering a given group S_i. Under integrality of x^c_{uv}, such a flow corresponds to a path composed of selected edges. Formally, for each group $S_i \in \mathcal{S}$, we compute the maximum flow from r to S_i, as illustrated in Fig. 1. If the flow value is smaller than one, then no feasible solution can connect r to S_i using the current edge selections. By the max-flow min-cut theorem [15], this implies the existence of a cut separating r from S_i with total capacity less than one, yielding a violated group-cutset constraint that can be added as a separating inequality. While effective at tightening LP relaxations, this oracle is computationally expensive, as fully verifying the validity of a candidate solution requires solving a max-flow problem for each group. Using standard algorithms, Alg. 1 runs in $O(|\mathcal{S}| \cdot |V|^2 \cdot |E|)$ time [1, 16], motivating its use only in a limited manner within a hybrid approach.

Combined separation oracle. The two prior oracles offer complementary strengths. The connectivity-based oracle is computationally efficient and guarantees correctness by validating integral candidate solutions. However, it can generate at most a single separating constraint per integral solution, which limits its ability to substantially tighten the relaxation. In contrast, the flow-based oracle is capable of generating multiple effective separating constraints, including at fractional solutions, but its computational cost prevents it from being used to validate or rule out candidate solutions. We therefore propose a combined separation oracle that applies connectivity-based checks to all integer-feasible candidates and selectively applies flow-based separation to a *uniformly sampled subset* of groups at fractional solutions. This balances relaxation strength and

Algorithm 1 Flow-based separation oracle ($\{x^c\}$, $G = (V, E)$, $\mathcal{S}$, r)

Initialize an empty set $\mathcal{C}$ of cuts
for each group $S_i \in \mathcal{S}$ **do**
  Define flow network $\mathcal{N}_i = (V_i, E_i, \kappa)$:
    $V_i = V \cup \{t\}$ ▷ t is an auxiliary sink vertex for S_i
    Generate auxiliary edges $A_i \leftarrow \{(v, t) : v \in S_i\}$ with $\kappa(v, t) \leftarrow 1$.
    $E_i = E \cup A_i$.
    Update capacities for E edges $\kappa(u, v) \leftarrow x^c_{uv}$ for any $(u, v) \in E$.
  Compute a minimum r–t cut in $\mathcal{N}_i$: $(R_i, (V \cup \{t\}) \setminus R_i)$.
  if $\sum_{(u,v)\in\delta^+(R_i)} x^c_{uv} < 1$ **then**
    Add the violated constraint $\sum_{(u,v)\in\delta^+(R_i)} x_{uv} \geq 1$ to $\mathcal{C}$.
return $\mathcal{C}$

computational cost while preserving correctness. Importantly, sampling introduces no loss of correctness: although violated constraints for unsampled groups may be missed at fractional solutions, any integral solution is ultimately validated by the connectivity oracle. As shown in Sec. 6.3, this combined approach yields strong cuts at low computational cost, significantly improving convergence while maintaining correctness guarantees. Furthermore, as we will demonstrate empirically, the method is insensitive to the exact value of the group sample size.

5.2 Primal Heuristic for GIP

We present a problem-specific primal heuristic for GIP. While applicable to all SEC variants (Sec. 4), it is particularly important for lazy-constraint formulations. In such settings, generic MILP heuristics (e.g. [6, 14, 18]) may generate solutions that satisfy only the currently enforced constraints, yet violate constraints that will be introduced later by the separation oracle. This issue is most pronounced in the early stages of the search, when the formulation is still highly partial, making incumbent generation unreliable and motivating a heuristic that is explicitly aware of the full GIP structure.

Our heuristic is tightly integrated with the BnC search and exploits information from the current LP relaxation. At a high level, it consists of three phases. First, edge costs are modified using the fractional LP solution $\{x_e\}_{e\in E}$ by defining $c_n(e) := c(e) \cdot (1 - x_e)$, which biases the search toward edges favored by the relaxation. Second, a group-covering tree rooted at the start vertex is greedily constructed in the discounted graph, incrementally connecting the root to a representative vertex of an uncovered group while ensuring connectivity. Third, the resulting tree is augmented and traversed to produce a valid GIP tour. Rather than doubling tree edges, we add a minimum-weight matching over odd-degree tree vertices, yielding an Eulerian subgraph whose traversal produces a lower-cost tour [11]. This heuristic reliably generates high-quality incumbent solutions early in the BnC search, leading to substantially improved upper bounds and faster convergence, and may be considered as a *problem-aware rounding mechanism* applied on fractional LP solutions. A complete algorithmic description and empirical evaluation are provided in the extended version.

6 Experimental Results

We empirically evaluate how different MILP formulations and their associated algorithmic components for GIP trade off scalability and solution-quality guarantees. We report two quantities that BnC solvers maintain throughout the search: an upper bound c_{UB}, given by the cost of the best *incumbent* feasible solution, and a lower bound c_{LB}, obtained from LP relaxations of the MILP. Another measure we report is the *optimality gap*, defined as $\mathrm{Gap} := 100 \cdot (c_{\mathrm{UB}} - c_{\mathrm{LB}})/c_{\mathrm{UB}}$, which quantifies the practical tightness of the solver's near-optimality certification. This measure is particularly important given the provable hardness of approximating GIP (see extended version).

We conduct our evaluation by progressing from medium-scale real-world benchmarks to larger-scale controlled simulated experiments, and finally to targeted ablation studies, in order to examine the effects of formulation and solver design. Across all settings, the proposed Group-Cutset formulation consistently produces stronger lower bounds than the other tested formulations, and exhibits slower performance degradation on larger instances. In the extended version, we complement these results with an evaluation of small-scale instances, and an ablation study of primal-heuristic design.

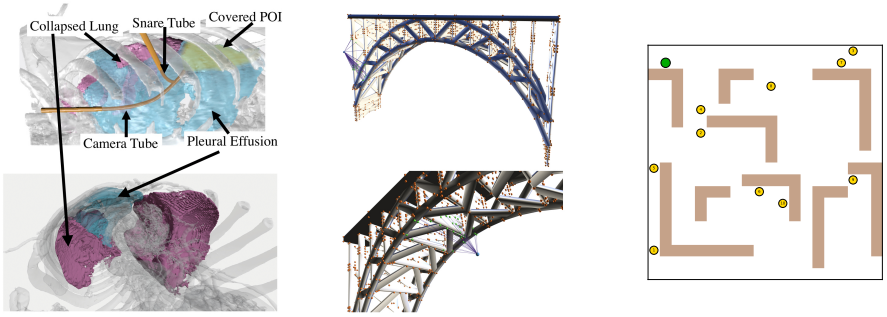
6.1 Evaluation Scenarios

Our evaluation uses both real-world inspection datasets and controlled simulated scenarios, to study solver behavior across varying graph sizes and numbers of POIs. The `CRISP` scenario [2, 31], adapted from [19], models medical inspection with a continuum robot tasked with inspecting 4,203 POIs in a confined anatomical cavity, yielding GIP instances with dense and highly overlapping coverage groups. The `Bridge` scenario, adapted from [20], considers aerial inspection of 3,346 POIs distributed across a large structure, requiring inspection plans that connect distant regions of the roadmap. `Controlled` scenarios complement these benchmarks, enabling systematic evaluation across a wide range of problem scales using a simplified planar point-robot simulator. Experiment scenarios are illustrated in Fig. 2.

Experiment hardware and simulator implementation details can be found in the extended version of this paper. Our code is publicly available in the GIP repository.

6.2 Comparative Evaluation of MILP Formulations

We compare the performance of the different MILP formulations, i.e., the baseline MILP detailed in Sec. 4.1 with the three different SEC detailed in Sec. 4.2, which we refer to as SCF, MCF and Group-Cutset together with the charge-variables based formulation of Mizutani et al. [34], which we refer to as Charge. The compact formulations (SCF, Charge) are solved within the BnB framework while the Group-Cutset formulation is solved using the BnC framework with the combined separation oracle using a group sample size of 100, accompanied by



(a) `CRISP` (b) `Bridge` (c) `Controlled`

Fig. 2: Experimental evaluation scenarios. (a) CRISP robot, composed of a snare-tube and a camera-tube, inside the cavity of a patient's lungs. (b) Drone inspecting POIs marked on a bridge structure. (c) Planar point robot in a maze, starting at the green point and tasked with inspecting all POIs (yellow).

our developed GIP primal heuristic. For each variant, we report $c_{\mathrm{UB}}, c_{\mathrm{LB}}$, and the optimality gap Gap as a function of each algorithm's running time. Due to its explicit large representation, the MCF formulation exceeds available memory at scales evaluated here, and is therefore omitted from these experiments and is discussed in small scale experiments in the extended version.

Medium-size Instances. We begin with medium-sized scenarios (`CRISP` and `Bridge`), with a time limit of 1,000 seconds per instance. Roadmaps are constructed using IRIS-CLI [20] with 1,000 and 2,000 vertices, yielding graphs with over 20,000 and 40,000 edges, respectively. Results are shown in Fig. 3.

Across all evaluated instances, Group-Cutset consistently produces tighter lower bounds than the compact formulations, an effect that is particularly pronounced in the `CRISP` scenarios, where its lower bounds continue to improve throughout the run while those of compact formulations stabilize early. In contrast, compact formulations, most notably SCF, tend to yield better incumbent solutions. Compared to the prior MILP solver Charge, SCF achieves better upper and lower bounds on both 1,000-vertex instances, whereas Charge performs best on the `Bridge-2000` instance. These results indicate instance-dependent behavior. While the Group-Cutset primal heuristic performs competitively on `CRISP`, it is weaker on `Bridge`, suggesting limited generality of the current problem-specific heuristic (Sec. 5.2). In contrast, the fully instantiated compact formulations allow Gurobi to exploit a broader set of generic primal heuristics, enabling better adaptation to different instance structures. Overall, these complementary strengths suggest that hybrid approaches combining strong incumbent generation from compact formulations with the tighter lower bounds of Group-Cutset may be a promising direction for future work.

Large Instances. The medium-sized GIP instances correspond to relatively small roadmap sizes when compared to those encountered in realistic motion-planning problems [36]. We therefore turn to simulated GIP instances with substantially larger graphs and numbers of POIs to explicitly stress solver scalability. Time limits are set to 500 seconds per instance. Solver setups are identical to

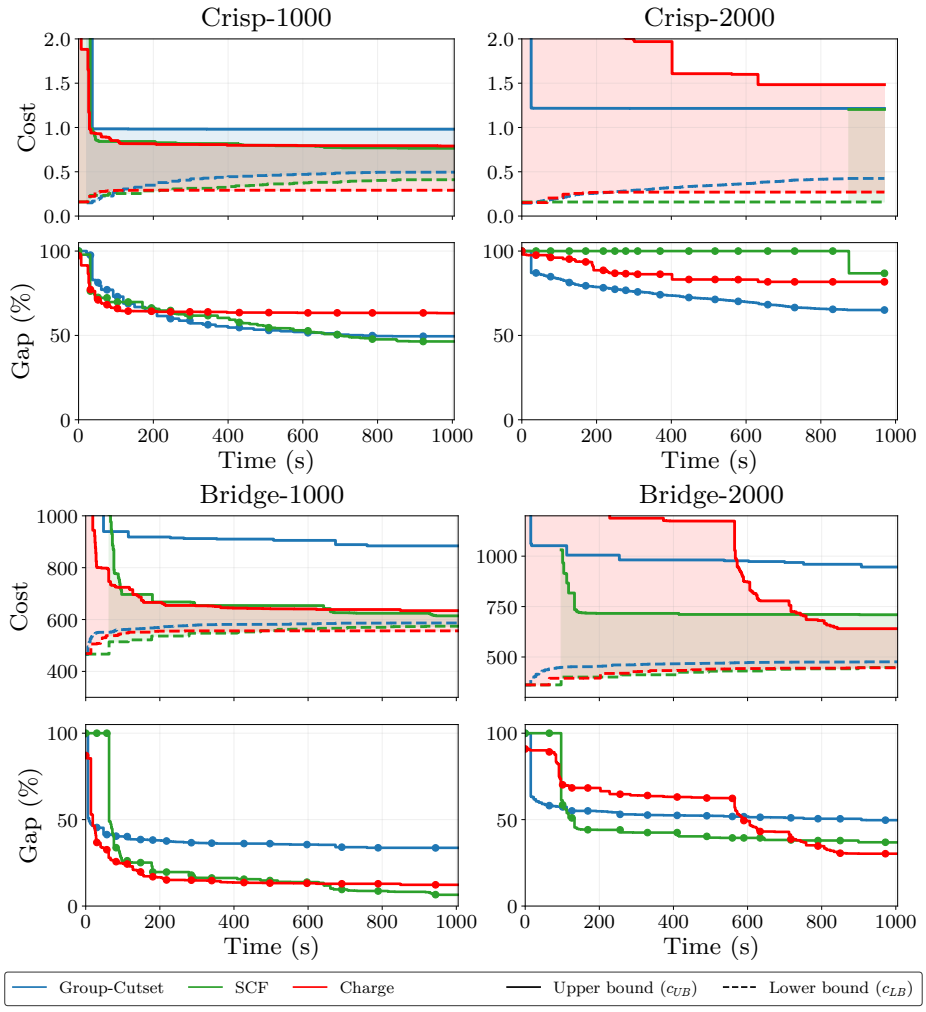


Fig. 3: Solvers evaluation on real-world instances.

those used in the previous experiments, with the exception that the compact formulations (SCF, Charge) are augmented with our GIP heuristic.[4]

Fig. 4 presents the optimality gap as a function of the number of POIs for different roadmap sizes. Group-Cutset maintains substantially smaller optimality gaps as instance size grows, whereas Charge exhibits limited gap reduction across all tested scales. The SCF formulation shows moderate improvements on smaller instances but degrades rapidly as problem size increases. This behavior reflects fundamental differences in how global connectivity is enforced in the corresponding LP relaxations. Both SCF and Charge rely on global flow or charge-balance

[4] This heuristic was added since Gurobi's internal heuristics struggled to produce feasible solutions within the time limits for the large instances considered. All reported results improved when this heuristic was included.

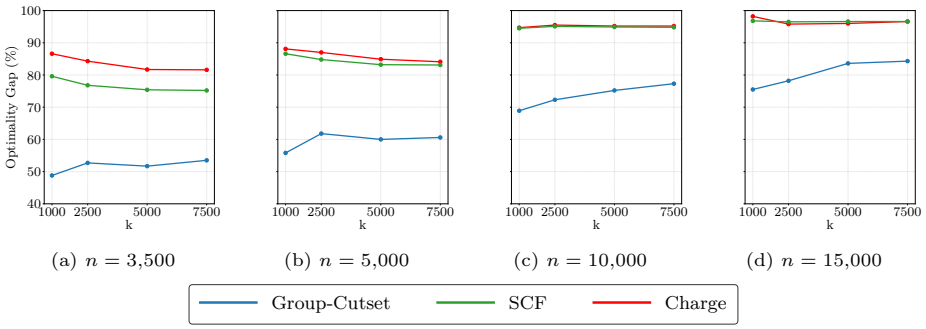


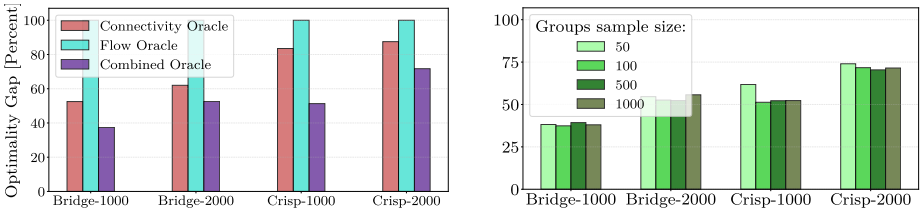
Fig. 4: Optimality gap on large-scale instances after 500 s for varying graph sizes (n) and number of POIs (k).

constraints to eliminate subtours [23, 38], which allow fractional solutions to satisfy connectivity by spreading flow across many weakly selected edges. As the graph size grows, such fractional connectivity patterns lead to increasingly weak lower bounds [32, 34]. In contrast, Group-Cutset enforces connectivity through explicit root-to-group cut constraints. Any feasible solution, fractional or integral, must allocate sufficient total edge capacity across cuts separating the root from each POI group. As the number of POIs increases, the growing number of such constraints progressively tightens the LP relaxation, explaining the more stable optimality gaps observed for Group-Cutset on large instances.

6.3 Separation Oracle Design Evaluation

We evaluate the impact of separation-oracle design on lower-bound strength and solver convergence of the Group-Cutset BnC solver (Sec. 5.1). We consider three variants: (i) a *connectivity-based oracle*; (ii) a *flow-based oracle* applied exhaustively to all groups; (iii) a *combined oracle* that validates integral candidates via connectivity checks and applies flow-based separation to a uniformly sampled subset of groups at fractional solutions.

Fig. 5(a) compares the final optimality gaps obtained using these variants on real-world instances. Using only the connectivity-based oracle (crimson) yields large final gaps across all instances, indicating a weak LP relaxation in which fractional solutions remain poorly constrained, leading to weak lower bounds and slow convergence. In contrast, the combined oracle (purple) substantially strengthens the formulation, consistently reducing the final optimality gap by approximately 10–30% relative to connectivity-only separation. This improvement stems from the ability of the flow-based approach to generate effective constraints, even when validating only a small sample of groups. We also evaluated a standalone flow-based oracle (turquoise bars in Fig. 5(a)) that performs exhaustive root-to-group separation. Although strong in principle, this approach is computationally prohibitive: validating a single candidate requires solving thousands of max-flow problems, leading to excessive separation time and negligible progress. As a result, exhaustive flow-based separation does not scale to large GIP instances. Fig. 5(b) examines the effect of group-sampling size in the combined



(a) Comparison between separation oracles.

(b) Effect of group sampling size.

Fig. 5: Ablation study for the separation oracles reporting the final optimality gaps after 500 s on the real-world instances. (a) Comparison of connectivity-only, flow-only, and combined separation oracles (using a group sample size of 100). (b) Effect of group sampling size for the combined oracle.

oracle. Final optimality gaps vary by only a few percentage points across a wide range of values, indicating low sensitivity to this parameter and enabling effective separation with modest samples. Taken together, these results highlight the importance of separation-oracle design for scalability, with the combined oracle achieving a practical balance between strength and efficiency.

7 Conclusion and Future Work

In this work, we have positioned the graph inspection planning (GIP) problem within the broader context of graph-based optimization, highlighting its algorithmic connections to Steiner tree, TSP, and network flow. Leveraging this perspective, we developed and analyzed three distinct MILP formulations, with our primary contribution being a scalable Branch-and-Cut solver centered on the Group-Cutset formulation. More broadly, our results suggest that lazy formulations within a Branch-and-Cut framework can provide a powerful paradigm for graph-based planning problems with capacity constraints, priority structures, or specialized objectives.

Several promising directions remain. For GIP, our results show substantial variation in solver performance across problem settings. Understanding the sources of this variation—particularly the role of inspection structure and the locality of sensor–POI visibility—could help develop solvers tailored to specific high-impact use cases. From a combinatorial optimization perspective, an important direction is to better characterize the trade-off between motion-planning fidelity and optimization complexity, since increasingly dense roadmaps improve geometric accuracy but can complicate approximation quality and optimality certification. We also aim to strengthen the solver by incorporating advanced MILP techniques, including additional classes of cutting planes and primal heuristics. Finally, the proposed framework could be extended to richer settings such as multi-robot inspection planning, partial and adaptive inspection objectives, and online or incremental formulations.

Acknowledgments. Large language models (ChatGPT and Gemini) were used for light editing and grammar refinement, as well as limited assistance with preliminary literature exploration and figure presentation and formatting.

Bibliography

[1] Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. *Network flows: theory, algorithms and applications*. Prentice hall, 1994.

[2] Patrick L Anderson, Arthur W Mahoney, and Robert James Webster. Continuum reconfigurable parallel robots for surgery: Shape sensing and state estimation with uncertainty. *IEEE robotics and automation letters*, 2(3): 1617–1624, 2017.

[3] David Applegate, Robert Bixby, Vašek Chvátal, and William Cook. TSP cuts which do not conform to the template paradigm. In *Computational Combinatorial Optimization: Optimal or Provably Near-Optimal Solutions*, pages 261–303. Springer, 2001.

[4] David L Applegate, Robert E Bixby, Vašek Chvátal, and William J Cook. The traveling salesman problem: a computational study. In *The Traveling Salesman Problem*. Princeton university press, 2011.

[5] Prasad N Atkar, Aaron Greenfield, David C Conner, Howie Choset, and Alfred A Rizzi. Uniform coverage of automotive surface patches. *The International Journal of Robotics Research*, 24(11):883–898, 2005.

[6] Livio Bertacco, Matteo Fischetti, and Andrea Lodi. A feasibility pump heuristic for general mixed-integer problems. *Discrete Optimization*, 4(1): 63–76, 2007.

[7] John Adrian Bondy and Uppaluri Siva Ramachandra Murty. *Graph theory with applications*. north-Holland, 1979.

[8] Peng Cheng, James Keller, and Vijay Kumar. Time-optimal UAV trajectory planning for 3D urban structure coverage. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2750–2757. IEEE, 2008.

[9] Brian Y Cho and Alan Kuntz. Efficient and accurate mapping of subsurface anatomy via online trajectory optimization for robot assisted surgery. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15478–15484. IEEE, 2024.

[10] Brian Y Cho, Tucker Hermans, and Alan Kuntz. Planning sensing sequences for subsurface 3D tumor mapping. In *2021 international symposium on medical robotics (ISMR)*, pages 1–7. IEEE, 2021.

[11] Nicos Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. In *Operations Research Forum*, volume 3, page 20. Springer, 2022.

[12] A Claus. A new formulation for the travelling salesman problem. *SIAM Journal on Algebraic Discrete Methods*, 5(1):21–25, 1984.

[13] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, Cambridge, MA, USA, 3 edition, 2009.

[14] Emilie Danna, Edward Rothberg, and Claude Le Pape. Exploring relaxation induced neighborhoods to improve MIP solutions. *Mathematical Programming*, 102(1):71–90, 2005.

[15] George Dantzig and Delbert Ray Fulkerson. On the max flow min cut theorem of networks. *Linear inequalities and related systems*, 38:225–231, 2003.

[16] Jack Edmonds and Richard M Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM (JACM)*, 19 (2):248–264, 1972.

[17] Uriel Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.

[18] Matteo Fischetti, Fred Glover, and Andrea Lodi. The feasibility pump. *Mathematical Programming*, 104(1):91–104, 2005.

[19] Mengyu Fu, Alan Kuntz, Oren Salzman, and Ron Alterovitz. Toward asymptotically-optimal inspection planning via efficient near-optimal graph search. *Robotics science and systems: online proceedings*, 2019:10–15607, 2019.

[20] Mengyu Fu, Oren Salzman, and Ron Alterovitz. Computationally-efficient roadmap-based inspection planning via incremental lazy search. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7449–7456. IEEE, 2021.

[21] Gerald Gamrath, Thorsten Koch, Stephen J Maher, Daniel Rehfeldt, and Yuji Shinano. Scip-jack—a solver for STP and variants with parallelization extensions. *Mathematical Programming Computation*, 9(2):231–296, 2017.

[22] Naveen Garg, Goran Konjevod, and Ramamoorthi Ravi. A polylogarithmic approximation algorithm for the group Steiner tree problem. *Journal of Algorithms*, 37(1):66–84, 2000.

[23] Bezalel Gavish and Stephen C Graves. The travelling salesman problem and related problems. Technical Report OR-078-78, Massachusetts Institute of Technology, Operations Research Center, 1978.

[24] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2024.

[25] Frank K. Hwang, Dana S. Richards, and Pawel Winter. *The Steiner Tree Problem*. Elsevier, 1992.

[26] Richard M. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, pages 85–103, 1972.

[27] Ailsa H Land and Alison G Doig. An automatic method for solving discrete programming problems. In *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art*, pages 105–132. Springer, 2009.

[28] Gilbert Laporte and Yves Nobert. Generalized travelling salesman problem through n sets of nodes: an integer programming approach. *INFOR: Information Systems and Operational Research*, 21(1):61–75, 1983.

[29] Eugene L Lawler and David E Wood. Branch-and-bound methods: A survey. *Operations research*, 14(4):699–719, 1966.

[30] Eugene L. Lawler, Jan Karel Lenstra, Alexander H. G. Rinnooy Kan, and David B. Shmoys. *The Traveling Salesman Problem*. Wiley, 1985.

[31] Arthur W Mahoney, Patrick L Anderson, Philip J Swaney, Fabien Maldonado, and Robert J Webster. Reconfigurable parallel continuum robots for incisionless surgery. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4330–4336. IEEE, 2016.

[32] Clair E Miller, Albert W Tucker, and Richard A Zemlin. Integer programming formulation of traveling salesman problems. *Journal of the ACM (JACM)*, 7(4):326–329, 1960.

[33] John E Mitchell. Branch-and-cut algorithms for combinatorial optimization problems. *Handbook of applied optimization*, 1(1):65–77, 2002.

[34] Yosuke Mizutani, Daniel Coimbra Salomao, Alex Crane, Matthias Bentert, Pål Grønås Drange, Felix Reidl, Alan Kuntz, and Blair D Sullivan. Leveraging fixed-parameter tractability for robot inspection planning. *arXiv preprint arXiv:2407.00251*, 2024.

[35] Manfred Padberg and Giovanni Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM review*, 33(1):60–100, 1991.

[36] Itai Panasoff and Kiril Solovey. Effective sampling for robot motion planning through the lens of lattices. In *Robotics: Science and Systems*, 2025.

[37] Sartaj Sahni and Teofilo Gonzalez. P-complete approximation problems. *Journal of the ACM*, 23(3):555–565, 1976.

[38] Richard T Wong. Integer programming formulations of the traveling salesman problem. In *Proceedings of the IEEE international conference of circuits and computers*, volume 149, page 152. IEEE Press Piscataway NJ, 1980.