Spanning-tree based coverage for a tethered robot

Xiao Peng, François Schwarzentruber, Olivier Simonin, Christine Solnon

Spanning-tree based coverage for a tethered robot

Xiao Peng[1], François Schwarzentruber[2], Olivier Simonin[1] and Christine Solnon[1]

***Abstract*—Tethered robots find widespread application in underwater and disaster recovery missions. This study focuses on the coverage path planning (CPP) problem for a tethered robot, considering cable constraints and the presence of forbidden areas in the environment. We propose adapting the spanning tree-based coverage algorithm to address CPP. Theoretical complexity analysis reveals NP-completeness in cases involving forbidden areas. We show how to solve CPP by searching for a tree in a configuration graph, and how to reduce the size of this graph to compute approximate solutions faster. We introduce Integer Linear Programming (ILP) models corresponding to these approximations and experimentally compare them on various instances.**

***Index Terms*—Coverage Path Planning, Tethered Robot, Computational Complexity.**

I. Introduction

COVERAGE Path Planning (CPP) is a fundamental problem in robotics that aims at finding a shortest cycle that fully covers a workspace for a mobile robot of a specified size. This problem has many applications such as automatic floor cleaning and area patrol [1]. In this paper, we study the CPP problem for a particular type of mobile system, a *tethered robot*, which plays a crucial role in some challenging contexts, where the cable provides stable access to electricity and network connectivity [2], [3], [4]. This new variant is termed TCPP for tethered CPP. Compared to classical CPP, the main challenge is related to cable constraints, wherein the cable has limited length and cannot be crossed by the robot. The approach to addressing these constraints depends on assumptions about the cable. Here, we consider the case where the cable is kept taut by a recoil system that adjusts it as the robot moves. Additionally, we explore scenarios where the workspace contains forbidden areas, barring the robot and its cable from passage. This prohibition stems from potential damage to the robot and cable or the presence of individuals susceptible to the robot or cable.

A. Related Work

To solve CPP problems, the workspace is usually discretized into a 4-connected grid graph g such that each grid cell has the same size as the robot [5], [6]. A Hamiltonian cycle in g represents the shortest possible solution, with each cell traversed precisely once. In [7], it is shown that the complexity of deciding the existence of a Hamiltonian cycle depends on g: it is in $\mathcal{O}(1)$ for obstacle-free rectangular grids, yet $\mathcal{NP}$-complete in the presence of obstacles. In this latter case, Spanning Tree Coverage (STC) offers a polynomial-time approximate solution, as described in [6]: STC involves constructing a graph G_4 whose vertices correspond to non-overlapping groups of 2×2 adjacent cells in g; if each cell of g belongs to exactly one 2×2 cell group, then there exists a Hamiltonian cycle in g if and only if G_4 is connected and, given a spanning tree T of G_4, a Hamitonian cycle in g can be constructed by circumnavigating T; when some cells of g are not covered by a vertex of G_4, an approximate solution may be built by completing the circumnavigation of T with a sub-cycle that visits each uncovered cell twice as shown in Fig. 1(a).

Existing tethered coverage algorithms can be broadly categorized as online or offline, depending on the level of knowledge about the environment. Offline approaches operate under the assumption of complete environmental knowledge and rely on how the environment is modelled. In [8], a decomposition of the environment into "split cell" and "corridor" structures is proposed. Split cells are maintained in a stack, and upon reaching a new split cell, the robot performs complete coverage of the corresponding corridor. Once coverage is complete, the robot proceeds to the next split cell until the environment is fully covered. In this method, solution optimality depends on environment characteristics. Alternatively, [9] uses a specific decomposition of the environment derived from "Morse-based Cellular Decomposition" [10]: the grid map is divided into interconnected rectangular shapes around obstacles, covered using a zig-zag motion pattern. However, their primary focus is on covering all the accessible areas, rather than minimizing the overall coverage path length. The online version of this problem, explored in [8], [11] assumes no prior environmental knowledge. In this setting, the environment is incrementally explored while simultaneously constructing a tree map to track frontiers of uncovered areas. [8] proposes an approximation algorithm with a factor $\frac{2L}{D}$ compared to the minimum path length for coverage, and it is further improved to $2(1 - \frac{1}{N})$ in [11], where L represents cable length, D denotes cell size, and N indicates the number of accessible obstacle-free cells.

B. Outline and contributions of the paper

The primary contribution of this work is to investigate the interest of using STC to solve offline TCPP. Indeed, in case of a tethered robot, STC ensures that the cable is fully retracted

Manuscript received: July 9, 2024; Revised: November 14, 2024; Accepted: December 20, 2024.

This paper was recommended for publication by Editor Hanna Kurniawati upon evaluation of the Associate Editor and Reviewers'comments. This work was supported by the European Commission under the H2020 project BugWright2 (871260): Autonomous Robotic Inspection and Maintenance on Ship Hulls and Storage Tanks.

[1]X. Peng, O. Simonin and C. Solnon are with CITI lab, INSA Lyon, Inria, F-69621 Villeurbanne, France. firstname.lastname@insa-lyon.fr

[2]F. Schwarzentruber is with Université de Lyon, ENS Lyon, UCB Lyon 1, CNRS, LIP, France. francois.schwarzentruber@ens-lyon.fr

Digital Object Identifier (DOI): see top of this page.

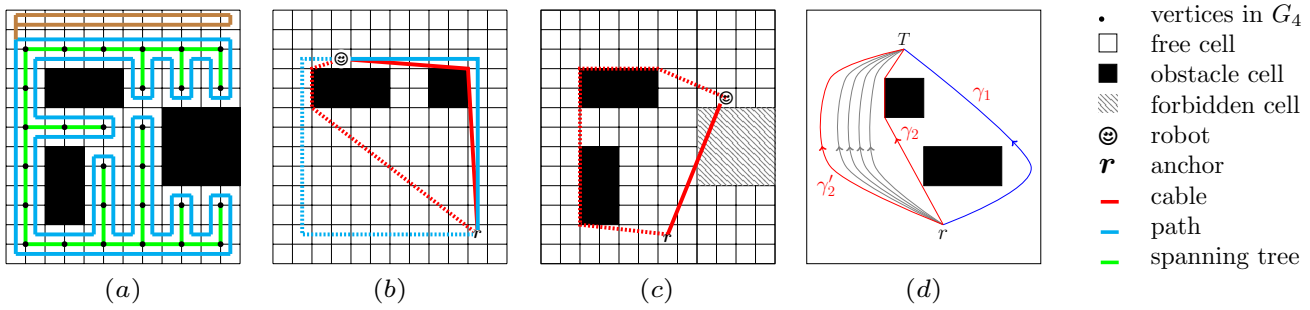


Fig. 1. (a): Illustration of STC. Vertices of G_4 are represented by black points (at the center of 2×2 free cells). In this example, G_4 does not cover the top row of g. A spanning tree T of G_4 is shown in green and the path π_T obtained by circumnavigating T is displayed in blue. π_T does not visit the free cells of g that are not covered by G_4 (top row). π_T may be completed by visiting twice each of these free cells, as displayed in brown. (b): The blue solid path is smaller than the blue dashed path, but the cable configuration associated with the solid path is longer than the cable configuration associated with the dashed path. (c): The cable configuration displayed with a red solid line is not valid because it crosses the forbidden area. The robot must bend its cable around the top obstacle to prevent it from crossing the forbidden area. (d): Curves γ_2 and γ_2' are homotopic, while γ_2 and γ_1 are not. The curve γ_2 is the shortest path within its homotopy class and it corresponds to a cable configuration.

when the robot returns to its initial position, which is not the case of arbitrary cycles that may encircle obstacles, as illustrated in Fig. 2. An extension of STC is proposed to address *cable constraints* by restricting the tree structure in the solution, which differs from the classic STC. Based on this concept, we show in Section III that TCPP can be solved in polynomial time by adapting Dijkstra's algorithm, when there is no forbidden area. A major contribution of our work is studying the case with *forbidden areas*. Specifically, in Section IV, we demonstrate that TCPP becomes $\mathcal{NP}$-complete by providing a reduction from Separable Planar 3-SAT. In Section V, we show how to solve TCPP with forbidden areas by searching for a tree in a graph that represents all possible cable configurations. To improve scale-up properties, Section VI introduces an approximate solution by removing some edges in the graph, along with an Integer Linear Programming (ILP) model to search for a tree in this filtered graph. In Section VII, we show how to further reduce the graph by merging some nodes. Section VIII provides a qualitative analysis on simulated instances discussing the factors contributing to problem hardness.

II. Problem Statement

We consider a discrete workspace defined by a 4-connected grid graph g composed of square cells which have the same size as the robot's footprint. The graph g contains 3 different kinds of cells: obstacle cells that cannot be crossed by robots and around which cables may bend, forbidden cells that cannot be crossed by robots nor by cables, and free cells that may be crossed by robots and cables. We note r the free cell from which the robot starts its path and to which its cable is anchored. The cell r is called the *anchor*.

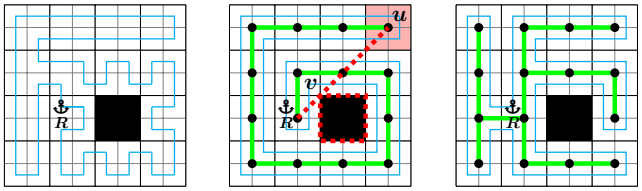
A path of length k is a sequence of $k+1$ free cells $\pi = \langle c_0, \dots, c_k \rangle$ such that $c_0 = r$ and $\forall i \in [1, n]$, c_i is adjacent to c_{i-1} in g. Given a path π followed by a robot, its cable configuration may be different from π as the cable is kept taut by a recoil system: the cable configuration associated with π is the sequence $cc_\pi = \langle c_0, v_1, \dots, v_l, c_k \rangle$ such that $\forall i \in [1, l], v_i$ is a vertex of an obstacle and cc_π is the shortest path homotopic to π, where two paths are homotopic if there exists a continuous deformation between them without crossing obstacles [12], as illustrated in Fig. 1(d). The (Euclidean) length of cc_π is denoted $\|cc_\pi\|$. A cable configuration cc_π is *valid* if (i) cc_π is not self-crossing, (ii) $\|cc_\pi\|$ does not exceed the length of the cable, and (iii) cc_π does not cross any forbidden cell. A path $\pi = \langle c_0, \dots, c_k \rangle$ is *valid* if $\forall i \in [1, k]$, the cable configuration $cc_{\langle c_0, \dots, c_i \rangle}$ is valid.

The TCPP problem aims to find a shortest valid path $\pi = \langle c_0, \dots, c_k \rangle$ that starts from and ends on r (*i.e.*, $c_0 = c_k = r$), visits each free cell at least once, and such that the cable is fully retracted when the robot arrives on c_k (*i.e.*, $cc_\pi = \langle r \rangle$).

To solve TCPP with an STC-based approach, we introduce the cell group graph $G_4 = (V_4, E_4)$ such that each vertex in V_4 is the center of a group of 2×2 free cells of g, where E_4 corresponds to adjacency relations between these cell groups, and each free cell belongs to at most one cell group of V_4. In this work, we assume that the grid graph g can be perfectly transformed into a cell group graph G_4, *i.e.*, each free cell belongs to exactly one cell group of V_4. The vertex of V_4 associated with the cell group that contains the anchor r is called the root and is denoted by R.

A *covering tree* of a graph $G = (V, E)$ is a connected graph $T = (V_T, E_T)$ such that $V_T \subseteq V$, $E_T \subseteq E \cap V_T \times V_T$, and $|E_T| = |V_T| - 1$. Given a covering tree $T = (V_T, E_T)$ of G_4 such that $R \in V_T$, we may obtain a cycle of g by circumnavigating T, starting from and ending on r, as illustrated in Fig. 1(a). This cycle, denoted π_T, always satisfies the property that the cable is fully retracted at the end, whereas this is not the case of all cycles, as illustrated in Fig. 2(a). This is a main motivation to solve TCPP by computing covering trees. However, other cable-related constraints may prevent us from finding a covering tree that covers all vertices of G_4, even if G_4 is connected, because the length of the cable must not be exceeded, and the cable cannot cross itself or forbidden areas, as illustrated in Fig. 2(b). Hence, we propose to solve TCPP by computing a Maximum Covering Tree (MCT).

Definition II.1 (MCT)**.** Given a cell group graph G_4, an MCT is a covering tree $T = (V_T, E_T)$ of G_4 such that π_T is valid, $R \in V_T$, and $|V_T|$ is maximum.



(a) (b) (c)

Fig. 2. (a): Example of Hamiltonian cycle in g that encircles an obstacle. (b)(c): When the Hamiltonian cycle (in blue) is obtained by circumnavigating a spanning tree of G_4 (in green) the cable is always completely retracted when the robot returns to its initial position. The cycle in (b) does not avoid cable crossing when the robot is on u (as shown by the dashed line). The cycle in (c) is the one computed by Algo. 1 and it avoids cable entanglements.

Algorithm 1: MCT(G_4, R, ℓ)

Input: A graph $G_4 = (V_4, E_4)$, the root R and the maximal cable length ℓ
Output: An MCT $T = (V_T, E_T)$

1 **for** *each vertex* $u \in V_4$ **do** $d[u] \leftarrow +\infty$;
2 $d[R] \leftarrow 0$; $V_T \leftarrow \emptyset$; $E_T \leftarrow \emptyset$
3 **while** *there exists* $u \in V_4 \setminus V_T$ *such that* $d[u] < +\infty$ **do**
4 let u be the vertex of $V_4 \setminus V_T$ such that $d[u]$ is minimal
5 add u to V_T
6 **if** $u \neq R$ **then** add $(p[u], u)$ to E_T;
7 **for** *each edge* $(u, v) \in E_4$ *such that* $v \notin V_T$ **do**
8 let π_u be the path from R to u in the tree $T = (V_T, E_T)$
9 let π_v be the path obtained by adding v at the end of π_u
10 **if** $\|cc_{\pi_v}\| \leq \ell$ *and* $\|cc_{\pi_v}\| < d[v]$ **then**
11 $d[v] \leftarrow \|cc_{\pi_v}\|$; $p[v] \leftarrow u$

To decide whether π_T is valid, for each vertex $u \in V_T$ such that u is the center of four free cells c_1, c_2, c_3, and c_4 in g, we approximate the cable configurations associated with the four paths from r to c_1, c_2, c_3, and c_4, respectively, by the cable configuration that starts from R and ends on u. The error induced by this approximation never exceeds the length of a cell diagonal.

III. Polynomial algorithm for computing MCTs when there are no Forbidden Cells

Consider the case where the workspace g does not contain forbidden cells. Depending on the total length of the cable, some free cells may be out of reach. We may perform a Breadth First Search (BFS) from R to compute a covering tree that minimizes the path length from R to u for each vertex $u \in V_4$. However, the length $\|cc_\pi\|$ of the cable configuration associated with a shortest path π in g is not necessarily minimum, as illustrated in Fig. 1(b). We show in Algo. 1 how to adapt Dijkstra's algorithm [13] to compute an MCT T. For each vertex $u \in V_4$, $p[u]$ is the predecessor of u in the best known path π_u from R to u, and $d[u] = \|cc_{\pi_u}\|$. The tree $T = (V_T, E_T)$ is incrementally built: at each iteration of the loop Lines 3-11, we add to V_T the vertex u such that $d[u]$ is minimal, and we add to E_T the edge $(p[u], u)$, except when u is the root R (lines 4-6). Then, for each neighbour v of u, we update $d[v]$ and $p[v]$ whenever we obtain a shorter cable configuration by visiting v just after u, provided that the maximal cable length is not exceeded (lines 10-11).

Correctness: The main difference with Dijkstra's algorithm comes from the fact that the cost of an edge (u, v) depends on the path π_u from R to u in T: this cost is equal to $\|cc_{\pi_v}\| - \|cc_{\pi_u}\|$. In some cases, this cost is negative (because the cable configuration length is reduced when the robot moves from u to v). The correctness of Algo. 1 relies on the fact that, for every vertex $u \in V_4$, the path $\pi = \langle v_0, \ldots, v_k \rangle$ from R to u that minimizes the length of the cable configuration is such that, $\forall i \in [1, k]$, $\|cc_{\langle v_0, \ldots, v_{i-1} \rangle}\| \leq \|cc_{\langle v_0, \ldots, v_i \rangle}\|$. In other words, whenever the robot follows π, the length of the cable configuration never decreases. This is a straightforward consequence of the fact that the cable configuration cc_π associated with a path π is the shortest path homotopic to π. This non decreasing property implies that the correctness proof of Dijkstra's algorithm can be transposed to Algo. 1 [13], even if some edges may have a negative cost *because these edges do not belong to best paths*. Hence, at the end of Algo. 1, the tree T contains all reachable vertices. Let us finally prove that this tree is an MCT by proving that, for each vertex $u \in V_T$, the cable configuration cc_π associated with the path π from R to u in T is not self-crossing. This is ensured by the fact that a self-crossing cable configuration cannot be a shortest cable configuration, given that we obtain a shorter cable configuration by removing the crossing as demonstrated in [14]. For example, in Fig. 2(b), the cable is self-crossing when the robot is on u. In this case, the cable configuration is not the shortest one as the path $\langle R, v, u \rangle$ is shorter.

Complexity: Lines 3-11 are iterated $\mathcal{O}(|V_4|)$ times, as a vertex of $V_4 \setminus V_T$ is added to V_T at each iteration, and iterations are stopped when $V_4 \setminus V_T = \emptyset$ in the worst case. Lines 8-11 are iterated at most 4 times, as each vertex in V_4 has at most 4 neighbours. A key point is to efficiently compute $\|cc_{\pi_v}\|$ (Line 10). This is done in $\mathcal{O}(|V_4|)$ by memorizing the cable configuration cc_{π_u} associated with the path from R to u, for each cell group $u \in V_T$, and by adapting the funnel algorithm described in [15] to incrementally compute cc_{π_v} from cc_{π_u}. Hence, the time complexity of Algo. 1 is $\mathcal{O}(|V_4|^2)$.

IV. NP-completeness in Case of Forbidden Cells

When g contains forbidden cells, we must ensure that the cable never crosses them, and this increases the complexity as stated in the following theorem.

Theorem IV.1. Given a cell group graph G_4 and an integer k, deciding whether there exists a covering tree $T = (V_T, E_T)$ of G_4 such that π_T is valid and $|V_T| \geq k$ is $\mathcal{NP}$-complete.

Let us first prove that this problem belongs to $\mathcal{NP}$. Indeed, the cable configuration associated with every vertex in T may be computed in polynomial time using the algorithm described in [15] and, given a cable configuration, we can check in polynomial time that it does not cross forbidden areas.

To prove $\mathcal{NP}$-hardness, we give a poly-time reduction from Separable Planar 3-SAT which is $\mathcal{NP}$-Complete [16]. An instance of Separable Planar 3-SAT is defined by a triple (X, F, μ) such that:

- $X = \{x_1, ..., x_n\}$ is a set of n boolean variables;
- $F = C_1 \wedge ... \wedge C_m$ is a conjunction of m clauses such that (i) each clause C_j is a disjunction of 2 or 3 literals, where a literal is either a variable x_i or its negation $\neg x_i$; and (ii) every variable x_i occurs in 2 or 3 clauses and at least once positively and once negatively;

- $\mu : X \cup F \rightarrow \mathbb{R}^2$ is a plane embedding that associates 2D coordinates to every variable and every clause such that (i) it is possible to draw continuous lines between every clause and its variables without line crossings, (ii) all variables are aligned, and when drawing the straight line that goes through all variables, thus separating the space in two half-spaces, for each variable x_i, all clauses that contain the literal x_i are located in the same half-space whereas all clauses that contain the literal $\neg x_i$ are located in the other half-space, as illustrated in Fig. 3 (see [17] for more details).

From a Separable Planar 3-SAT instance (X, F, μ), an instance of our problem is constructed as follows. The graph G_4 is composed of two parts: a variable part, displayed in green in Fig. 3, and a clause part, displayed in blue in Fig. 3.

The green part of G_4 contains n aligned horizontal obstacles surrounded by two horizontal lines of vertices. These obstacles are separated by couples of vertical arms. Each couple is associated with a different variable x_i, in the same order as in the plane embedding defined by μ: the upper arm (resp. lower arm) is labelled with x_i if all clauses that contain the literal x_i are above (resp. below) the variable line defined by μ, and it is labelled with $\neg x_i$ otherwise. Forbidden areas are located between these arms. The root R is on the left of the first obstacle. A tree T rooted in R cannot cover the two arms of a same variable. Indeed, each horizontal obstacle may be used to prevent the cable from crossing forbidden areas when turning on a vertical arm, but we have to choose: to cover the upper (resp. lower) arm, the robot must pass below (resp. above) the obstacle before turning left (resp. right) on the arm and it is not possible to cover both arms without creating a cycle in T as there is only one vertex between two obstacles. Hence, there are 2^n different possible MCTs of the green part of G_4, each of them corresponding to a different truth assignment of the n variables.

The blue part of G_4 contains a convex area of K vertices for each clause. Each convex area associated with a clause is connected to the green arms associated to its literals using blue corridors. Obstacles are added at intersections of blue corridors to prevent cable configurations from crossing forbidden areas. The separable plane embedding μ associated with (X, F) ensures that the convex areas are properly nested so that corridors and arms do not cross each other. This is why we use an instance of Separable Planar 3-SAT to make this reduction, rather than the classical 3-SAT. We make K greater than the total number of vertices in corridors and arms. This condition can be satisfied if we scale properly the length of the arms. Finally, we set $k = mK$, and the cable length to ∞. The construction can be done in polynomial time in the size of F.

We now prove that F is satisfiable iff G_4 has a coverage tree $T = (V_T, E_T)$ such that $|V_T| \geq k$.

$\boxed{\Rightarrow}$ Let $\{\tilde{x}_1, ..., \tilde{x}_n\}$ be a truth assignment of X satisfying F. We build the tree T such that, for each variable x_i, T covers the green arm labelled with $\tilde{x}_i$. To connect this arm to R, we add to T the green cells below (resp. above) the horizontal obstacle if the arm is above (resp. below) the obstacle. As each clause C_j is satisfied by the truth assignment, there is at least one of its literals set to true, and T covers the blue

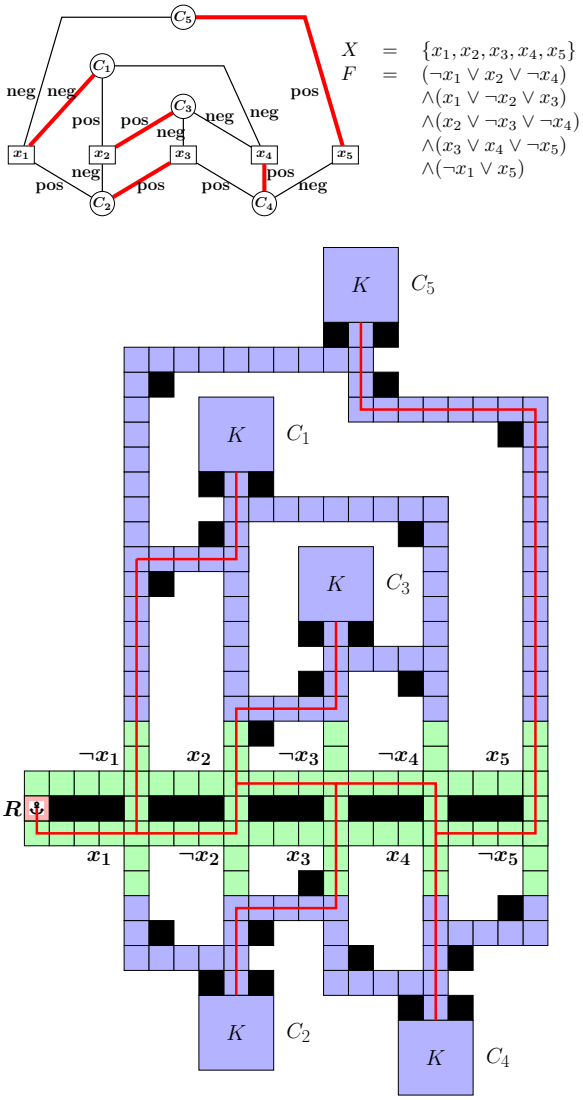


Fig. 3. Top: An instance of Separable Planar 3-SAT. The embedding μ is displayed on the left, and X and F on the right. The truth assignment $A = \{\neg x_1, x_2, x_3, x_4, x_5\}$ satisfies all clauses and we highlight in red edges that connect each clause with one satisfied literal. Bottom: Graph G_4 built from (X, F, μ). The tree in red connects the root R with the five K-cell blue blocks associated with the five clauses using the same assignment as in A. Empty areas are considered as forbidden zones.

corridor that connects this literal to the convex area of size K associated with C_j. Finally, T covers each convex area using a comb-shaped tree. Therefore, T has a size greater than k.

$\boxed{\Leftarrow}$ Let $T = (V_T, E_T)$ be a covering tree of G_4 such that $|V| \geq k$. V_T must contain all vertices in all convex areas associated with clauses since we assume that K is greater than the total size of all corridors. Each convex area is connected to T by using a green arm associated with a literal and it is not possible to have in a same covering tree an arm associated with x_i and an arm associated with $\neg x_i$, as explained above. We set each variable x_i to true (resp. false) if T contains an arm labeled with x_i (resp. $\neg x_i$). It may happen that T contains no arm associated with a variable x_i, when each clause that contains x_i is satisfied by another variable than x_i. In this case we set x_i to true by convention. The resulting assignment satisfies F. For example, in Fig. 3, there exists a tree that

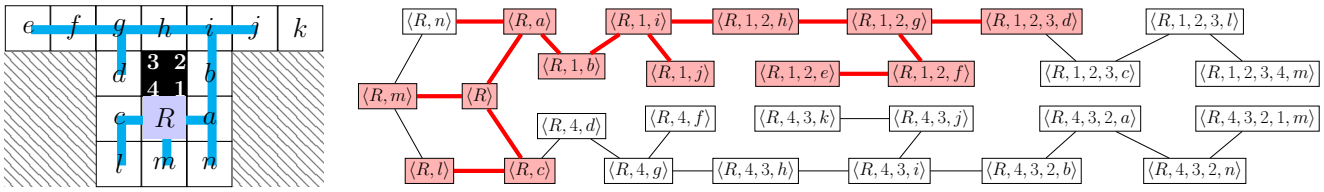


Fig. 4. Left: Example of cell group graph G_4, with an obstacle in black (surrounded by vertices 1 to 4), and two forbidden areas in grey. The robot must take path $\langle R, a, b, i, h, g, f, e\rangle$ (resp. $\langle R, c, d, g, h, i, j, k\rangle$) in order to reach cell e. (resp. k). An MCT is displayed in blue. Right: Associated configuration graph G_C. The MCT of G_C corresponding to the blue MCT of G_C is displayed in red.

covers the five clauses by choosing the arms $\neg x_1$, x_2, and x_3. In this case, we set x_4 and x_5 to true.

V. Computing MCTs using Configuration Graphs

When there are forbidden areas, the complexity of MCT comes from the fact that some cable configurations are not compatible, as illustrated in the $\mathcal{NP}$-completeness proof. To compute an MCT, we must explore all possible cable configurations to reach each cell group. Given a cable configuration $cc = \langle R, v_1, \dots, v_k, u\rangle$ from R to u, we note d_{cc} its destination, *i.e.*, $d_{cc} = u$. The relations between cable configurations can be represented as a configuration graph, similar to the grid-based homotopy-augmented graph of [3].

Definition V.1 (Configuration graph)**.** The *configuration graph* associated with $G_4 = (V_4, E_4)$ is the graph $G_C = (V_C, E_C)$ such that V_C is the set of all valid cable configurations in G_4, and $E_C = \{(cc, cc') \in V_C \times V_C \mid (d_{cc}, d_{cc'}) \in E_4$ and cc is homotopic to $cc' \circ \langle d_{cc}\rangle\}$, where $\circ$ denotes concatenation.

Each vertex $cc \in V_C$ corresponds to a valid cable configuration from R to d_{cc} in G_4. In Fig. 4, for example, there are two possible configurations to reach f: $\langle R, 1, 2, f\rangle$, and $\langle R, 4, f\rangle$. The neighbors of a configuration $cc \in V_C$ correspond to the valid moves from d_{cc} to a neighbour of d_{cc} in G_4. For example, when the configuration is $\langle R, 1, 2, f\rangle$, the robot can move to g or e (leading to configurations $\langle R, 1, 2, g\rangle$ or $\langle R, 1, 2, e\rangle$, respectively), whereas when the configuration is $\langle R, 4, f\rangle$ the robot can only move to g (leading to configuration $\langle R, 4, g\rangle$) because if the robot moves to e then its cable crosses forbidden cells.

To efficiently enumerate all valid cable configurations, we perform a Depth First Search (DFS) in the visibility graph associated with G_4 [18], and use a tree to represent the set of all consistent cable configuration prefixes as proposed in [4].

As G_C represents valid moves, we can use it to build an MCT of G_4. More precisely, given a covering tree $T = (V_T, E_T)$ of G_C such that the vertices of T correspond to cable configurations that end on different vertices of G_4 (*i.e.*, $\forall\{cc, cc'\} \subseteq V_T, d_{cc} \neq d_{cc'}$), we can build the covering tree $T' = (V_{T'}, E_{T'})$ of G_4 such that $V_{T'} = \{d_{cc} | cc \in V_T\}$ and $E_{T'} = \{(d_{cc_1}, d_{cc_2}) | (cc_1, cc_2) \in E_T\}$. The validity of $\pi_{T'}$ is ensured by Def. V.1. Hence, to solve the MCT problem, we may search for an MCT in G_C, as illustrated in Fig. 4.

The number of vertices in G_C is in $\mathcal{O}(|V_4| \cdot |CP|)$ where CP is the set of all valid cable configuration prefixes. As the size of CP grows exponentially with respect to the number of obstacles in g, and as the maximum degree of a vertex in G_C is four, the size of G_C grows linearly with respect to $|V_4|$ and exponentially with respect to the number of obstacles in g. We have designed an ILP model to search for an MCT in G_C, but it does not scale well enough. We show in the next two sections how to compute approximate solutions more efficiently by reducing the size of G_C.

VI. Approximating MCTs using Directed Configuration Graphs

The configuration graph G_C is undirected and usually contains cycles. When designing an ILP model to search for an MCT in G_C, we initially tested flow-based constraints to handle subtour elimination [19]. However, its performance is heavily constrained by the graph size, limiting its effectiveness to very small instances.[1] To improve scale-up properties, we propose to orientate the edges of G_C to forbid moves that decrease the cable configuration length, thus obtaining a Directed Acyclic Graph (DAG) in which subtour elimination constraints become useless. More precisely, given a configuration graph G_C, we define the DAG $\hat{G}_C = (\hat{V}_C, \hat{E}_C)$ such that $\hat{V}_C = V_C$, $\hat{E}_C = \{(cc, cc') \in E_C, \|cc\| < \|cc'\|\}$. $\hat{G}_C$ cannot contain cycles as it is not possible to have a cycle of decreasing lengths.

When there is no forbidden area, we can remove edges corresponding to decreasing cable configuration lengths without changing the optimal solution, because a path that minimizes the cable configuration length never uses these decreasing edges. However, this is no longer the case when there are forbidden areas: an MCT of $\hat{G}_C$ is a covering tree of G_C, but it may not be maximum, as shown in Fig. 5.

The ILP model displayed in Fig. 6, denoted $ILP(\hat{G}_C)$, computes an MCT of $\hat{G}_C$. For each cable configuration $cc \in \hat{V}_C$, the binary variable z_{cc} is set to 1 if cc is selected in the MCT. For each edge $(cc, cc') \in \hat{E}_C$, the binary variable $x_{cc,cc'}$ is set to 1 if (cc, cc') is selected in the MCT. Eq. (1) ensures that we maximize the number of selected cable configurations. Eq. (2) ensures that at most one selected cable configurations reaches vertex v, $\forall v \in V_4$. Eq. (3) ensures that $\langle R\rangle$ is selected. Eq. (4) ensures that each selected cable configuration (except $\langle R\rangle$) has exactly one predecessor in the MCT. Eq. (5) ensures that the number of selected edges is equal to the number of selected cable configurations minus one. Eq. (6) ensures that

[1]For more implementation details, see the thesis: https://hal.science/tel-04553494

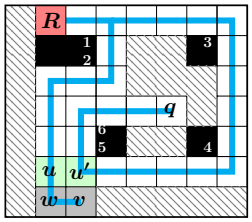


Fig. 5. Example of graph G_4 such that the optimal solution computed in G_C (displayed in cyan) covers all vertices. Vertex q can only be reached with configuration $\langle R, 3, 4, 5, 6, q\rangle$, making u' already covered, so that the only possibility to reach v is to come from w. However, $\|\langle R, 1, 2, w\rangle\| > \|\langle R, 1, 2, v\rangle\|$ so that $(\langle R, 1, 2, w\rangle, \langle R, 1, 2, v\rangle) \notin \hat{E}_C$. Therefore, the MCT in $\hat{G}_{cc}$ cannot cover all vertices of G_4.

$$\max \sum_{cc \in \hat{V}_C} z_{cc} \quad (1)$$

$$\text{s.t.} \sum_{cc \in \hat{V}_C, d_{cc}=v} z_{cc} \leq 1, \forall v \in V_4 \quad (2)$$

$$z_{\langle R\rangle} = 1 \quad (3)$$

$$\sum_{cc' \in \hat{V}_C} x_{cc',cc} = z_{cc}, \forall cc \in V_C \setminus \{\langle R\rangle\} \quad (4)$$

$$\sum_{(cc,cc') \in \hat{E}_C} x_{cc,cc'} = (\sum_{cc \in \hat{V}_C} z_{cc}) - 1 \quad (5)$$

$$x_{cc,cc'} \leq z_{cc} z_{cc'}, \forall (cc, cc') \in \hat{E}_C \quad (6)$$

$$z_{cc} \in \{0, 1\}, \forall cc \in \hat{V}_C \quad (7)$$

$$x_{cc,cc'} \in \{0, 1\}, \forall (cc, cc') \in \hat{E}_C \quad (8)$$

Fig. 6. ILP model $ILP(\hat{G}_C)$ for computing an MCT of $\hat{G}_C$.

an edge is selected only if its two endpoints are selected, and it can be implemented with linear constraints by introducing the auxiliary variables $w_{cc,cc'}$ associated with each edge, such that: (i) $x_{cc,cc'} \leq w_{cc,cc'}$, (ii) $w_{cc,cc'} \leq z_{cc}$, (iii) $w_{cc,cc'} \leq z_{cc'}$, (iv) $z_{cc} + z_{cc'} - 1 \leq w_{cc,cc'}$.

VII. Approximating MCTs using Quotient Graphs

The size of G_C grows linearly with respect to $|V_4|$ and exponentially with respect to the number of obstacles in g. However, $|V_4|$ is often large whereas the number of obstacles may be small. In this case, the size of G_C is mostly impacted by $|V_4|$, and we may reduce the size of G_C by merging similar cable configurations. For example, let us consider the graph G_4 displayed in Fig. 4, and let us imagine that we add a large set V' of aligned vertices on the left of e. For each vertex $i \in V'$, there is only one possible configuration that ends on i, *i.e.*, $\langle R, 1, 2, i\rangle$, and we propose to merge all these configurations to treat all vertices in $V' \cup \{e\}$ at once. This is done by introducing an equivalence relation between cable configurations, based on configuration prefixes. Given a configuration $cc = \langle R, v_1, \ldots, v_k, u\rangle$, we note p_{cc} its prefix, *i.e.*, $p_{cc} = \langle R, v_1, \ldots, v_k\rangle$. Given a vertex $u \in V_4$, we note $\mathcal{P}(u)$ the set of prefixes of configurations that reach u, *i.e.*, $\mathcal{P}(u) = \{p_{cc} \mid cc \in V_C \text{ and } d_{cc} = u\}$.

Definition VII.1 (Equivalence relation on V_C)**.** Two configurations $cc, cc' \in V_C$ are *equivalent*, denoted $cc \sim cc'$ if $p_{cc} = p_{cc'}$ and $\mathcal{P}(d_{cc}) = \mathcal{P}(d_{cc'})$.

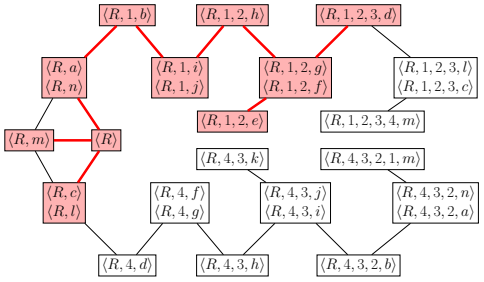


Fig. 7. Quotient graph G_q associated with the configuration graph G_C displayed in Fig. 4. A covering tree is displayed in red.

In Fig. 4, for example, $\langle R, 1, 2, f\rangle \sim \langle R, 1, 2, g\rangle$ because both configurations share the same prefix $\langle R, 1, 2\rangle$, and $\mathcal{P}(f) = \mathcal{P}(g) = \{\langle R, 4\rangle, \langle R, 1, 2\rangle\}$.

The equivalence class of a configuration $cc \in V_C$, noted $[cc]$, is the set of all configurations equivalent to cc, *i.e.*, $[cc] = \{cc' \in V_C \mid cc \sim cc'\}$, and the set of all equivalence classes defines a partition of V_C. We propose to reduce the configuration graph G_C by considering its quotient graph $G_q = (V_q, E_q)$ with respect to $\sim$: the vertices of G_q are the equivalence classes, *i.e.*, $V_q = \{[cc] \mid cc \in V_C\}$, and there is an edge between two equivalence classes $[cc]$ and $[cc']$ whenever there exists an edge between a configuration in $[cc]$ and a configuration in $[cc']$, *i.e.*, $E_q = \{([cc], [cc']) \mid (cc, cc') \in E_C\}$.

For example, we display in Fig. 7 the quotient graph G_q associated with the configuration graph G_C displayed in Fig. 4. G_q has 20 vertices, whereas G_C has 28 vertices. Note that if we subdivide each vertex of G_4 in k^2 vertices (and scale the obstacle and forbidden areas consequently), the number of vertices in G_q does not increase, whereas the number of vertices in G_C is increased by a factor k^2.

Note that for every pair of equivalence classes $[cc], [cc'] \in V_q$, either $[cc]$ and $[cc']$ have the same set of destinations, or their sets of destinations are disjoint (*i.e.*, either $\{d_{cc_i} \mid cc_i \in [cc]\} = \{d_{cc'_i} \mid cc'_i \in [cc']\}$ or $\{d_{cc_i} \mid cc_i \in [cc]\} \cap \{d_{cc'_i} \mid cc'_i \in [cc']\} = \emptyset$). This comes from the fact that Def. VII.1 ensures that all the destinations of the configurations in a same equivalence class share the same set of configuration prefixes.

Given a covering tree $T = (V_T, E_T)$ of G_q such that classes in V_T have disjoint sets of destinations (*i.e.*, $\forall \{[cc], [cc']\} \subseteq V_T, \{d_{cc_i} | cc_i \in [cc]\} \cap \{d_{cc'_i} | cc'_i \in [cc']\} = \emptyset$), we can build a covering tree $T' = (V_{T'}, E_{T'})$ of G_C. $V_{T'}$ is the set of all destinations in all equivalence classes in V_T, *i.e.*, $V_{T'} = \cup_{[cc] \in V_T} \{d_{cc_i} | cc_i \in [cc]\}$. The subgraph of G_C induced by $V_{T'}$ is connected because, for each equivalence class $[cc]$, the subgraph of G_C induced by $[cc]$ is connected. Hence, we can easily select $|V_{T'}| - 1$ edges, among the set of edges of the subgraph of G_C induced by $V_{T'}$ so that the tree T' is connected.

For example, the tree highlighted in the quotient graph G_q displayed in Fig. 7 corresponds to the tree highlighted in the configuration graph G_C displayed in Fig. 4.

The covering tree T' of G_c may not be maximal. To increase the number of vertices of V_4 that are covered by a cable configuration in $V_{T'}$, we may search for the tree T that maximizes the sum of the number of configurations in each

equivalence class in V_T, *i.e.*, $\sum_{[cc]\in V_T} |[cc]|$. However, the tree T that is maximal with respect to this objective function may not correspond to an MCT of G_C. This comes from the fact that we deliberately choose the same cable configuration prefix for covering all destinations within a same equivalence class whereas, in some cases, the MCT of G_C uses different prefixes. For example, in the MCT displayed in Fig. 5, u and u' share the same two cable configuration prefixes, *i.e.*, $\langle R, 3, 4\rangle$ and $\langle R, 1, 2\rangle$ and there are two equivalence classes that contain these vertices, *i.e.*, $[\langle R, 3, 4, u\rangle]$ and $[\langle R, 1, 2, u\rangle]$. If $[\langle R, 3, 4, u\rangle]$ is selected in the covering tree of G_q, then both u and u' are reached from $\langle R, 3, 4\rangle$ and, therefore, v cannot be covered. If $[\langle R, 1, 2, u\rangle]$ is selected in the covering tree of G_q, then both u and u' are reached from $\langle R, 1, 2\rangle$ and, therefore, q cannot be covered.

Finally, we may further relax the problem by converting G_q into a DAG, denoted $\hat{G}_q$, based on the shortest length of configurations contained in each equivalence class of V_q, similar to what has been done with G_C. A lower bound can be computed by searching for a maximum arborescence $\hat{T}_q$ in $\hat{G}_q$. We propose to apply the same ILP model as that of $ILP(\hat{G}_C)$ to solve this problem. The only difference is that in this case, the objective function becomes: $\max \sum_{i\in V_q} c_i z_i$ where c_i is the number of cells in the i^{th} equivalence class of V_q. This approach is referred to as $ILP(\hat{G}_q)$. Note that both of the two approximated solutions via ILP are guaranteed to be feasible, meaning that all the constraints are satisfied.

VIII. Experimental Results

A. Description of Benchmarks

To analyze the factors affecting the complexity of the problem, we generated 6 workspaces by incorporating randomly positioned rectangular obstacles and forbidden areas, as depicted in Fig. 8. The bounding polygon for workspace (a) is defined as the rectangle $[0, 25] \times [0, 20]$, and the other 5 workspaces are bounded by $[0, 70] \times [0, 60]$. Another variable in our problem is the location of the anchor point, and by choosing different anchor points in each workspace, we generate a total of 10 scenarios, as described in Tab. I. For each scenario, we fix a limit ℓ_{max} on the maximum cable length value to ensure its solvability. We generated 107 instances by incrementing the cable length ℓ from 10 to ℓ_{max} for each scenario. Our ILP models $ILP(\hat{G}_C)$ and $ILP(\hat{G}_q)$ are both solved using the Gurobi Solver (version 9.1.2) [20], and the run time is limited to 3600 seconds. All experiments reported here are implemented in Python[2] and run on a computer with an Intel Core Intel Xeon E5-2623v3 processor operating at 3.0 GHz with 16 cores and 32GB of RAM.

B. Computational Performance

We denote ub as an upper bound of TCPP, which counts all free cells that can be reached by a certain cable configuration under the given cable length, hence it can be larger than the optimal solution. Each method $x \in \{ILP(\hat{G}_C), ILP(\hat{G}_q)\}$ computes a lower bound lb_x and these lower bounds are not necessarily equal. To evaluate the quality of these bounds, we calculate the bound gap $g_x = \frac{ub - lb_x}{ub}$ for each method. The results reveal that for $ILP(\hat{G}_C)$ and $ILP(\hat{G}_q)$, 40.2% and 39.3% of the instances, respectively, have a bound gap of 0, and 73.8% and 72.9% of instances below 0.05. The maximum observed g_x is 0.197. Fig. 9 presents a comparison of the solving times for the two models. Since each model computes different bounds, we evaluate them based on the bound gap g_x and the instance hardness, as measured by the cable length. This comparison aligns with the theoretical complexity analysis, indicating that $ILP(\hat{G}_q)$ proves to be more efficient when $|\mathcal{V}_4|$ is much larger than the number of obstacles.

[2]Our implementation is publicly available at https://gitlab.inria.fr/xipeng/tcpp.git.

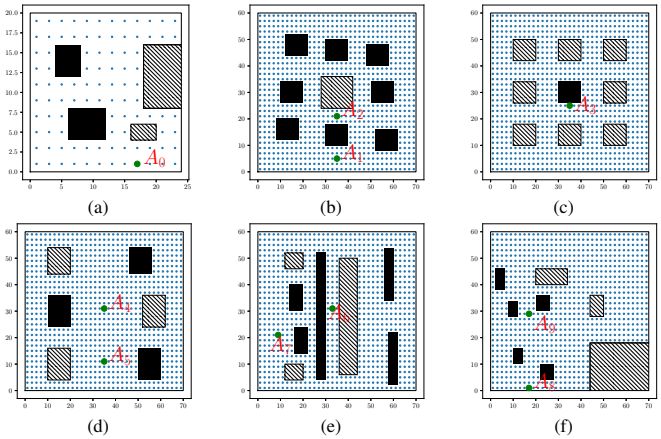


Fig. 8. The 6 simulated workspaces tested in our experimental analysis. In each workspace, the black rectangles are the obstacles, the forbidden areas are filled with slash lines, and the green points are the anchor points. Workspaces are discretized, and the blue points depict the vertices in G_4.

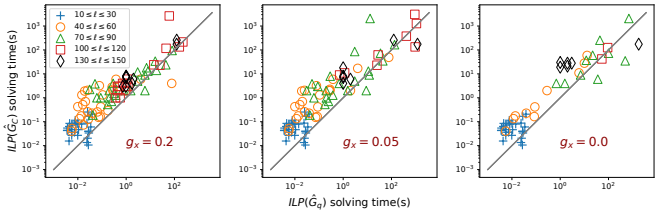


Fig. 9. Comparison of the two ILP model's solving times with regard to solution quality and instance hardness. Each point represents an instance, with solving time of $ILP(\hat{G}_q)$ on the x-axis and $ILP(\hat{G}_C)$ on the y-axis.

TABLE I

Experimental scenarios: Each line successively gives the scenario name S, the workspace g, the anchor position R, the number N_{free} of free cells in g, and the maximum cable length ℓ_{max}. For each ℓ_{max}, we give the number N_{max} of cells that can be reached and the coverage ratio $\frac{N_{max}}{N_{free}}$.

S	g	R	N_{free}	ℓ_{max}	N_{max}	$\frac{N_{max}}{N_{free}}$
I_0	(a)	A_0	96	50	96	1.0
I_1	(b)	A_1	848	100	848	1.0
I_2	(b)	A_2	848	100	848	1.0
I_3	(c)	A_3	870	50	492	0.566
I_4	(d)	A_4	880	100	880	1.0
I_5	(d)	A_5	880	140	880	1.0
I_6	(e)	A_6	820	140	705	0.86
I_7	(e)	A_7	820	150	820	1.0
I_8	(f)	A_8	862	140	856	0.993
I_9	(f)	A_9	862	100	856	0.993

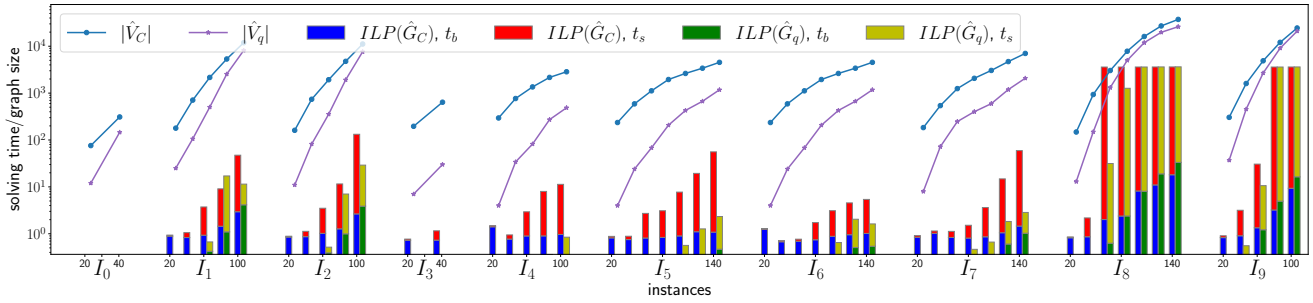


Fig. 10. Evolution of graph sizes ($|\hat{V}_C|$ and $|\hat{V}_q|$) and solving times (y-axis with a log scale) with respect to the cable length (x-axis, incremented by 20) for each instance. t_s represents ILP solving time and t_b is for building time. These bars of small value are not shown.

In Fig. 10, we present the evolution of total solving times, composed of ILP model building times (t_b) and ILP solving times (t_s). Specifically, building times include enumerating all consistent cable configurations and constructing corresponding graphs ($\hat{G}_C$ and $\hat{G}_q$). Graph sizes ($|\hat{V}_C|$ and $|\hat{V}_q|$) are displayed to understand their impact on solving times. Larger workspaces generally require more time to solve, especially when ℓ is sufficiently large. We observe that both $|\hat{V}_C|$ and $|\hat{V}_q|$ grow exponentially with cable length. In each method, t_s correlates strongly with the graph size. However, the performance of $ILP(\hat{G}_q)$ is degraded as ℓ increases. Consider instance I_8 and I_9, when ℓ reaches 80, the gap between $|\hat{V}_C|$ and $|\hat{V}_q|$ shrinks, and so does t_s, reaching the 1-hour time limit. In fact, as the number of homotopy classes expands with the cable length, the weight of each vertex of $\hat{G}_q$, which is equivalent to the size of the corresponding partition of $\hat{G}_C$ decreases. In other words, the partitions of G_4 become finer and $\hat{G}_q$ is closer to $\hat{G}_C$. Generally, t_b is relatively smaller compared to t_s. It increases with the cable length ℓ when ℓ is small. However, in the presence of more obstacles in the workspace, t_b increases exponentially, as can be observed in instance I_8 and I_9. When ℓ is small, it takes less time to construct $\hat{G}_q$ than $\hat{G}_C$ because $|\hat{V}_C| > |\hat{V}_q|$. As ℓ becomes larger, $|\hat{V}_q|$ gets closer to $|\hat{V}_C|$, and connecting two vertices in $\hat{G}_q$ is more computationally expensive than connecting two simple vertices in $\hat{G}_C$. This is because we have to find the boundary cells and check their connectivity in $\hat{G}_q$. It explains why t_b in $ILP(\hat{G}_q)$ exceeds that for $ILP(\hat{G}_C)$ when ℓ increases.

While the presence of forbidden zones renders the problem $\mathcal{NP}$-complete, hardness depends on the number of obstacles rather than the number of forbidden zones (comparing I_1, I_2 and I_3). More obstacles, coupled with longer cable lengths, allow to explore more homotopy classes, thereby enlarging the graph size and increasing solving time. Another factor that could have an impact on the hardness of the problem is the anchor position, as we can compare the result of I_6 with I_7, and I_8 with I_9. Intensely distributed obstacles surrounding an anchor point near their center can generate a multitude of homotopy classes, further increasing problem complexity. In this study, we provide a qualitative analysis of these results, as we believe that instance hardness stems from a combination of various topological factors, and hence difficult to discuss each factor separately. Our aim is to show in which cases a TCPP instance can be hard, and how to calculate a satisfactory approximate solution for it.

REFERENCES

[1] H. Choset, "Coverage for robotics–a survey of recent results," *Annals of mathematics and artificial intelligence*, vol. 31, pp. 113–126, 2001.

[2] T. Igarashi and M. Stilman, "Homotopic path planning on manifolds for cabled mobile robots," in *Algorithmic Foundations of Robotics IX: Selected Contributions of the Ninth International Workshop on the Algorithmic Foundations of Robotics*. Springer, 2010, pp. 1–18.

[3] S. Kim, S. Bhattacharya, and V. Kumar, "Path planning for a tethered mobile robot," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 1132–1139.

[4] O. Salzman and D. Halperin, "Optimal motion planning for a tethered robot: Efficient preprocessing for fast shortest paths queries," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 4161–4166.

[5] A. C. Kapoutsis, S. A. Chatzichristofis, and E. B. Kosmatopoulos, "Darp: Divide areas algorithm for optimal multi-robot coverage path planning," *Journal of Intelligent & Robotic Systems*, vol. 86, pp. 663–680, 2017.

[6] Y. Gabriely and E. Rimon, "Spanning-tree based coverage of continuous areas by a mobile robot," *Annals of mathematics and artificial intelligence*, vol. 31, pp. 77–98, 2001.

[7] A. Itai, C. H. Papadimitriou, and J. L. Szwarcfiter, "Hamilton paths in grid graphs," *SIAM Journal on Computing*, 1982.

[8] I. Shnaps and E. Rimon, "Online coverage by a tethered autonomous mobile robot in planar unknown environments," *IEEE Transactions on Robotics*, vol. 30, no. 4, pp. 966–974, 2014.

[9] L. Mechsy, M. Dias, W. Pragithmukar, and A. Kulasekera, "A novel offline coverage path planning algorithm for a tethered robot," in *2017 17th International Conference on Control, Automation and Systems (ICCAS)*. IEEE, 2017, pp. 218–223.

[10] E. U. Acar and H. Choset, "Sensor-based coverage of unknown environments: Incremental construction of morse decompositions," *The International Journal of Robotics Research*, 2002.

[11] G. Sharma, P. Poudel, A. Dutta, V. Zeinali, T. T. Khoei, and J.-H. Kim, "A 2-approximation algorithm for the online tethered coverage problem." in *Robotics: Science and Systems*, 2019.

[12] S. Bhattacharya, "Search-based path planning with homotopy class constraints," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 24, no. 1, 2010, pp. 1230–1237.

[13] Cormen, Thomas H., Leiserson, Charles E., Rivest, Ronald L., and Stein, Clifford, *Introduction to Algorithms*, 2009.

[14] X. Peng, O. Simonin, and C. Solnon, "Non-crossing anonymous mapf for tethered robots," *Journal of Artificial Intelligence Research (JAIR)*, vol. 78, 2023.

[15] P. Brass, I. Vigan, and N. Xu, "Shortest path planning for a tethered robot," *Computational Geometry*, vol. 48, no. 9, pp. 732–742, 2015.

[16] D. Lichtenstein, "Planar formulae and their uses," *SIAM journal on computing*, vol. 11, no. 2, pp. 329–343, 1982.

[17] C. Solnon, G. Damiand, C. De La Higuera, and J.-C. Janodet, "On the complexity of submap isomorphism and maximum common submap problems," *Pattern Recognition*, vol. 48, no. 2, pp. 302–316, 2015.

[18] M. De Berg, *Computational geometry: algorithms and applications*. Springer Science & Business Media, 2000.

[19] P. C. Pop, "The generalized minimum spanning tree problem: An overview of formulations, solution procedures and latest advances," *European Journal of Operational Research*, vol. 283, no. 1, 2020.

[20] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2023. [Online]. Available: https://www.gurobi.com