Heterogeneous Multi-Robot Graph Coverage with Proximity and Movement Constraints

Dolev Mutzari, Yonatan Aumann, Sarit Kraus

Department of Computer Science, Bar Ilan University, Ramat Gan, Israel
dolevmu@gmail.com, aumann@cs.biu.ac.il, sarit@cs.biu.ac.il

Abstract

Multi-Robot Coverage problems have been extensively studied in robotics, planning and multi-agent systems. In this work, we consider the coverage problem when there are constraints on the proximity (e.g., maximum distance between the agents, or a blue agent must be adjacent to a red agent) and the movement (e.g., terrain traversability and material load capacity) of the robots. Such constraints naturally arise in many real-world applications, e.g. in search-and-rescue and maintenance operations. Given such a setting, the goal is to compute a covering tour of the graph with a minimum number of steps, and that adheres to the proximity and movement constraints. For this problem, our contributions are four: (i) a formal formulation of the problem, (ii) an exact algorithm that is FPT in parameters $\|\mathcal{F}\|$, d and ω - the set of robot formations that encode the proximity constraints, the maximum nodes degree, and the tree-width of the graph, respectively, (iii) for the case that the graph is a tree: a PTAS approximation scheme, that given an ε produces a tour that is within a $1+\varepsilon\cdot\mathsf{error}(\|\mathcal{F}\|, d))$ of the optimal one, and the computation runs in time $\mathsf{poly}(n)\cdot h(\frac{1}{\varepsilon}, \|\mathcal{F}\|)$. (iv) for the case that the graph is a tree, with $k=3$ robots, and the constraint is that all agents are connected: a PTAS scheme with multiplicative approximation error of $1+\mathcal{O}(\varepsilon)$, independent of d.

Extended version — https://arxiv.org/abs/2412.10083

1 Introduction

Multi-robot graph-coverage (MRGC) models a multitude of real-world robotic scenarios, including surveillance (Scherer and Rinner 2020; Zhang et al. 2020; Vallejo et al. 2020; Lee et al. 2021; Gans and Rogers 2021; Lee 2023), cleaning applications (Nemoto and Mohan 2020; Miao, Lee, and Kang 2020), environmental monitoring (Wang et al. 2023), search and rescue operations (Queralta et al. 2020; Rodríguez et al. 2020; Yang and Parasuraman 2020; Drew 2021), warehouse automation (Salzman and Stern 2020; Bolu and Korçak 2021; Zaccaria et al. 2021; Zhang et al. 2024), and agricultural field management (Govindaraju et al. 2023; Choton and Prabhakar 2023; Mukhamediev et al. 2023), where robots are tasked with covering a defined graph-like structure efficiently and effectively (Galceran and Carreras 2013). In the *cleaning* scenario, for example, a set of *cleaning robots* is tasked with cleaning some environment, such as an office building, visiting (and cleaning) every room in the building. If every robot is independent, this is a simple instance of a multi-robot graph coverage. In many cases, however, robots act as a *team* (Nemoto and Mohan 2020), possibly with different specializations. For example, some robots may be capable of the actual *cleaning*, while other robots may be *carrier robots* - engineered to carry large loads of water or rubbish. In this case, each site/room must be visited by a cleaning robot, but the carrier robots must always be in close proximity. Similarly, in search and rescue operations, some robots may have searching capabilities, while other *rescue robots* (e.g., diggers, medical, etc.) must tag along and be available in close proximity. Even if all agents are of the same type, communication requirements may constrain them to stay within small proximity of each other. Additionally, graph edges (e.g. doors in a building) may display different physical properties (e.g., width), constraining the passage of the different robots along different edges (e.g., carrier robots cannot traverse some doors, while cleaning can). In all, graph coverage in such team settings may impose additional constraints on the maximum distance between agents, the permissible formations, and transitions.

In this paper, we consider this *team* multi-robot graph coverage problem. Specifically, we address minimizing the number of steps to cover an input graph with a team of heterogeneous robots (/agents), given constraints on (i) the permissible formations of the agents, (ii) the permissible transitions between agent formations, and (iii) the passage of different agents along graph edges. To the best of our knowledge, no previous work has addressed this setting. Notably, even if the graph is a tree and there are no constraints on agent formations, minimizing the number of steps for coverage is NP-hard (Fraigniaud et al. 2006).

Our Contributions. Our contributions are four. We provide (i) a formal formulation of the problem, (ii) an exact algorithm that is FPT in parameters d, tw, and $||\mathcal{F}||$, respectively: the maximum node degree in the graph, the tree-width of the graph, and the size of the representation of the constraints, (iii) for the case that the graph is a tree: a PTAS approximation scheme, that given an ε produces a tour that is within a $1+\varepsilon\cdot\mathsf{error}(||\mathcal{F}||, d))$ of the optimal one (where

$\mathsf{error}(\cdot)$ is independent of the graph size), and the computation runs in time $\mathsf{poly}(n) \cdot h(\frac{1}{\varepsilon}, k)$. (iv) for the case that the graph is a tree, the only constraint is that all agents are connected, and there are three agents: a PTAS scheme with approximation error of $1 + \mathcal{O}(\varepsilon)$, independent of d.

2 Related Work

We identify a variety of works that are related to multi-robot coverage problems with constraints. Nevertheless, we did not find previous work on the exact problem of interest, that is, multi-robot coverage of graphs (with bounded treewidth) under proximity or connectivity constraints.

In (Fraigniaud et al. 2006), it is shown that the *Multi-Robot Connected Tree Coverage (MRCTC)* is NP-hard. However, the parameterized complexity is not analysed. A follow-up work by (Cabrera-Mora and Xiao 2012) also considered MRCTC and, in addition, restricted the number of robots allowed to traverse an edge and enter a vertex during each step. Nevertheless, they also did not consider proximity constraints. Instead, coordination is achieved by dropping landmarks at explored vertices, enabling decentralized exploration. However, this approach has drawbacks: (i) landmarks incur costs; (ii) in rescue scenarios, they may be unavailable in time or quantity; (iii) placing them can take additional time. Later, (Sinay et al. 2017) proposed an algorithm for MRCTC and focused on connectivity constraints. Since the problem is NP-hard, they focused on the *speedup factor*, that is, the ratio between the multi-robot and the single-robot traversal time. However, no comparison with the optimal solution is provided, and only trees are considered. In (Charrier et al. 2020), the complexity of multi-agent path finding (MAPF) and multi-robot coverage is analysed for topological graphs $G = (V, E_m, E_c)$ with *movement edges* and *communication edges*, where robots must stay connected to a base station. The main results are negative, suggesting both problems are PSPACE-complete.

A well-studied use case for multi-robot coverage is mapping and model reconstruction (see (Almadhoun et al. 2019) for a recent survey). In these settings, the environment is unknown in advance. (Brass et al. 2011) considers Multi-Robot *Unknown Graph* Coverage, and focuses on exploration. In (Banfi et al. 2016) the robots must connect to a base station only when information is collected, allowing robots to disconnect for arbitrarily long periods. We view this line of work as complementary to ours. First, robots can explore and map an environment, but from that point on, we may assume the graph is given as input, e.g. for patrolling.

Another closely related area to our work is Multi-Robot Coverage Path Planning (mCPP), which involves using multiple robots to scan a continuous planar environment. Studies like (Tang, Sun, and Zhang 2021) and (Lu et al. 2023) explore mCPP under physical constraints, similar to our work, but they do not address proximity constraints. (Jensen and Gini 2018) compares mCPPs with varying communication levels. (Mechsy et al. 2017) considers a tethered robot CPP problem, where the robot has a chain structure with a constrained length. Additionally, Multi-Agent Path Finding (MAPF) (Erdem et al. 2013) focuses on planning non-colliding paths for multiple robots, and (Dutta, Ghosh, and Kreidl 2019) examines informative path planning with continuous connectivity constraints.

Proximity and connectivity are vital when considering robot *swarms*. Both (Panerati et al. 2018) and (Siligardi et al. 2019) study the problem of maintaining swarm connectivity while performing a coverage of an area of interest. In (Liu et al. 2023), land is scanned by UAV swarms with limited perception, while (Tran et al. 2023) extends this to repeated coverage with heterogeneous robots for dynamic environments. While swarm robotics is designed to scale with the number of robots, the model is somewhat limited. Indeed, robots cannot fully coordinate; they must follow relatively simple rules based on local observation and local communication, and decisions are made in real-time, individually, and asynchronously. A centralized planner, despite its limitations, can provide more efficient coverage.

3 Multi-Robot Connected Graph Coverage

Consider an undirected graph $G = (V, E)$, parameterized by its treewidth tw and maximal degree d. An edge $e_{ij} = \{v_i, v_j\}$ exists if a robot can move directly from v_i to v_j.

A *configuration* of robots $\mathbf{x} : V \to \mathbb{N}$ specifies how many robots occupy each vertex. In a *connected* configuration, the set of *occupied* vertices $\mathsf{Occupied}(\mathbf{x}) := \{v \in V : x(v) > 0\}$ forms a connected sub-graph of G. A simple interpretation is line-of-sight. A *transition* is a pair of connected configurations $(\mathbf{x}, \mathbf{x}')$, where $\mathbf{x}'$ can be reached from $\mathbf{x}$ by moving each robot along an edge. A *t-traversal* of T is a sequence of $t + 1$ connected configurations $\mathcal{X} = (\mathbf{x}^0, \dots, \mathbf{x}^t)$, that form t sequential transitions, where each vertex $v \in V$ is visited at least once by at least one robot.

The *Multi-Robot Connected Graph Coverage (MRCGC)* Problem is defined as follows: Given a graph G, a number of robots $k \in \mathbb{N}$ initially located at an entry point $s \in V$, find a *traversal* $\mathcal{X}$ of minimal time $\mathsf{time}(\mathcal{X}) := |\mathcal{X}| = t_{\mathsf{optimal}}$ that starts and terminates with all robots at s.[1]

4 Multi-Robot Formation Graph Coverage

In this work, we study an extension of MRCGC that considers a finite set M of robot *types*. In the heterogeneous setting, for each $m \in M$, there are k_m robots, and $k = \sum_m k_m$ robots overall. A configuration is now given as $\mathbf{x} : V \times M \to \mathbb{N}$, stating for each vertex $v \in V$, and for each robot type $m \in M$, how many robots of type m occupy v.

The set of valid configurations $\mathcal{C}$ is extended as well by considering *formations*. A *formation* of robots is a pair $\alpha = \langle G_\alpha, \mathbf{x}_\alpha \rangle$ where $G_\alpha = (V_\alpha, E_\alpha)$ is a connected, undirected graph, and $\mathbf{x}_\alpha : V_\alpha \times M \to \mathbb{N}$ is a configuration of k robots on G_α. A formation represents a valid way of positioning the robots. We say that $\mathbf{x}$ *is in α-form* if there exists a *graph monomorphism* $\phi : V_\alpha \to V$ such that $\mathbf{x}(\phi(v), m) = \mathbf{x}_\alpha(v, m)$ for each $v \in V_\alpha$ and each $m \in M$. We call $\mathsf{Active}(\mathbf{x}) := \phi(V_\alpha)$ the set of *active* vertices in configuration $\mathbf{x}$. Note that $\mathsf{Occupied}(\mathbf{x}) \subseteq \mathsf{Active}(\mathbf{x})$. However, in general unoccupied vertices may be active as well.

[1] We identify traversal time with traversal length. In practice, long edges can be split into shorter edges by inserting vertices.

The set of valid configurations $\mathcal{C}$ is then dictated by the set of formations $\mathcal{F} = \{\langle G_\alpha, \mathbf{x}_\alpha \rangle\}_\alpha$. We denote by $\|\mathcal{F}\|$ the representation length of the set $\mathcal{F}$, where graphs are represented with adjacency lists and maps are stored as $V_\alpha \times M$ tables.

Lastly, transitions are restricted by considering *transpositions*. A transposition is a pair $\langle G_{\{\alpha,\alpha'\}}, \{\mathbf{x}_\alpha, \mathbf{x}_{\alpha'}\}\rangle$, where $\mathbf{x}_\alpha, \mathbf{x}_{\alpha'}$ are configurations in $G_{\{\alpha,\alpha'\}}$ of α, α' form respectively, where $\mathbf{x}_{\alpha'}$ can be reached from $\mathbf{x}_\alpha$ by moving each robot along up to one edge in $G_{\{\alpha,\alpha'\}}$. A transposition represents one possible valid way of moving the robots. We say that transition $(\mathbf{x}, \mathbf{x}')$ is in (α, α') form if $\mathbf{x}, \mathbf{x}'$ are in α, α' form respectively. The set of valid transitions is determined by the set of transpositions $\mathcal{L}$. We refer to (Mutzari, Aumann, and Kraus 2024) Appendix A for illustrative examples.

The *Multi-Robot Formation Graph Coverage (MRFGC)* Problem is defined as follows: Given a Graph G, a set $\mathcal{F}$ of formations, a set $\mathcal{L}$ of transpositions, a start configuration $\mathbf{x}_0$ and an end configuration $\mathbf{x}_f$, find a *traversal* $\mathcal{X}$ of minimal time that starts at $\mathbf{x}_0$ and ends at $\mathbf{x}_f$.

5 Z-Lemma: Transitions do not Repeat

In this section, we prove that in an optimal traversal a transition cannot repeat. The proof technique is the cornerstone that enables all follow-up results presented in this work.

Lemma 1. *If $\mathcal{X}$ is optimal, then no transition repeats.*

Proof. Assume in contradiction that some transition repeats. That is, there exist $i < i'$ such that $\mathbf{x}^i = \mathbf{x}^{i'}$ and $\mathbf{x}^{i+1} = \mathbf{x}^{i'+1}$. Then we can construct a shorter traversal, denoted (of course) $\mathcal{Z}$ in contradiction, as depicted in Figure 1.

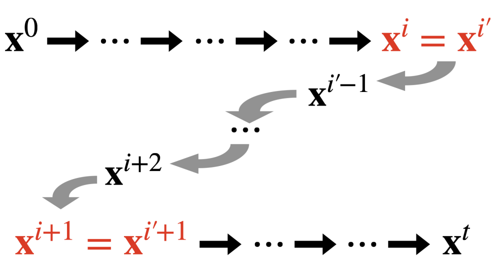

Figure 1: Traversal $\mathcal{Z}$. The repeated configurations are highlighted in red, flipped transitions are colored in gray.

Traversal $\mathcal{Z}$ is formally defined as follows. It follows $\mathcal{X}$ from initial configuration $\mathbf{x}^0$ up until $\mathbf{x}^i$. By assumption, this is the same configuration as $\mathbf{x}^{i'}$. From there, traversal $\mathcal{Z}$ follows $\mathcal{X}$ *in reverse*, to configuration $\mathbf{x}^{i'-1}$, and follows this sequence until reaching configuration $\mathbf{x}^{i+1}$. Here we critically rely on the transpositions being undirected, so that whenever $(\mathbf{x}, \mathbf{x}')$ is a valid transition (of some form (α, α')), so is $(\mathbf{x}', \mathbf{x})$ (in form (α', α)). Finally, as $\mathbf{x}^{i+1} = \mathbf{x}^{i'+1}$, traversal $\mathcal{Z}$ follows $\mathcal{X}$ from this point onward.

Hence, $\mathcal{Z}$ is composed of valid transitions. It also covers G, as it consists of the same set of configurations as $\mathcal{X}$. However, traversal $\mathcal{Z}$ avoids repeating $\mathbf{x}^i = \mathbf{x}^{i'}$ and $\mathbf{x}^{i+1} = \mathbf{x}^{i'+1}$, and so $\mathsf{time}(\mathcal{Z}) = \mathsf{time}(\mathcal{X}) - 2$, a contradiction. □

6 Solving MRFGC in FPT-time

In this section, we develop an algorithm for MRFGC that is FPT in treewidth $\mathsf{tw}(G)$, maximal degree of the graph d, and $\|\mathcal{F}\|$. We use a bottom-up dynamic programming approach on a *nice tree decomposition (Kloks 1994)* (see (Mutzari, Aumann, and Kraus 2024) Appendix B) $\mathcal{T}$ of G. It recursively computes a *table of signatures* at each node of $\mathcal{T}$, starting from the leaves. In our case, a signature at some node j is a sequence of robot configurations that visit the corresponding bag $B_j \in \mathcal{B}$, separated by special $\uparrow$ and $\downarrow$ formal symbols that encode that the robots have "left" the bag. Due to Lemma 1, the sequence length is independent of graph size $|V|$ (Lemma 4). We show we can recursively update the *cost* of each signature bottom-up, and also backtrack a traversal from a signature at the root, by maintaining back-pointers to signatures of children in $\mathcal{T}$.

Let $(\mathcal{B}, \mathcal{T})$ be a nice tree decomposition of G. Here $\mathcal{B} = \{B_j\}_{j\in J}$ is the set of bags ($B_j \subseteq V, |B_j| \le \mathsf{tw} + 1$), and $\mathcal{T}$ is the tree structure over $\mathcal{B}$. For $B \in \mathcal{B}$, denote by $\mathcal{C}_B$ the set of configurations whose set of active vertices intersects B. Let $V_\downarrow(j) = \cup_{j'\in\mathcal{T}_j} B_{j'} \setminus B_j$ be the vertices that appear solely in the bags under B_j. Similarly, let $V_\uparrow(j) = V \setminus V_\downarrow(j) \setminus B_j$ be the vertices that appear solely not in or under j.

Definition 2. *Let $\mathcal{X} = (\mathbf{x}^0, \ldots, \mathbf{x}^t)$ be a traversal and fix a bag $j \in J$. The* projection *of $\mathcal{X}$* on *j, denoted $\mathcal{X}|_j$, is the sequence $\mathcal{Y} = (\mathbf{y}^i)_{0\le i\le t} \in \mathcal{C}_{B_j} \cup \{\uparrow, \downarrow\}$, where for each i:*

$$\mathbf{y}^i = \begin{cases} \mathbf{x}^i & \mathit{Active}(\mathbf{x}^i) \cap B_j \neq \varnothing \\ \uparrow & \mathit{Active}(\mathbf{x}^i) \subseteq V_\uparrow(j) \\ \downarrow & \mathit{Active}(\mathbf{x}^i) \subseteq V_\downarrow(j) \end{cases}$$

The condensed form *of $\mathcal{X}|_j$, denoted $\bar{\mathcal{X}}|_j$, is obtained from $\mathcal{X}|_j$ by replacing any consecutive repetition of the same element with a single occurrence of that element.*

Lemma 3. *Let $\mathcal{X}$ be an optimal traversal and $j \in J$. Then $\bar{\mathcal{X}}|_j = (\bar{\mathbf{x}}|_j^0, \bar{\mathbf{x}}|_j^1, \ldots)$ admits the following:*

1. *If $\bar{\mathbf{x}}|_j^i \notin \{\uparrow, \downarrow\}$, then $\bar{\mathbf{x}}|_j^i \in \mathcal{C}_{B_j}$.*
2. *If $\bar{\mathbf{x}}|_j^i = \bar{\mathbf{x}}|_j^{i'} \in \mathcal{C}_{B_j}$, and $\bar{\mathbf{x}}|_j^{i+1} = \bar{\mathbf{x}}|_j^{i'+1}$, then $\bar{\mathbf{x}}|_j^{i+1} \notin \mathcal{C}_{B_j}$ (*no transition repeats*).*
3. *If $\bar{\mathbf{x}}|_j^i, \bar{\mathbf{x}}|_j^{i+1} \in \mathcal{C}_{B_j}$, then it is a transition.*

Denote by $\mathsf{PS}(j)$ the set of condensed sequences over $\mathcal{C}_{B_j} \times \{\uparrow, \downarrow\}$, for which 1-3 of Lemma 3 hold.

Lemma 4. *There exists an algorithm* enumerate_patterns *and function $h(\|\mathcal{F}\|, d, \mathsf{tw})$ such that given: a graph G, a tree decomposition $(\mathcal{B}, \mathcal{T})$ of G with treewidth* tw*, and a bag $j \in J$, enumerates $\mathsf{PS}(j)$ in time $h(\|\mathcal{F}\|, d, \mathsf{tw})$.*

Proof. Let $\mathcal{Y}$ be a condensed sequence satisfying 1-3. We first bound the length of the sequence $\mathcal{Y}$. Let $v \in B_j$, and let I_v be the set of indices i where $\mathbf{y}^i \in \mathcal{C}_v$, that is, $v \in \mathsf{Active}(\mathbf{y}^i)$. The number of such possible configurations in some α form is bounded by some $f_0(\|\langle G_\alpha\rangle\|, d) = d^{|V_\alpha|-1}$, since formations are represented in explicit form, and the graph monomorphism maps neighbors to neighbors. Therefore, there are up to $f(\|\mathcal{F}\|, d) = \sum_\alpha f_0(\|\langle G_\alpha\rangle\|, d) \le |\mathcal{F}| \cdot d^{\max_\alpha |V_\alpha|}$ possible configurations where v is activated overall. Then, the number of possible transitions from a given configuration with v activated can be similarly bounded by $g_0(\|\mathcal{F}\|, d) = \binom{|\mathcal{F}|}{2} \cdot d^{\max_\alpha |V_\alpha|}$. Fix $g := 2g_0$ to account for transitions where v is activated in the second configuration. Then by the pigeon-hole principle, if $|I_v| > f(\|\mathcal{F}\|, d) \cdot g(\|\mathcal{F}\|, d)$, there is a transition that repeats, violating Condition 2. Therefore, $|\bigcup_{v\in B} I_v| \le f(\|\mathcal{F}\|, d) \cdot g(\|\mathcal{F}\|, d) \cdot (\mathsf{tw}+1)$.

Therefore, the length of a condensed sequence is bounded by $f(\|\mathcal{F}\|, d) \cdot g(\|\mathcal{F}\|, d) \cdot (\mathsf{tw}+1)$, and it is over an alphabet of size $f(\|\mathcal{F}\|, d) \cdot (\mathsf{tw}+1) + 2$. An exhaustive search may enumerate over all such sequences in time $h(\|\mathcal{F}\|, d, \mathsf{tw}) := (f(\|\mathcal{F}\|, d) \cdot (\mathsf{tw}+1) + 2)^{f(\|\mathcal{F}\|, d) \cdot g(\|\mathcal{F}\|, d) \cdot (\mathsf{tw}+1)}$, and exclude any sequence that violates Condition 3. □

6.1 FPT Algorithm

In this section we present our FPT algorithm. We first introduce the table data structure that is computed for each bag.

Data Structure For each bag $j \in J$, create a table table_j of size $|\mathsf{PS}(j)|$. Each row ℓ in the table corresponds to a candidate condensed sequence, and consists of three entries:

1. $\sigma_\ell^j \in \mathsf{PS}(j)$ - the ℓ^{th} condensed sequence on j.
2. $\mathsf{cost}_\ell^j \in \mathbb{N} \cup \{\infty\}$ - the (encountered) minimal number of configurations to cover all of $V_\downarrow(j)$ with a traversal $\mathcal{X}$ such that $\bar{\mathcal{X}}|_j = \sigma_\ell^j$. Initialized to ∞.
3. $\mathsf{pointers}_\ell^j$ - pointers to entries in the tables of the (one or two) children of $j \in J$. Initialized to null.

Some rows are then deleted from the following tables:

- For the root r of $\mathcal{T}$, all rows that contain a pattern with an ↑ symbol are deleted.
- If $\mathbf{x}_0 \in \mathcal{C}_{B_j}$, keep in table_j only rows with patterns that start with configuration $\mathbf{x}_0$.
- If $\mathbf{x}_f \in \mathcal{C}_{B_j}$, keep in table_j only rows with patterns that end with configuration $\mathbf{x}_f$.

Next, we define *reduce*, *lift* and *combine*. These definitions will help express how a signature of a traversal $\mathcal{X}$ at an add, forget and join nodes $j \in J$ respectively, are related to the signature of their children bag(s).

Reduce For a condensed sequence $\mathcal{X} = (\mathbf{x}^i)_i \in \mathcal{C} \cup \{\uparrow, \downarrow\}$, and a set of vertices $A \subseteq V$, let $\mathsf{reduce}(\mathcal{X}, A)$ be the sequence obtained from $\mathcal{X}$ by first changing to ↑ any entry $\mathbf{x}^i$ of $\mathcal{X}$ such that $\mathbf{x}^i \in \mathcal{C}$ and $\mathsf{Active}(\mathbf{x}^i) \cap A = \varnothing$ and then condensing the resultant sequence. Given traversal $\mathcal{X}$ and j is an add node, we have $\bar{\mathcal{X}}|_{j'} = \mathsf{reduce}(\bar{\mathcal{X}}|_j, B_{j'})$.

Lift Similarly, $\mathsf{lift}(\mathcal{X}, A)$ is the condensed sequence obtained from changing to ↓ any $\mathbf{x}^i \in \mathcal{C}$ where $\mathsf{Active}(\mathbf{x}^i) \cap A = \varnothing$. If $j \in J$ is a forget node, $\bar{\mathcal{X}}|_j = \mathsf{lift}(\bar{\mathcal{X}}|_{j'}, B_j)$.

Algorithm 1: UpdateAllTables

for each bag $j \in J$, from leaves up **do** UpdateTable(j);

UpdateTable(j):
for $\sigma_\ell^j \in$ enumerate_patterns(j) **do**
  if j is a `leaf` node in $\mathcal{T}$ **then**
    if $\sigma_\ell^j = (\uparrow)$ **then** set $\mathsf{cost}_\ell^j = 0$;
  else if j is an `add` node, with $B_j = B_{j'} \cup \{v\}$, where j' is the child of j in $\mathcal{T}$ **then**
    Let ℓ' in $\mathsf{table}_{j'}$ satisfy $\sigma_{\ell'}^{j'} = \mathsf{reduce}(\sigma_\ell^j, B_{j'})$;
    if σ_ℓ^j visits v **then** set $\mathsf{cost}_\ell^j = \mathsf{cost}_{\ell'}^{j'}$;
    Set $\mathsf{pointers}_\ell^j := \ell'$;
  else if j is a `forget` node, with $B_j = B_{j'} \setminus \{v\}$, where j' is the child of j in $\mathcal{T}$ **then**
    Let $L' = \{\ell' : \sigma_\ell^j = \mathsf{lift}(\sigma_{\ell'}^{j'}, B_j)\}$.
    Set $\ell' = \arg\min_{\ell' \in L'} \mathsf{cost}_{\ell'}^{j'} + |\{i : (\sigma_{\ell'}^{j'})^i \cap B_j = \varnothing\}|$;
    Set $\mathsf{cost}_\ell^j = \mathsf{cost}_{\ell'}^{j'} + |\{i : (\sigma_{\ell'}^{j'})^i \cap B_j = \varnothing\}|$;
    Set $\mathsf{pointers}_\ell^j := \ell'$;
  else if j is a `join` node, with childs j', j'' in $\mathcal{T}$ **then**
    Let $L^2 = \{(\ell', \ell'') : \sigma_\ell^j$ combines $\sigma_{\ell'}^{j'}$ and $\sigma_{\ell''}^{j''}\}$;
    Set $(\ell', \ell'') = \arg\min_{(\ell', \ell'') \in L^2} \mathsf{cost}_{\ell'}^{j'} + \mathsf{cost}_{\ell''}^{j''}$;
    Set $\mathsf{cost}_\ell^j = \mathsf{cost}_{\ell'}^{j'} + \mathsf{cost}_{\ell''}^{j''}$; Set $\mathsf{pointers}_\ell^j = (\ell', \ell'')$;

Combinations For condensed sequences $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$ of equal length, we say that $\mathcal{X}$ *combines* $\mathcal{Y}$ and $\mathcal{Z}$ if for all i:

- If $\mathbf{x}^i \in \mathcal{C} \cup \{\uparrow\}$ then $\mathbf{x}^i = \mathbf{y}^i = \mathbf{z}^i$.
- If $\mathbf{x}^i = \downarrow$ then either $(\mathbf{y}^i, \mathbf{z}^i) = (\downarrow, \uparrow)$ or $(\mathbf{y}^i, \mathbf{z}^i) = (\uparrow, \downarrow)$.

Indeed, for a given traversal $\mathcal{X}$ and a join node $j \in J$ with children $j', j'' \in J$, $\bar{\mathcal{X}}|_j$ combines $\bar{\mathcal{X}}|_{j'}$ and $\bar{\mathcal{X}}|_{j''}$.

The process The tables for each $j \in J$ are scanned from the leaves in $\mathcal{T}$ up as specified in UpdateAllTables (Algorithm 1). The leaves in $\mathcal{T}$ consist of empty bags, and therefore the only possible condensed sequence is $(\uparrow)$ for which the cost is set to 0. The subroutine UpdateTable is then used to update parent nodes in $\mathcal{T}$ until reaching the root. UpdateTable considers each node type of the nice tree decomposition: *add*, *forget* and *join*, and handles it accordingly.

After updating table_r where $r \in J$ is the root in $\mathcal{T}$, let ℓ be the row with lowest cost. Following the pointers, we can *reconstruct* an optimal traversal $\mathcal{X}$. The detailed proof is provided in (Mutzari, Aumann, and Kraus 2024) Appendix C.

Theorem 1. *MRFGC can be solved in time $\mathcal{O}(n \cdot h(\|\mathcal{F}\|, d, \mathsf{tw}))$, FPT in $\|\mathcal{F}\|, d, \mathsf{tw}$. In particular, MRCGC is FPT in k, d, tw.*

Essentially, Theorem 1 follows from the following two observations. First, if the set of active vertices $\mathsf{Active}(\mathbf{x})$ intersects two bags j', j'', then it must intersect their common parent j. Indeed, by the definition of tree decomposition, removing bag j breaks the graph into disconnected components that separate bag j' from bag j''. By the definition of a formation, the set of activated vertices is a connected sub-graph of G. Therefore, it must intersect bag j. This observation ensures that the costs are updated correctly, and no configuration is double-counted. Second, if two traversals $\mathcal{X}, \mathcal{Y}$ have the same signature at j, we can replace $\mathsf{reduce}(\mathcal{X}, V_{\downarrow}(j))$ with $\mathsf{reduce}(\mathcal{Y}, V_{\downarrow}(j))$, and get a valid traversal. Therefore, picking child signatures with minimal cost ensures an optimal traversal, breaking ties arbitrarily.

Although Theorem 1 is interesting theoretically, the complexity grows quickly with $\|\mathcal{F}\|$, d and tw, as we enumerate over configuration sequences. A natural follow-up question is whether efficient approximation algorithms exist. We next focus on trees ($\mathsf{tw} = 1$), and show that computation time may be independent of d, for a variety of formation families.

7 Approximating MRFTC in PTAS-time

In this section, we study *polynomial time approximation schemes (PTAS)* for trees, that is, MRFTC. A PTAS algorithm is given as input an error parameter ε, and should output a traversal $\mathcal{X}_{\approx}$ that takes $\mathsf{time}(\mathcal{X}_{\approx}) \le t_{\mathsf{optimal}} \cdot (1 + \varepsilon \cdot \mathsf{error}(k, d))$ and runs in time $\mathsf{poly}(n) \cdot h(\frac{1}{\varepsilon}, k)$. We stress that the run-time of the approximation algorithm is independent of the maximal degree d of the tree T. For $k = 3$ connected robots, we show in Section 8 that the approximation error is independent of d as well. For ease of exposition, we assume that $\mathbf{x}_0 = \mathbf{x}_f = r$, the root of the input tree T.

We observe that if $(\mathcal{F}, \mathcal{L})$ is *collapsible*, an approximation of an optimal traversal can be computed in time independent of d. Essentially, a formation is collapsible if the robots are always allowed to get closer. Recall that a *contraction* of a graph G along an edge $e = \{u, v\}$ is a graph G' where u, v are replaced with a single vertex w, and every edge in G that was incident to either u or v is now incident to w in G'.

Definition 5. *We say that $(\mathcal{F}, \mathcal{L})$ is* collapsible *if for each formation $\langle G_\alpha, \mathbf{x}_\alpha \rangle \in \mathcal{F}$ and each* contraction *$G_{\alpha'}$ of G_α, we have $\langle G_\alpha, \{\mathbf{x}_\alpha, \mathbf{x}_{\alpha'}\} \rangle \in \mathcal{L}$. Configuration $\mathbf{x}'_\alpha$ is defined by the graph contraction, where the number of robots of each type at the end-points of the contracted edge is added-up.*

Intuitively, the idea is to cover the tree T with $\mathcal{O}(n\varepsilon)$ sub-trees of size $\mathcal{O}(1/\varepsilon)$, for which an optimal traversal can be found with exhaustive search. Then, the approximate traversal is defined to traverse each such tree optimally, and spend $\mathcal{O}(\|\mathcal{F}\|)$ time to regroup at the root of each sub-tree. It is possible for robots to regroup at some occupied node in $|V_\alpha - 1|$ steps, since we assume $(\mathcal{F}, \mathcal{L})$ are collapsible. Recall that t_{optimal} is the optimal time to cover the tree, denote by $t_{\mathsf{tree\text{-}cover}}$ the sum of optimal times to cover each $\mathcal{O}(1/\varepsilon)$ sub-tree in the cover, and t_{greedy} the time it takes to cover the tree with the greedy algorithm.

Therefore, we have three tasks at hand: (i) efficiently compute an ε tree-coverage; (ii) bound $t_{\mathsf{greedy}} - t_{\mathsf{tree\text{-}cover}} = \mathcal{O}(f_+(\|\mathcal{F}\|)n\varepsilon)$; and (iii) bound $t_{\mathsf{tree\text{-}cover}} - t_{\mathsf{optimal}} = \mathcal{O}(f_-(\|\mathcal{F}\|, d)n\varepsilon)$. We start by defining a tree-cover:

Definition 6. *A* tree-cover *$\mathcal{P} = (\mathcal{T}, \mathcal{V})$ of T is a tree $\mathcal{T}$ and a family $\mathcal{V} = (V_\tau)_{\tau \in V(\mathcal{T})}$ of sub-trees of V such that:*

1. ***Coverage**: It covers V, that is, $\bigcup_{\tau \in V(\mathcal{T})} V_\tau = V$.*
2. ***Small Overlap**: For any $\tau \ne \tau' \in V(\mathcal{T})$, $|V_\tau \cap V_{\tau'}| \le 1$.*
3. ***Connectivity**: For any $\tau \ne \tau' \in V(\mathcal{T})$, $\{\tau, \tau'\} \in E(\mathcal{T})$ iff $|V_\tau \cap V_{\tau'}| = 1$, and the common vertex v is a leaf in τ and a root in τ', or vice versa, or it is the common root.*

Notably, $\mathcal{V}$ uniquely identifies the tree-cover. Therefore, we can denote it by $\mathcal{P}(\mathcal{V})$. Next, we parameterize a tree-coverage with the maximal size of a sub-tree in the coverage:

Definition 7. *Let $T = (V, E, r)$ be a rooted tree and $\varepsilon > 0$. An ε-tree-cover $\mathcal{P} = (\mathcal{T}, \mathcal{V})$ is a tree-cover where each sub-tree $\tau \in \mathcal{T}$ satisfies $|\tau| \le \frac{2}{\varepsilon}$, and $|\mathcal{V}| \le n\varepsilon + 1$.*

TreeCover (Algorithm 2) efficiently computes an ε-tree-cover. In particular, an ε-tree-cover always exists:

Algorithm 2: TreeCover

Input: A rooted tree $T = (V, E, r)$;
Parameter $0 < \varepsilon < 1$;
Output: The size $0 \le \mathsf{size} < \frac{1}{\varepsilon}$ of remaining tree to be covered;
The sub-tree τ remained to be covered;
A family $\mathcal{V}$ of sub-trees of $V \setminus \tau$, with $|\mathcal{V}| \le n\varepsilon$, where the size of each tree is in $[\frac{1}{\varepsilon}, \frac{2}{\varepsilon}]$;
$\mathcal{C}(\mathcal{V} \cup \{\tau\})$ is an ε-tree-cover of T;

```
Initialize size ← 1, τ ← Tree({r}), V ← ∅;
if r is a leaf then
    return size, τ, V;
for u ∈ children(r) do
    size', τ', V' ← TreeCover((V, E, u), ε);
    size += size'; τ.add_subtree.(τ'); V.add(V');
    if size > 1/ε then
        size ← 1; V.add({τ}); τ ← Tree({r});
return V, τ, size;
```

Lemma 8. *TreeCover runs in time $\mathcal{O}(n \log(1/\varepsilon))$ and returns a tree-cover $\mathcal{P}(\mathcal{V} \uplus \{\tau\})$ of T.*

Proof. As for complexity, TreeCover is a DFS search over T starting from the root r. After each visit, basic operations such as adding (a pointer to) a sub-tree, adding (pointers / indices) of sub-trees to the cover, and adding-up / comparing $\mathcal{O}(\log(\frac{1}{\varepsilon}))$-bit numbers.

Coverage. Since DFS will scan the whole tree, eventually all vertices will appear in a sub-tree in $\mathcal{V} \cup \{\tau\}$.

Small Overlap. Once a sub-tree is added to $\mathcal{P}$, all vertices except the root are forgotten, and therefore the intersection can only include the root, which is the current vertex that the DFS algorithm visits.

Connectivity. In $\mathcal{C}$, connectivity holds by definition, however, we must show that $\mathcal{T}$ is indeed a tree. By keeping the root of each tree τ' that is added to $\mathcal{V}$, a path between τ' and the root τ in $\mathcal{T}$ is ensured by induction. Hence, $\mathcal{T}$ is connected. In addition, it contains no cycles as it will translate to a cycle in T. □

In addition, TreeCover outputs an ε-tree-cover:

Algorithm 3: GreedyTraverse

Input: A rooted tree (T, r), a collapsible $(\mathcal{F}, \mathcal{L})$, a tree-cover $\mathcal{P}$;
Output: A traversal $\mathcal{X}_{\approx}$;

Let $\mathcal{V}_r \subseteq \mathcal{V}$ be the sub-trees rooted at r;
for sub-tree τ in $\mathcal{V}_r$ **do**
  $\mathcal{Y}_\tau \leftarrow \mathsf{MRFTC}(\tau, r, k)$;
  Compute the following traversal $\mathcal{X}_\tau$ of sub-tree T_τ:
    Follow $\mathcal{Y}_\tau$;
    if a leaf v of τ is visited for the first time **then**
      Regroup at v;
      Follow $\mathsf{GreedyTraverse}(T_v, v, k, \mathcal{P}|_v)$;
      Get back to the configuration that visited v.
Set $\mathcal{X}_{\approx}$ as the concatenation of the $\mathcal{X}_\tau$'s;
return $\mathcal{X}_{\approx}$;

Lemma 9. *TreeCover runs in time $\mathcal{O}(n \log(\frac{1}{\varepsilon}))$ and returns an ε-tree-cover $\mathcal{P}(\mathcal{V} \uplus \{\tau\})$ of T.*

Proof. By Lemma 8, $\mathcal{P}$ is a tree-cover. The size of each added tree always coincides with size by induction. For the leaves it is 1, and then as long as it doesn't pass $\frac{1}{\varepsilon}$, it is updated by adding the sizes of the sub-trees of each child (which are correct by induction). Once a tree is added to $\mathcal{V}$, all vertices except the root are forgotten and the size of the tree is reset to 1.

Therefore, since sub-trees are added to $\mathcal{V}$ only after checking their size is greater than $\frac{1}{\varepsilon}$, we have that $|\tau| < \frac{1}{\varepsilon}$, as otherwise it would have been added to $\mathcal{V}$.

Now, suppose in contradiction that there is a tree $T_p \in \mathcal{P}$ rooted at u of size greater than $\frac{2}{\varepsilon}$. Then there must be a child of u for which TreeCover returned a tree of size $\geq \frac{1}{\varepsilon}$. But this is a contradiction, since we proved that the size of τ' is always strictly less than $\frac{1}{\varepsilon}$.

Finally, denote by $P := |\mathcal{V}|$. Then $|V| = n = |\tau| + \sum_{\tau' \in \mathcal{V}} |\tau'| \geq 0 + P \cdot \frac{1}{\varepsilon}$. Therefore, $P \leq n\varepsilon$. ☐

7.1 PTAS Algorithm

Next, we describe GreedyTraverse (Algorithm 3). Given an ε tree-coverage $\mathcal{P}$ of T, it computes a traversal of the tree T in time $t_{\mathsf{optimal}}(1 + n\varepsilon \cdot \mathsf{error}_{\mathsf{greedy}}(\|\mathcal{F}\|, d)$. The algorithm runs in time $n f_{\mathsf{greedy}}(\varepsilon, \|\mathcal{F}\|)$, independent of d.

In order to bound $\mathsf{error}_{\mathsf{greedy}}$, we compare it with the total time to cover each sub-tree $\tau \in \mathcal{V}$ individually, denoted $t_{\mathsf{tree\text{-}cover}}$, and show that the difference is bounded by $n\varepsilon \cdot \mathsf{error}_{\mathsf{greedy}}(\|\mathcal{F}\|)$. We show that $t_{\mathsf{tree\text{-}cover}} \leq t_{\mathsf{optimal}} + n\varepsilon h_{\mathsf{tree\text{-}cover}}(\|\mathcal{F}\|, d)$ and conclude that GreedyTraverse is a PTAS algorithm for MRCTC.

Intuitively, GreedyTraverse traverses each sub-tree $\tau \in \mathcal{V}$ in the tree-cover $\mathcal{P}$ optimally. It can do so in time independent of n as the size of each sub-tree of an ε tree-cover is bounded by $2/\varepsilon$. In addition, whenever an optimal traversal of some sub-tree τ visits a leaf v of τ for the first time, all the robots regroup at that leaf. Assuming the robots are in α form, they can regroup at v in $|E_\alpha|$ steps, by contracting all edges in G_α. Then, they traverse the sub-tree rooted at v by recursively calling GreedyTraverse, and then return back to that configuration that visited v. Therefore, $f_+(\|\mathcal{F}\|) := 2\max_\alpha |E_\alpha|$. Note that the robots already start and end at r, so there is no need to account for τ_r. Therefore, $\mathsf{error}_{\mathsf{greedy}} = f_+(\|\mathcal{F}\|)n\varepsilon$.

Finally, we must bound $t_{\mathsf{tree\text{-}cover}} - t_{\mathsf{optimal}}$. In an optimal traversal, robots may cover vertices from different sub-trees $\tau \neq \tau' \in \mathcal{V}$ simultaneously, and therefore it could be that $t_{\mathsf{tree\text{-}cover}} > t_{\mathsf{optimal}}$. In addition, the robots may get in-and-out of a sub-tree in a way that may help them cover the sub-tree faster. We observe that the latter cannot be the case for collapsible formations, since equivalently the robots may wait at the boundary (leaves of τ and its root) instead of leaving it. Hence, we show that the difference is bounded by $n\varepsilon f_-(\|\mathcal{F}\|, d)$.

Lemma 10. $t_{\mathit{tree\text{-}cover}} - t_{\mathit{optimal}} \leq n\varepsilon f_-(\|\mathcal{F}\|, d)$.

Proof. First, we prove that the time to cover a sub-tree τ is the same regardless of whether it is part of a bigger tree T or not. We can view τ as a contraction of T by contracting all the edges that are not in τ. Then, any traversal in T that covers τ is mapped by the contraction to a traversal within τ that covers τ. The latter is a valid traversal of τ since $(\mathcal{F}, \mathcal{L})$ is collapsible.

Next, consider the following traversal $\mathcal{X}_{\approx}$. It follows the optimal traversal $\mathcal{X}$, but every time the robots occupy the root of some sub-tree τ for the first time, the robots regroup at its root τ_r, and then by enumerate over all $f(\|\mathcal{F}\|, d)$ possible configurations in $\mathcal{C}_{\tau_r}$, by following the reverse process of regrouping and then regrouping at τ_r. This takes at most $f_-(\|\mathcal{F}\|, d) := 2\max_\alpha |E_\alpha| \cdot f(\|\mathcal{F}\|, d)$ time. Clearly, $\mathsf{time}(\mathcal{X}_{\approx}) \leq t_{\mathsf{optimal}} + f_-(\|\mathcal{F}\|, d)n\varepsilon$. On the other hand, we also have $t_{\mathsf{tree\text{-}cover}} \leq \mathsf{time}(\mathcal{X}_{\approx})$. ☐

Therefore, we obtain the following result:

Theorem 11. *Assume $(\mathcal{F}, \mathcal{L})$ is collapsible. Then MRFTC is in PTAS. Specifically, there exist an algorithm that computes a traversal of T in time $t_{\mathit{optimal}}(1 + n\varepsilon f_{\approx}(\|\mathcal{F}\|, d))$. The approximate traversal can be computed in time $\mathcal{O}(n \cdot g_{\approx}(\|\mathcal{F}\|, \varepsilon))$.*

8 Approximate 3-Robot MRCTC

For the special case of MRCTC with three robots, we prove that the approximation error of GreedyTraverse is independent of d as well.

Proposition 12. *For $k = 3$, $t_{\mathit{tree\text{-}cover}} - t_{\mathit{optimal}} \leq 52n\varepsilon$.*

Intuitively, we provide a tight analysis of transitions of 3-robot configurations in trees. We classify the transitions into 12 categories, depicted in Figure 2. We use a technique similar to the one used in the proof of the Z-Lemma 1, to prove that there exist an optimal traversal where no transition category repeats, entering the same sub-tree. In such optimal traversal, the number of times that robots enter each sub-tree is bounded by 12, independent of d. We may now construct a sub-optimal traversal that enters each sub-tree once by gluing all visits of a subtree with regrouping at the tree-root. Since the number of regroupings is bounded by 12, and the cost of regrouping is $\leq 2(k-1) = 4$, this results with a traversal that visits each sub-tree separately, and hence its traversal time is $\geq t_{\text{tree-cover}}$. Since $t_{\text{greedy}} - t_{\text{tree-cover}} \leq 2(k-1)n\varepsilon$, we get an overall error that is $\leq 52n\varepsilon k$.

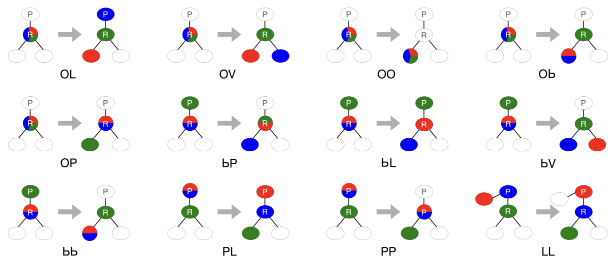

Figure 2: Transition Shapes for $k = 3$ Robots

In Figure 2, we enumerate over all possible *transition shapes*, that start from a configuration that occupies some vertex R with parent P, and end at a configuration that occupies at least one of the children of R. We use red, blue and green for the colors of the three robots, in order to demonstrate which robot goes where. A transition shape specifies how many robots visit R,P, a neighbor of P (other than R), and a child of R, before and after the transition. We also distinguish between a configuration where the same child is occupied by 2 (Ƅ) or 3 (O) robots, and a configuration where two children are occupied (V). An exhaustive search yields 12 such transition shapes. We want to show that any repetition of a transition shape can be avoided:

Lemma 13. *There exists an optimal* tree *traversal with $k = 3$ connected robots, where no transition shape repeats.*

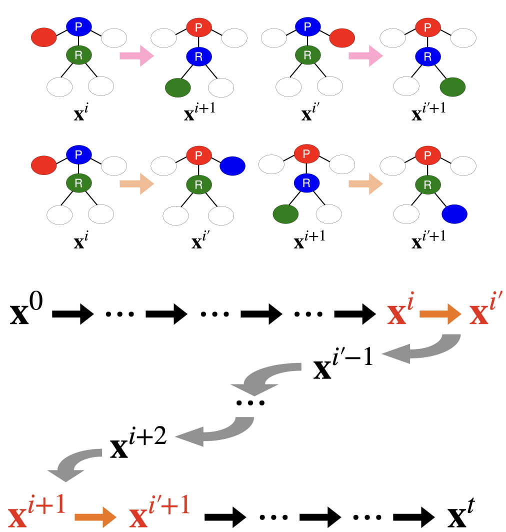

Figure 3: Z-transform for a repeated LL transition shape. Removed transitions are colored in pink and added transitions are colored in orange.

Proof. Let $\mathcal{X}$ be an optimal traversal, and assume in contradiction that a transition shape repeats. We construct a new traversal $\mathcal{Y}$ with $\mathsf{time}(\mathcal{Y}) \leq \mathsf{time}(\mathcal{X})$, that reduces the number of transition shape repetitions by one. We will show how it works for LL transitions, the other 11 cases are provided in (Mutzari, Aumann, and Kraus 2024) Appendix D.

Assume there exist $i < i'$ and $R \in V$, such that the transitions $(\mathbf{x}^i, \mathbf{x}^{i+1})$ and $(\mathbf{x}^{i'}, \mathbf{x}^{i'+1})$ are both LL-transitions that enter the sub-tree rooted at R. Then consider traversal $\mathcal{Y}$ that is defined as follows. Traversal $\mathcal{Y}$ follows $\mathcal{X}$ from initial configuration $\mathbf{x}^0$ up until $\mathbf{x}^i$. It then moves to configuration $\mathbf{x}^{i'}$, and then follows $\mathcal{X}$ in reverse, to configuration $\mathbf{x}^{i'-1}$, and keeps going in reverse until reaching configuration $\mathbf{x}^{i+1}$. It then proceeds to configuration $\mathbf{x}^{i'+1}$, and then follows $\mathcal{X}$ from this point onward. Since $\mathcal{Y}$ consists of the same set of configurations in a different order, it visits all vertices, and is therefore a valid traversal.

Note that $(\mathbf{x}^i, \mathbf{x}^{i'})$ and $(\mathbf{x}^{i+1}, \mathbf{x}^{i'+1})$ are valid transitions specifically for the L-shaped configurations, as depicted in Figure 3. This is not the case in all transition shapes. Indeed, for an ƄV transition, $(\mathbf{x}^{i+1}, \mathbf{x}^{i'+1})$ is not a valid transition. Nevertheless, we the robots can get from $\mathbf{x}^{i+1}$ to $\mathbf{x}^{i'+1}$ by going through a configuration where all robots are at R. While this increases the traversal time by 1, we note that in this case also $\mathbf{x}^i = \mathbf{x}^{i'}$, and therefore the overall traversal time is maintained. This happens in all 12 cases. □

9 Applications

In this section we discuss some applications of MRFGC. We may consider the following applications:

1. *Local communication.* All robots must be within some small communication distance.
2. *Collision avoidance.* Robots cannot occupy the same vertex.
3. *Passage width/material load capacity.* Edges of the graph are parameterized with material load capacity and a passage width. These could restrict the set of robots that can go through an edge simultaneously, that is, reduce the set of valid transitions.
4. *Guards.* For robots of a certain type to cross and edge, the end-points must be occupied by robots of some type. This can model securing a passage before a more valuable robot can cross it.
5. *Cleaning.* The graph must be covered by robots of a certain type. For example in cleanup tasks, one type of robots may be used for cleaning, and another for grinding the garbage.

Note that application 2 is not collapsible, and hence the PTAS algorithm does not apply in this setting. Also, the PTAS algorithm for 3 robots assumes homogeneous robots, and hence only the first application applies.

Preliminary Experimental Results. We tested the performance of our approach (with some heuristics simplifications) on floor plans of several large hotel buildings, and obtained consistent improvement in traversal time compared to the current state-of-the-art approach by (Sinay et al. 2017).

Acknowledgments

The research of Yonatan Aumann is supported in part by ISF grant 3007/24. The research of Sarit Kraus is supported in part by ISF grant 2544/24.

References

Almadhoun, R.; Taha, T.; Seneviratne, L.; and Zweiri, Y. 2019. A survey on multi-robot coverage path planning for model reconstruction and mapping. *SN Applied Sciences*, 1: 1–24.

Banfi, J.; Li, A. Q.; Basilico, N.; Rekleitis, I.; and Amigoni, F. 2016. Asynchronous multirobot exploration under recurrent connectivity constraints. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 5491–5498. IEEE.

Bolu, A.; and Korçak, Ö. 2021. Adaptive task planning for multi-robot smart warehouse. *IEEE Access*, 9: 27346–27358.

Brass, P.; Cabrera-Mora, F.; Gasparri, A.; and Xiao, J. 2011. Multirobot tree and graph exploration. *IEEE Transactions on Robotics*, 27(4): 707–717.

Cabrera-Mora, F.; and Xiao, J. 2012. A flooding algorithm for multirobot exploration. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(3): 850–863.

Charrier, T.; Queffelec, A.; Sankur, O.; and Schwarzentruber, F. 2020. Complexity of planning for connected agents. *Autonomous Agents and Multi-Agent Systems*, 34: 1–31.

Choton, J. C.; and Prabhakar, P. 2023. Optimal multi-robot coverage path planning for agricultural fields using motion dynamics. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 11817–11823. IEEE.

Drew, D. S. 2021. Multi-agent systems for search and rescue applications. *Current Robotics Reports*, 2: 189–200.

Dutta, A.; Ghosh, A.; and Kreidl, O. P. 2019. Multi-robot informative path planning with continuous connectivity constraints. In *2019 international conference on robotics and automation (ICRA)*, 3245–3251. IEEE.

Erdem, E.; Kisa, D.; Oztok, U.; and Schüller, P. 2013. A general formal framework for pathfinding problems with multiple agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 27, 290–296.

Fraigniaud, P.; Gasieniec, L.; Kowalski, D. R.; and Pelc, A. 2006. Collective tree exploration. *Networks: An International Journal*, 48(3): 166–177.

Galceran, E.; and Carreras, M. 2013. A survey on coverage path planning for robotics. *Robotics and Autonomous systems*, 61(12): 1258–1276.

Gans, N. R.; and Rogers, J. G. 2021. Cooperative multirobot systems for military applications. *Current Robotics Reports*, 2: 105–111.

Govindaraju, M.; Fontanelli, D.; Kumar, S. S.; and Pillai, A. S. 2023. Optimized Offline-Coverage Path Planning Algorithm for Multi-Robot for Weeding in Paddy Fields. *IEEE Access*.

Jensen, E. A.; and Gini, M. L. 2018. Online Multi-Robot Coverage: Algorithm Comparisons. In *AAMAS*, 1974–1976.

Kloks, T. 1994. *Treewidth: computations and approximations*. Springer.

Lee, K. M. B.; Kong, F.; Cannizzaro, R.; Palmer, J. L.; Johnson, D.; Yoo, C.; and Fitch, R. 2021. An upper confidence bound for simultaneous exploration and exploitation in heterogeneous multi-robot systems. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 8685–8691. IEEE.

Lee, S. 2023. An efficient coverage area re-assignment strategy for multi-robot long-term surveillance. *Ieee Access*, 11: 33757–33767.

Lu, J.; Zeng, B.; Tang, J.; Lam, T. L.; and Wen, J. 2023. Tmstc*: A path planning algorithm for minimizing turns in multi-robot coverage. *IEEE Robotics and Automation Letters*.

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

Miao, X.; Lee, H.-S.; and Kang, B.-Y. 2020. Multi-cleaning robots using cleaning distribution method based on map decomposition in large environments. *IEEE Access*, 8: 97873–97889.

Mukhamediev, R. I.; Yakunin, K.; Aubakirov, M.; Assanov, I.; Kuchin, Y.; Symagulov, A.; Levashenko, V.; Zaitseva, E.; Sokolov, D.; and Amirgaliyev, Y. 2023. Coverage path planning optimization of heterogeneous UAVs group for precision agriculture. *IEEE Access*, 11: 5789–5803.

Mutzari, D.; Aumann, Y.; and Kraus, S. 2024. Heterogeneous Multi-Robot Graph Coverage with Proximity and Movement Constraints. arXiv:2412.10083.

Nemoto, T.; and Mohan, R. E. 2020. Heterogeneous multi-robot cleaning system: State and parameter estimation. *Automation in Construction*, 109: 102968.

Panerati, J.; Gianoli, L.; Pinciroli, C.; Shabah, A.; Nicolescu, G.; and Beltrame, G. 2018. From swarms to stars: Task coverage in robot swarms with connectivity constraints. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 7674–7681. IEEE.

Queralta, J. P.; Taipalmaa, J.; Pullinen, B. C.; Sarker, V. K.; Gia, T. N.; Tenhunen, H.; Gabbouj, M.; Raitoharju, J.; and Westerlund, T. 2020. Collaborative multi-robot search and rescue: Planning, coordination, perception, and active vision. *Ieee Access*, 8: 191617–191643.

Rodríguez, M.; Al-Kaff, A.; Madridano, Á.; Martín, D.; and de la Escalera, A. 2020. Wilderness search and rescue with heterogeneous multi-robot systems. In *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*, 110–116. IEEE.

Salzman, O.; and Stern, R. 2020. Research challenges and opportunities in multi-agent path finding and multi-agent pickup and delivery problems. In *Proceedings of the 19th International Conference on Autonomous Agents and Multi-Agent Systems*, 1711–1715.

Scherer, J.; and Rinner, B. 2020. Multi-robot persistent surveillance with connectivity constraints. *IEEE Access*, 8: 15093–15109.

Siligardi, L.; Panerati, J.; Kaufmann, M.; Minelli, M.; Ghedini, C.; Beltrame, G.; and Sabattini, L. 2019. Robust area coverage with connectivity maintenance. In *2019 International Conference on Robotics and Automation (ICRA)*, 2202–2208. IEEE.

Sinay, M.; Agmon, N.; Maksimov, O.; Kraus, S.; and Peleg, D. 2017. Maintaining Communication in Multi-Robot Tree Coverage. In *IJCAI*, 4515–4522.

Tang, J.; Sun, C.; and Zhang, X. 2021. MSTC*: Multi-robot coverage path planning under physical constraints. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2518–2524. IEEE.

Vallejo, D.; Castro-Schez, J. J.; Glez-Morcillo, C.; and Albusac, J. 2020. Multi-agent architecture for information retrieval and intelligent monitoring by UAVs in known environments affected by catastrophes. *Engineering Applications of Artificial Intelligence*, 87: 103243.

Wang, G.; Wang, W.; Ding, P.; Liu, Y.; Wang, H.; Fan, Z.; Bai, H.; Hongbiao, Z.; and Du, Z. 2023. Development of a search and rescue robot system for the underground building environment. *Journal of Field Robotics*, 40(3): 655–683.

Yang, Q.; and Parasuraman, R. 2020. Needs-driven heterogeneous multi-robot cooperation in rescue missions. In *2020 ieee international symposium on safety, security, and rescue robotics (ssrr)*, 252–259. IEEE.

Zaccaria, M.; Giorgini, M.; Monica, R.; and Aleotti, J. 2021. Multi-robot multiple camera people detection and tracking in automated warehouses. In *2021 IEEE 19th International Conference on Industrial Informatics (INDIN)*, 1–6. IEEE.

Zhang, C.; Zhan, Q.; Wang, Q.; Wu, H.; He, T.; and An, Y. 2020. Autonomous dam surveillance robot system based on multi-sensor fusion. *Sensors*, 20(4): 1097.

Zhang, Y.; Fontaine, M. C.; Bhatt, V.; Nikolaidis, S.; and Li, J. 2024. Multi-robot coordination and layout design for automated warehousing. In *Proceedings of the International Symposium on Combinatorial Search*, volume 17, 305–306.