Universal Solvability for Robot Motion Planning on Graphs

Anubhav Dhar*
Max Planck Institute for Informatics
Saarbrücken, Saarland, Germany
anubhavldhar@gmail.com

Pranav Nyati*
Indian Institute of Technology
Kharagpur, India
pranavnyati26@gmail.com

Tanishq Prasad*
Indian Institute of Technology
Kharagpur, India
tanishqprasad1003@gmail.com

Ashlesha Hota*
Indian Institute of Technology
Kharagpur, India
ashleshahota.23@kgpian.iitkgp.ac.in

Sudeshna Kolay
Indian Institute of Technology
Kharagpur, India
skolay@cse.iitkgp.ac.in

ABSTRACT

Autonomous multi-robot systems in modern warehouses often need robots to be rearranged for efficient task execution. A key question is whether the underlying layout allows transformation between any two configurations without collisions. We model the layout as a graph $G(V, E)$, with vertices as locations and edges as paths, and study the Universal Solvability of Robot Motion Planning on Graphs (USolR) problem: given G and p robots, does G allow any configuration to be transformed into any other via valid moves? For this, we design a linear-time randomized algorithm with one-sided error that always correctly identifies universally solvable graphs and may fail only on non-universally solvable instances; derandomization incurs a factor-p overhead, giving deterministic running times of $O(p(|V| + |E|))$ for sparse and $O(|V| + |E|)$ for dense graphs. Finally, we consider the Graph Edge Augmentation for Universal Solvability (EAUS) problem: given a connected graph G not universally solvable for p robots, can at most β edges be added to make it universally solvable? We show an upper bound of $p-2$ on β for general graphs and give examples requiring $\Theta(p)$ edge additions. We then study Graph Vertex and Edge Augmentation for Universal Solvability (VEAUS), where α vertices and β edges may be added, and provide lower bounds on these parameters.

KEYWORDS

Robot Motion Planning; Equivalence classes; Randomized Algorithms; Universal Solvability

ACM Reference Format:
Anubhav Dhar, Pranav Nyati, Tanishq Prasad, Ashlesha Hota, and Sudeshna Kolay. 2026. Universal Solvability for Robot Motion Planning on Graphs. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 9 pages. https://doi.org/10.65109/VEXW9837

*These authors contributed equally to this work.

Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus. © 2026 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). https://doi.org/10.65109/VEXW9837

1 INTRODUCTION

Multi-Robot Motion Planning is a fundamental problem in robotics, AI, and computational geometry. It studies how to coordinate multiple robots to move collision-free so that any configuration can reach any other. Attaining this full reachability is of utmost importance in practical systems such as automated warehouses, factories, or swarm robotics. Without full reachability, certain configurations may become inaccessible, causing inefficiencies, bottlenecks, or even system failures. This motivates the study of the Universal Solvability of Robot Motion Planning on Graphs (USolR) problem within the graph-based planning framework. In this work, we model the workspace as an undirected graph $G(V, E)$, where vertices represent discrete locations and edges represent paths. We consider $p \leq |V|$ robots, each occupying a distinct vertex. At each step, a robot may stay or move to an adjacent unoccupied vertex, but adjacent robots cannot swap positions along an edge in a single move. A configuration of the system at time t specifies the positions of all robots on the graph at that time. A move is defined as a transition between two such configurations. Let S denote the initial configuration, and let T denote the final desired configuration. We require the robots to move from S to T without any collisions or deadlocks. As a practical example, consider a warehouse where

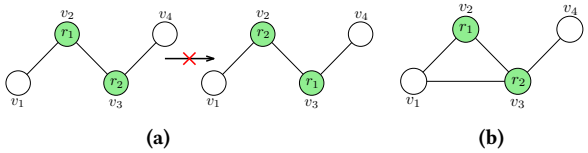

Figure 1: Illustration of reconfiguration instances: (a) a NO instance where the robots cannot be swapped, and (b) a YES instance where any target configuration is reachable.

autonomous robots move items between storage shelves and packing stations. The warehouse can be modeled as a graph $G(V, E)$, with vertices representing locations like storage shelves or packing stations and edges representing traversable paths. Suppose p robots operate simultaneously, each occupying a unique vertex. Over time, the system may need to rearrange robots from an initial configuration S to a target configuration T. To perform the desired task efficiently, the system requires that such transformations be possible for any pair of configurations.

In USolR, we address the following decision question: Given a graph $G(V, E)$ and an integer p, can any configuration S of p

robots be transformed into any other configuration T via a sequence of valid moves? As shown in Figure 1, in Figure 1a the initial configuration is $s_1 = v_2$ and $s_2 = v_3$. We ask whether it is possible to reconfigure the robots to any desired target positions. This instance is a *NO* instance, since it is impossible to transform it to the target configuration $t_1 = v_3$ and $t_2 = v_2$ via valid moves.

In contrast, Figure 1b shows an *YES* instance, where the configuration can be reconfigured to any desired target placement. Graphs with this property are called *universally solvable*. Not all graphs satisfy this property; for example, the presence of a cut edge may block reachability between certain configurations. This motivates augmentation questions: can a non-universally solvable graph be made universally solvable by adding at most β edges? More generally, if edge additions alone are insufficient, can universal solvability be achieved by adding α vertices along with β edges? For example, a star graph with one robot per vertex is not universally solvable and cannot be fixed by few edge additions, but adding two vertices adjacent to the center suffices. Designing minimal augmentations to ensure full reconfigurability is important in practical settings where structural changes are costly.

Our work is also motivated by prior studies on coordinated motion planning in restricted environments [1, 16, 18, 21], which give efficient algorithms for deciding reachability between a given pair of configurations S and T, but do not address whether *all* configurations are mutually reachable. Although group-theoretic methods in [16] characterize certain graph classes with full reachability, the notion of *universal solvability* in general graphs remains largely unexplored. We bridge this gap by studying the USolR problem from both structural and algorithmic perspectives.

1.1 Related Work

Robot motion planning has received significant attention, both in theoretical and applied contexts over the past few decades [13, 14, 19]. Closely related problems are: token swapping [3, 22] and pebble motion on graphs [10, 12, 15, 16, 20]. More recently, coordinated motion planning has been studied with respect to minimising various target objectives, such as the makespan of the schedule of the robots [2, 8, 23] and the total distance travelled by all robots [5, 8, 11, 23]. Problems related to robot motion planning arise in robotics [4], motion planing [6], puzzle design [17], and distributed systems [9], and have motivated extensive research on their complexity, solvability, and algorithmic strategies. Now, we discuss a few key contributions related to the feasibility variants of robot motion planning.

Kornhauser et al. [16] study the Pebble Motion problem on graphs, where $p < n$ labelled pebbles are placed on distinct vertices of an n-vertex graph G, and the goal is to reach a target configuration via moves to adjacent unoccupied vertices. They reduce the problem to a *permutation group* setting, where valid move sequences form a group $R(P)$. For various graph classes (e.g., biconnected, separable, trees), $R(P)$ is shown to be transitive or decomposable into transitive components. The authors prove that $R(P)$ contains either the *alternating group* $\mathbf{A}_p$ or the *symmetric group* $\mathbf{S}_p$, depending on the structure of G. Auletta et al. [1] address the Pebble Motion problem on trees and present a linear-time algorithm that reduces it to the Pebble Permutation problem, where the set of occupied vertices remains the same and the goal is to realise a given permutation of the pebbles in these vertices through a sequence of valid moves. Their approach yields an $O(n)$-time decision algorithm and an $O(n + \ell)$-time constructive algorithm, where ℓ is the length of the move sequence and n is the number of vertices.

Yu and Rus [24] generalise the classical Pebble Motion problem to *pebble motion with rotations* (PMR) for p pebbles on an n-vertex graph, where in addition to simple moves into unoccupied vertices ($p < n$), synchronous cyclic rotations along disjoint cycles are also allowed. In the fully packed case ($p = n$), only rotations are feasible. They model reachable configurations as elements of a subgroup $\mathbf{G} \leq \mathbf{S}_n$, generated by these rotations. For 2-edge connected but not 2-connected graphs, they prove that $\mathbf{G}$ contains either the alternating group $\mathbf{A}_n$ or the symmetric group $\mathbf{S}_n$, and establish an upper bound of $\mathrm{diam}(\mathbf{G}) = O(n^2)$, ensuring reachability in polynomial time. These results extend to general 2-edge-connected graphs and enable a unified framework where feasibility can be tested in linear time, and a complete plan can be computed in $O(n^3)$ time for all values of $p \leq n$.

While most existing works focus on checking the feasibility of transforming a specific start configuration S into a target configuration T, relatively little attention has been paid to the question of *universal solvability*: whether all configurations on a given graph are mutually reachable. This motivates our study, which aims to analyse the universal solvability of general graphs and bridge this gap in the literature.

1.2 Our Contributions

Our primary contributions are as follows:

(1) **Structural characterization:** Inspired by [24], we prove that all equivalence classes have equal size, implying that the number of reachable configurations is a factor of the total number of configurations when $p = |V|$, and a similar argument extends to $p \leq |V|$. This structural insight underpins our algorithmic results.
(2) **Efficient algorithms:** We give a linear-time randomized algorithm with one-sided error for deciding USolR. Derandomization yields deterministic algorithms running in $O(p(|V| + |E|))$ for sparse graphs and $O(|V| + |E|)$ for dense graphs, improving on earlier deterministic quadratic-time methods for planar graphs [24].
(3) **Optimized deterministic algorithm:** Building on the randomized algorithm, we first derandomize it to check universal solvability using only $p - 1$ carefully chosen reachability tests, yielding a polynomial-time deterministic algorithm. We then present a more involved, nontrivial optimized deterministic algorithm that leverages structural properties of the graph such as connectivity and sufficiently large 2-connected components—to reduce unnecessary reachability checks. This optimized algorithm runs in $O(p \cdot (|V| + |E|))$ for sparse graphs, which further reduces to $O(|E|)$ for dense graphs.
(4) **Edge augmentation (EAUS):** We bound the number β of edges needed to make a connected graph universally solvable for p robots, showing $\beta \leq p - 2$ in general and $\Theta(p)$ is tight in the worst case.

(5) **Vertex and edge augmentation (VEAUS):** Allowing α added vertices and β edges, we prove lower bounds showing that for an infinite family of graphs, $\alpha = \Omega(\sqrt{|V|})$ and $\beta = \Theta(\sqrt{|V|})$ are necessary when $p = |V|$.

Due to space constraints, some results (marked with ⋆), including the Accumulation section, are deferred to the full version [7].

2 PRELIMINARIES

Basic Notations. We denote the set of positive integers by $\mathbb{N}$ and the set of integers by $\mathbb{Z}$. For $k \in \mathbb{N}$, we denote the set $\{1, 2, \ldots, k\}$ by $[k]$. For a set S, we denote its power set (family of all subsets of S) by $2^S = \{S' \mid S' \subseteq S\}$. For $k \in \mathbb{N}$, we denote the family of subsets of S of size k by $\binom{S}{k}$. For a function $f : X \to Y$, and a subset $X' \subseteq X$, the set of images of X' is denoted by $f(X') = \{f(x) \mid x \in X'\}$. Further, let $\mathfrak{S}_p$ be the set of all permutations from $[p]$ to $[p]$, thus, $|\mathfrak{S}_p| = p!$. For functions $f : X_1 \to X_2$, and $g : X_2 \to X_3$, we denote their composition as $f \circ g : X_1 \to X_3$, i.e., $f \circ g(x) = f(g(x))$ for all $x \in X_1$.

Graph Theory. We denote by $G(V, E)$, an undirected graph with vertex set V and edge set E. Throughout this paper, we only consider undirected graphs. A cut vertex (resp. edge) of a graph is a vertex (resp. edge) whose deletion increases the number of connected components. The edge connectivity, $\lambda(G)$ of a graph $G(V, E)$ is the minimum number of edges required to be deleted to make the graph disconnected. We say that a graph $G(V, E)$ is k-edge connected if $\lambda(G) \geq k$. The vertex connectivity, $\kappa(G)$ of a graph $G(V, E)$, is the minimum number of vertices required to be deleted to make the graph disconnected. We say that a graph $G(V, E)$ is k-connected if $\kappa(G) \geq k$.

Motion Planning. We consider an underlying undirected graph $G(V, E)$ with $|V| = n$ and $|E| = m$. Consider a set R of $p \in \mathbb{N}$ robots. For convenience, we assume the robots are named 1, 2, …, p; hence $R = [p]$. We use both R and $[p]$ to denote the set of robots, depending on the context. A *configuration*, $S : R \to V$ is an injective map from the set of robots to the set of vertices of the underlying graph $G(V, E)$; for robot i, $S(i)$ denotes the vertex occupied by it. $S(R) \subseteq V$ denotes the set of vertices occupied by the set of robots R. The configuration is an injective map as it models the constraint that no two robots can occupy the same vertex in a configuration, i.e., $i \neq j \implies S(i) \neq S(j)$. We denote by $\mathcal{S}_G = \{S : R \to V \mid S \text{ is a configuration}\}$ the set of all configurations of the robots in R that are possible on the graph G. Note that $|\mathcal{S}_G| = \frac{n!}{(n-p)!}$.

Valid Moves. We now formally define the notion of valid/feasible moves in terms of pairs of configurations that can be reached from each other by one such move. We introduce three different types of valid moves as follows:

- **Simple Path Move:** A simple path move is defined as a pair of configurations that can be achieved from one another by sliding robots along a path synchronously, as shown in Figure 2. Formally, for configurations S, S', the pair (S, S') defines a simple path move if there is a path $P = (u_0, u_1, u_2, \ldots, u_k)$ in G such that,
  - $u_k \notin S(R)$ (empty in S) and $u_0 \notin S'(R)$ (empty in S').
  - For all $i \in R$ such that $S(i) \notin P$, we have $S'(i) = S(i)$ (robots outside the path P do not move).
  - For all $i \in R$ such that $S(i) = u_j$, with $j \in \{0, 1, \ldots, k-1\}$, we have $S'(i) = u_{j+1}$ (robots in P synchronously move one unit each).

  We say that S' is obtained by pushing S along the path P.

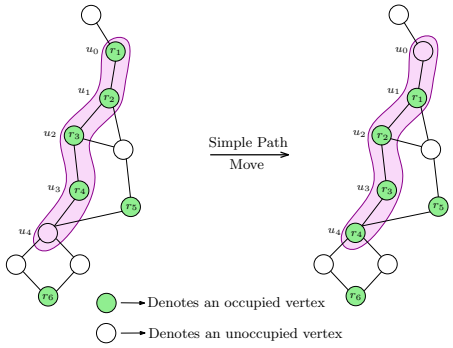

Figure 2: Simple path move, where r_i denotes robot i.

- **Simple Rotation Move:** A simple rotation move is a move along a simple cycle (all whose vertices are occupied by robots) such that all the vertices in that cycle synchronously move to their immediate neighbour in either clockwise or anticlockwise direction in one such move.Formally, for configurations S, S', the pair (S, S') defines a simple rotation if there is a cycle $C = (u_0, u_1, u_2, \ldots, u_k = u_0)$ in G such that,
  - $\{u_0, \ldots, u_{k-1}\} \subseteq S(R)$ (occupied vertices in S) and $\{u_0, \ldots, u_{k-1}\} \subseteq S'(R)$ (occupied vertices in S').
  - For all $i \in R$ such that $S(i) \notin C$, we have $S'(i) = S(i)$ (robots outside the cycle C do not move).
  - For all $i \in R$ such that $S(i) = u_j$, with $j \in \{0, 1, \ldots, k-1\}$, we have $S'(i) = u_{j+1}$ (robots in C synchronously move one unit each).
- **Dummy Move:** A dummy move is the pair (S, S) for any configuration S; this corresponds to not moving any robot from its occupied vertex in S.

A *valid move* is either a simple path move, a simple rotation move, or a dummy move. Note that we do not allow robots to swap along an edge as a move on its own (see, Figure 1a). Trivially, (S, S') is a valid move if and only if (S', S) is a valid move as it is an undirected graph, and each of these valid moves are reversible.

Further, for a configuration S and a permutation $\pi : [p] \to [p]$, their composition $S \circ \pi$, corresponds to another configuration where robots are renamed according to the permutation π^{-1}; if robot i occupies vertex $S(i)$ in S, then robot $\pi^{-1}(i)$ also occupies vertex $S \circ \pi(\pi^{-1}(i)) = S(i)$ in the configuration $S \circ \pi$, as illustrated in Figure 3. We state an observation regarding valid moves.

Observation 2.1. *For configurations S and S', (S, S') is a valid move if and only if $(S \circ \pi, S' \circ \pi)$ is a valid move for any permutation $\pi : [p] \to [p]$.*

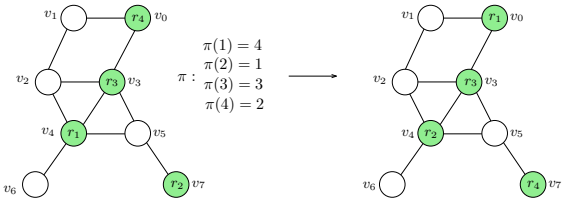

Figure 3: Composition of a configuration over a permutation, where r_i denotes robot i.

2.1 Equivalence Relation based on Reachable Configurations

We now define what is meant by a configuration to be reachable from another configuration and define a binary relation based on reachability. We will use this to define equivalence classes which would further help us in analysing the structure of the problem instance.

Definition 2.1 (Reachability of configurations). *We say that a configuration T is reachable from S, if there is a finite sequence of configurations $(S_0, S_1, \ldots, S_t)$ with $S_0 = S$, $S_t = T$ where (S_{k-1}, S_k) is a valid move for all $k \in [t]$.*

Informally, T is reachable from S if, for all $i \in R$, robot i starts at vertex $S(i)$, and after a sequence of valid moves, it reaches vertex $T(i)$. Since moves are reversible, for an undirected graph, if T is reachable from S, then S is also reachable from T. Further, S is reachable from itself as a dummy move is a valid move.

Based on the above definition of reachability, a *homogeneous binary relation* (i.e., a binary relation between a set and itself) $\tilde{R}$, can be defined as follows:

Definition 2.2. *$\tilde{R}$ is the homogeneous binary relation over the set of all configurations $\mathcal{S}_G$ defined by reachability as,*

$$\tilde{R} = \{(S, T) \mid S, T \in \mathcal{S}_G \text{ and } T \text{ is reachable from } S\}$$

We denote $(S, T) \in \tilde{R}$ as $S \sim T$ and $(S, T) \notin \tilde{R}$ as $S \nsim T$.

The above relation $\tilde{R}$ is an *equivalence relation*, as it is reflixive, symmetric and transitive. Since an equivalence relation provides a partition of the underlying set into disjoint equivalence classes, the reachability relation $\tilde{R}$ also partitions the set $\mathcal{S}_G$ into disjoint equivalence classes based on mutual reachability. This gives us a generalisation of Observation 2.1.

Observation 2.2. *For configurations S and S', $S \sim S'$ if and only if $S \circ \pi \sim S' \circ \pi$ for any permutation $\pi : [p] \to [p]$.*

Proof. Suppose $S \sim S'$. Then there exists a sequence $(S_0 = S, S_1, \ldots, S_t = S')$ where each (S_{k-1}, S_k) is a valid move. By Observation 2.1, every $(S_{k-1} \circ \pi, S_k \circ \pi)$ is also a valid move. Thus, $(S_0 \circ \pi, S_1 \circ \pi, \ldots, S_t \circ \pi)$ is a valid sequence, and therefore $S \circ \pi \sim S' \circ \pi$. □

2.2 Problem Definition

Let us define the problem Universal Solvability of Robot Motion Planning on Graphs (USolR).

> Universal Solvability of Robot Motion Planning on Graphs (USolR)
> **Input:** A graph $G(V, E)$, an integer $p \in \mathbb{N}$ $(2 \le p \le |V|)$
> **Question:** Decide if for all configurations S, T of p robots on the underlying graph G, are S and T reachable from each other using valid moves

In this paper we also require the definition of a related problem, Feasibility of Robot Motion Planning on Graphs.

> Feasibility of Robot Motion Planning on Graphs (FRMP)
> **Input:** A graph $G(V, E)$, an integer $p \in \mathbb{N}$ $(2 \le p \le |V|)$, a pair of robot configurations (S, T)
> **Question:** Decide if the configuration T is reachable from S using only valid moves on the underlying graph G

Yu and Rus [24] propose an algorithm, which we denote as $\mathcal{A}$ in our work, that solves the FRMP problem for a given instance consisting $(G(V, E), p, (S, T))$ in time $O(|V| + |E|)$. They prove the correctness and time-complexity of $\mathcal{A}$ in Theorem 20 in their paper, which we restate as the following proposition.

Proposition 2.1. *[24] Given an instance $(G(V, E), p, (S, T))$ of FRMP, there exists a deterministic algorithm, denoted as Algorithm $\mathcal{A}$, to decide if T is reachable from S using only valid moves on G, that runs in time $O(|V| + |E|)$.*

3 [⋆] ACCUMULATING ROBOTS TO SPECIFIC VERTICES

We now describe how to accumulate robots into specific vertices using valid moves. This algorithm serves to reveal structural properties of configurations rather than acting as a standard subroutine.

For a connected graph $G(V, E)$ with a fixed total ordering Intr on V, and let $R = [p]$ denote the robots, a deterministic BFS starting from the least indexed vertex enumerates the vertices in a specific order: $v_1, v_2, \ldots, v_n$. For a system of p robots, we define $V_p = \{v_1, v_2, \ldots, v_p\}$, representing the first p visited vertices. The

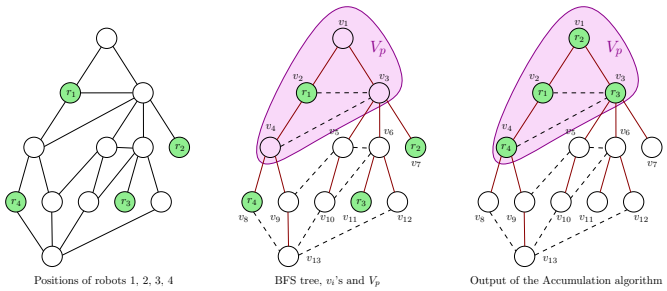

Figure 4: The accumulation procedure in action, illustrating a robot moving along the unique BFS tree path towards the least unoccupied target vertex.

following key lemma establishes that we can accumulate robots to any arbitrary subset of vertices using these moves.

Lemma 3.1 (⋆). *Consider a graph $G(V, E)$, a set of robots $R = [p]$ and an intrinsic ordering* Intr *of V. Let $S : R \to V$ be any arbitrary*

configuration. Let $X \subseteq V$ be any arbitrary subset of size $|X| = p$. There exists a configuration $T : R \to V$ such that $T(R) = X$ and $S \sim T$.

Proof Sketch. The accumulation procedure operates on a BFS tree $\mathfrak{T}$ of G, generated according to the intrinsic ordering Intr. If the current occupied set does not match the target set X (for instance, if $X = V_p$), we identify the least unoccupied target vertex $v_a \in X$ and the least occupied vertex outside the target set, $v_b \notin X$.

We then identify the unique path P between v_b and v_a in $\mathfrak{T}$, and perform a simple path move for the robot along P. Repeating this entire procedure on the resulting configurations continually advances robots toward X. This strictly decreases the distance to the target state, ensuring the algorithm always terminates in finite time with the robots occupying the desired subset. □

The pseudocode of this algorithm, named Algorithm 1 is given below.

Algorithm 1 **Input:** $G(V, E)$, a configuration S of p robots

$(v_1, \ldots, v_n) \leftarrow$ BFS ordering from running BFS on G
$\mathfrak{T} \leftarrow$ BFS tree from running BFS on G
$V_p \leftarrow \{v_1, \ldots, v_p\}$
while $S(R) \neq V_p$ **do**
  $a \leftarrow \min\{i \in [p] \mid v_i \notin S(R)\}$
  $b \leftarrow \min\{i \in [n] \setminus [p] \mid v_i \in S(R)\}$
  $P \leftarrow$ unique path from v_b to v_a in $\mathfrak{T}$
  $S' \leftarrow$ configuration obtained by pushing S along P
  $S \leftarrow S'$
end while
return S'

Please refer to Figure 4 for a visual representation of the execution of the algorithm. By construction, if S' is the output of Algorithm 1 when the configuration S is part of the input, then $S'(R) = V_p$.

4 RANDOMIZED ALGORITHM FOR USOLR

In this section, we design a polynomial-time randomized algorithm for the USolR problem based on the sizes of the equivalence classes of the relation $\tilde{R}$ (Definiton 2.2). We also prove that this algorithm can be derandomized in polynomial time.

The notions of the reachability relation and equivalence classes have resemblance to groups and subgroups. These similarities, as well as the group-theoretic techniques used in the works of Yu and Rus [24] motivated us to deeply examine the structure of these equivalence classes of the reachability relation $\tilde{R}$, and we provide a characterisation of the equivalence classes of $\tilde{R}$.

Let $(G(V, E), p)$ be the instance of USolR, and let $V_p = \{v_1, v_2, \ldots, v_p\}$ be as defined in Section 3. Consider the identity configuration S_I with $S_I(i) = v_i$ for all $i \in [p]$. For configuration S_I, we have $S_I(R) = V_p$, so $Y_{S_I} = S_I$.

For a graph $G(V, E)$ to be universally solvable for p robots, it must hold true that for any pair of configurations $S, T \in \mathcal{S}_G$, configuration T is reachable from S, i.e., $S \sim T$. Thus, for any configuration $S \in \mathcal{S}_G$, $S_I \sim S$ must also hold true. Further, from the discussion in Section 2, we have that $\tilde{R}$ is an equivalence relation, and since all configurations in $\mathcal{S}_G$ belong to the same equivalence class as S_I, there is essentially only one equivalence class comprising of all configurations $S \in \mathcal{S}_G$. Let us denote by $\mathcal{E}_S$, the set of configurations reachable from the configuration S. Then, for a YES-instance, we have $\mathcal{E}_{S_I} = \mathcal{S}_G$.

On the contrary, for a NO-instance, we have at least one configuration that is not reachable from the configuration S_I. Let S_I be denoted by S_0^*. Let $S_0^*, S_1^*, S_2^*, .., S_k^*$ be configurations that are not reachable from each other, and the equivalence classes $\mathcal{E}_{S_0^*}$, $\mathcal{E}_{S_1^*}$, $\mathcal{E}_{S_2^*}$, ..., $\mathcal{E}_{S_k^*}$ exhaust the entire set $\mathcal{S}_G$, i.e., $\mathcal{S}_G = \bigcup_{i=0}^{k} \mathcal{E}_{S_i^*}$. Consider one of such configurations S_j^*, for $j \geq 1$ that is not reachable from $S_I (= S_0^*)$. We now show that the equivalence classes $\mathcal{E}_{S_I}$ and $\mathcal{E}_{S_j^*}$ have the same size. This essentially means $|\mathcal{E}_S| = |\mathcal{E}_T|$ for all configurations $S, T \in \mathcal{S}_G$. We will be crucially using this result to design an efficient randomized algorithm.

Lemma 4.1 (★). *Consider a graph $G(V, E)$, set of robots $R = [p]$ and an intrinsic ordering* Intr *of V. Let $\tilde{R}$ be the equivalence relation on configurations as per Definition 2.2. Then the size of the equivalence class $\mathcal{E}_{S_I}$ equals the size of the equivalence class $\mathcal{E}_{S_j^*}$ for all $j \in \{1, 2, \ldots, k\}$.*

This allows us to prove a crucial result, which helps us in designing an efficient randomized algorithm.

Lemma 4.2 (★). *Let $G(V, E)$ with p robots be a NO-instance of USolR, and let $\mathcal{S}_G$ denote the set of all configurations over G, then there exist at least $|\mathcal{S}_G|/2$ many distinct configurations $S : R \to V$ in $\mathcal{S}_G$ such that S is not reachable from S_I.*

We now propose in the following theorem, a randomized algorithm for USolR based on the properties of equivalence classes discussed above.Given an instance $(G(V, E), p)$, the algorithm correctly outputs YES for a YES-instance (universally solvable instance) with probability 1 and outputs YES for a NO-instance (not universally solvable instance) with probability at most $1/2$.

Theorem 4.1. *Given a graph $G(V, E)$ and p robots, Algorithm 2 correctly outputs YES for a universally solvable instance with probability 1, and outputs YES for a not universally solvable instance with probability at most $1/2$. The running time of this algorithm is $O(|V| + |E|)$.*

Proof. Given an instance $(G(V, E), p)$ for USolR, we propose a randomized algorithm, denoted by Algorithm 2. Algorithm 2 first finds the connected components of the given graph G using a Depth-First Search algorithm. If there are 2 or more connected components in G, then it outputs NO and terminates, as a disconnected graph is never universally solvable. Next, Algorithm 2 samples a configuration S_R uniformly at random from the set of all configurations $\mathcal{S}_G = \{S : R \to V \mid S \text{ is a configuration}\}$. Then, it runs Algorithm $\mathcal{A}$ (Proposition 2.1) as a subroutine, with $(G(V, E), p, (S_I, S_R))$ as input to check if S_R is reachable from S_I. If Algorithm $\mathcal{A}$ outputs YES for reachability of S_R from S_I, then Algorithm 2 outputs that G is universally solvable for p robots (i.e., $(G(V, E), p)$ is a YES-instance), otherwise it outputs that G is not universally solvable for p robots (i.e., $(G(V, E), p)$ is a NO-instance).

For a YES-instance $(G(V,E),p)$, all configurations $S \in \mathcal{S}_G$ are reachable from S_I. Therefore, when Algorithm 2 runs on a YES-instance, for any uniformly sampled configuration S_R, the feasibility subroutine Algorithm $\mathcal{A}$ will always return that S_R is reachable from S_I, hence Algorithm 2 would return YES with probability 1.

For a NO-instance, the number of distinct configurations that are reachable from S_I is at most $(|\mathcal{S}_G|/2)$ (Lemma 4.2). Algorithm 2 outputs YES for a NO-instance if Algorithm $\mathcal{A}$ outputs that S_R (the uniformly sampled configuration) is reachable from S_I. Therefore, the probability that a uniformly sampled configuration S_R is reachable from S_I is equal to the ratio of the number of configurations in $\mathcal{S}_G$ reachable from S_I to the total number of configurations in $\mathcal{S}_G$.

$$\Pr(\text{Algorithm 2 outputs YES for a NO instance}) \leq \frac{|\mathcal{S}_G|/2}{|\mathcal{S}_G|} \leq 1/2$$

Algorithm 2 **Input:** $G(V,E)$, p = number of robots

1: $n_{cc} \leftarrow$ Number of connected components of G (using a DFS)
2: **if** $n_{cc} > 1$ **then**
3: **return** NO
4: **end if**
5: Define the Identity Configuration S_I as $S_I(i) = v_i$ for all $i \in [p]$
6: Sample uniform random configuration S_R from the set of all configurations
7: **return** output of Algorithm $\mathcal{A}$ on input $(G(V,E), p, (S_I, S_R))$ ▷ Proposition 2.1

We now analyse the time complexity of Algorithm 2. Sampling a random configuration S_R from the set of all possible configurations can be done in linear time, $O(|V| + |E|)$. Algorithm $\mathcal{A}$ runs in linear time, $O(|V| + |E|)$ (Proposition 2.1). Moreover, all other steps (taking input, running DFS, etc.) are at most linear time, $O(|V| + |E|)$ operations. Hence, Algorithm 2 runs in time $O(|V| + |E|)$. This completes the proof of the theorem. □

5 OPTIMIZED DETERMINISTIC ALGORITHM FOR USOLR

In this section, we explore the plausibility of determinsistic algorithms for the randomised algorithm designed in the previous section. We first derandomize the earlier randomized Algorithm 2, which allows checking universal solvability using $p-1$ carefully chosen reachability tests. Compared to the randomized version, the running time increases by a factor of p, but it yields a fully deterministic, polynomial-time algorithm. The detailed results are given in the full version [7]. We provide the pseudocode in Algorithm 3.

Theorem 5.1 (⋆). *Given a graph $G = (V,E)$ and p robots, there exists a deterministic algorithm solving USolR, with running time $O(p \cdot (|V| + |E|))$.*

We now provide an optimised version of the algorithm from Theorem 5.1, that exploits the structural properties of the input graph.

We start of by recalling Theorem 5 in the paper due to Yu and Rus [24].

Algorithm 3 **Input:** $G(V,E)$, p = number of robots

1: $n_{cc} \leftarrow$ Number of connected components of G ▷ Using DFS
2: **if** $n_{cc} > 1$ **then**
3: return NO
4: **else**
5: Define the Identity Configuration S_I as $S_I(i) = v_i, \forall i \in [p]$
6: **for** $t \leftarrow 1$ to $p-1$ **do**
7:
8: Define π_t^* as $\pi_t^*(i) = \begin{cases} i & \text{if } i \neq t \text{ and } i \neq t+1 \\ t+1 & \text{if } i = t \\ t & \text{if } i = t+1 \end{cases}$
10: $b \leftarrow$ output of Algorithm $\mathcal{A}$ on $(G(V,E), p, (S_I, S_{\pi_t^*}))$
11: **if** b = NO **then**
12: return NO
13: **end if**
14: **end for**
15: return YES
16: **end if**

Proposition 5.1. *Let G be a 2-connected graph that is not a cycle and let there be p robots. Then for all $p \leq |V|$, G is universally solvable for p robots.*

This allows us to prove the following result.

Lemma 5.1. *For a set R of p robots, a connected graph $G(V,E)$ with a 2-connected component of size at least p that is not a cycle, is universally solvable.*

Proof. Let $C \subseteq V$ be a 2-connected component of size at least p that is not a cycle. Let $C' \subset C$ be an arbitrary subset of C having size $|C'| = p$.

Let $S : R \to V$ and $T : R \to V$ be arbitrary configurations. It follows from Lemma 3.1 that there exists a configuration $S' : R \to C'$ such that $S \sim S'$. Similarly, there exists a configuration $T' : R \to C'$ such that $T \sim T'$. Further, due to Proposition 5.1, S' and T' are reachable from each other, just by using the vertices and edges in the graph induced by C; this gives $S' \sim T'$. Putting everything together, we have $S \sim S' \sim T' \sim T$. This holds true for any pair of arbitrary configurations S and T, hence G is universally solvable for p robots. □

With this, we are ready to state and prove the following lemma crucial for optimizing the deterministic algorithm.

Lemma 5.2 (⋆). *For $p \geq 2$, let $G(V,E)$ be a connected graph without any 2-connected component of size at least p which is not a cycle. Then $|E| < p \cdot |V|$.*

Proof Sketch. We decompose the connected graph G into its biconnected components and consider the tree structure formed by these components connected via cut vertices. Cycles contribute linearly to the edge count, while non-cyclic components of size less than p contribute at most quadratically in their size. By accounting for repeated counting of cut vertices across components, we bound the total size of all components by $2|V| - 2$. Combining these observations, we conclude that the total number of edges $|E|$ is strictly less than $p \cdot |V|$, as claimed. □

This, with Lemma 5.1, gives us the following observation.

Observation 5.1. *Every graph $G(V, E)$ that is not universally solvable for p robots, must satisfy $|E| < p \cdot |V|$.*

With this, we are ready to state our final result corresponding to the optimized deterministic algorithm.

Theorem 5.2 (⋆). *Let $G(V, E)$ be a graph and let p denote the number of robots. There exists a deterministic polynomial-time algorithm for the USolR problem whose running time is bounded as follows:*

$$\begin{cases} O(p \cdot (|V| + |E|)) & \text{if } |E| < p \cdot |V|, \\ O(|V| + |E|) & \text{if } |E| \geq p \cdot |V|. \end{cases}$$

Proof Sketch. Let us consider the instance $G(V, E)$, p of USolR. If G is disconnected, it is not universally solvable. Otherwise, by Observation 5.1, whenever $|E| \geq p \cdot |V|$, the input instance must be a universally solvable. Therefore, we can check if $|E| \geq p \cdot |V|$ and output YES if the check passes. Otherwise we run Algorithm 3 on the input instance to get the answer. The correctness of this algorithm is due to Observation 5.1 and the correctness of Algorithm 3. All that remains is to analyse the running time of this algorithm.

If G is disconnected, we detect in linear time, $O(|V| + |E|)$, and output NO correctly. If $|E| \geq p \cdot |V|$, we take input the graph, perform the check, and output YES correctly. This takes linear time, $O(|V| + |E|)$, in total. Otherwise, if $|E| < p \cdot |V|$, then we run Algorithm 3, which takes time $O(p \cdot (|V| + |E|))$ (Theorem 5.1). This completes the proof. □

The pseudocode of this algorithm, denoted as Algorithm 4. This algorithm performs at least as good as the algorithm in Theorem 5.1 for all graphs. Moreover, when p is small, and the graph is dense enough, it utilises Observation 5.1 to output YES correctly. Therefore, the speed up is significant for dense graphs with small number of robots.

Algorithm 4 **Input:** $G(V, E)$, p = number of robots

```
n_cc ← Number of connected components in G    ▷ Using DFS
if n_cc > 1 then
    return NO
else if |E| ≥ p · |V| then    ▷ A YES-instance, Observation 5.1
    return YES
else
    return output of Algorithm 3 run on G and p
end if
```

6 AUGMENTING GRAPHS FOR UNIVERSAL SOLVABILITY

Having designed polynomial-time algorithms for the USolR problem, we now ask a natural follow-up question: Given an instance $(G(V, E), p)$ that is not universally solvable, can we augment the graph by adding a limited number of vertices and edges to make it universally solvable for p robots? In this Section, we give combinatorial bounds on the minimum number of additional edges and/or vertices that need to be added to make a graph universally solvable for a given number of robots p.

Augmenting by a set of edges. We now formally state the edge augmentation problem.

Graph Edge Augmentation for Universal Solvability (EAUS)
Input: A connected graph $G(V, E)$, the number of robots p (such that (G, p) is not universally solvable), budget β for edge addition
Question: Decide if the graph G can be made universally solvable for the given p by adding at most β new edges to G

Note that the number of robots p for which we want G to be universally solvable is part of the input. We ideally want for an input graph universal solvability for any number of robots $p \leq |V|$ and simply solving the above problem for $p = |V|$ guarantees universal solvability for all $p \leq |V|$. This is because, for any configuration S with the number of robots $p < |V|$, we can consider a configuration S' of $|V|$ robots, with p robots occupying the same vertices as in S and $(|V| - p)$ dummy robots occupying the remaining $(|V| - p)$ empty vertices of S.

We study the EAUS problem only for connected graphs and further assume that the graph G has at least four vertices. For the case where the number of vertices $|V|$ is less than or equal to three, a simplistic brute force approach can be used to solve EAUS. On general graphs we do a case analysis for different classes of graphs based on vertex and edge connectivity in order to obtain upper bounds on the number of edge additions required to achieve universal solvability. We also rely upon certain structural results related to universal solvability discussed by Yu and Rus [24] for our upper bound analysis.

Theorem 6.1 (⋆). *The following number of additional edges β are sufficient to make a graph universally solvable.*

- *For 2-connected graphs which are not simple cycles: The graph is already universally solvable (Proposition 5.1)*
- *For simple cycles: $\beta = 1$ is sufficient.*
- *For 2-edge connected but not 2-connected graphs: $\beta = 1$ is sufficient.*
- *For 1-edge connected graphs: $\beta = p - 2$ is sufficient.*

The case of simple cycles and 2-edge connected but not 2-connected graphs has been discussed in the full version [7].

For 1-edge connected graphs, attaining universal solvability is more intricate unlike the other three cases, as here, a single edge addition is insufficient. We build a p-sized cycle with at most $p - 2$ extra edges and attach a "waiting vertex" to ensure reachability, handling the cases $p = |V|$ and $p < |V|$ while preserving connectivity. The detailed analysis of the upper bound is provided in the full version [7].

Now, we provide an asymptotically matching lower bound of $\Theta(p)$, even for the restricted class of 1-edge connected graphs. Consider the case $p = |V|$. Consider the star graph $G(V, E)$ with $|V| - 1$ leaves. Note that for $p = |V|$, a robot can move only if it is located on a cycle. Since there are $|V| - 1$ leaves, for all robots in these leaves to even be able to move, we need to at least make each leaf, a part of some cycle. This would require at least $(|V| - 1)/2 = \Theta(|V|) = \Theta(p)$ additional edges. Moreover, for any tree with $\Theta(|V|) = \Theta(p)$ many leaves, at least $\Theta(|V|) = \Theta(p)$ additional

edges would be required to make the graph universally solvable. Thus, we get a lower bound of $\beta \geq (|V|-1)/2$.

Theorem 6.2. *There is an infinite family of graphs, such that for some graph $G(V,E)$ in it, for $p = |V|$ robots, universal solvability requires an edge budget of $\beta \geq \frac{|V|-1}{2} = \frac{p-1}{2}$.*

Thus, we have both an upper-bound and a lower-bound for β for 1-edge connected graphs, thus, showing that a constant budget for edge additions does not suffice to achieve universal solvability for such graphs.

Augmenting by a set of connected vertices and edges

Adding vertices along with edges to a graph $G(V,E)$ to achieve universal solvability is another interesting augmentation variant. Looking back at the instances showing lower bound of β in 1-connected graphs, where a star graph with $p-1$ leaves required $\beta = \Theta(p) = \Theta(|V|)$ additional edges to make it universally solvable, we observe that adding two more leaves to the central vertex of the star graph makes it universally solvable (it is easy to verify that a star graph with $p+1$ leaves is universally solvable for p robots). Therefore, instead of a linear number of edge additions ($\beta = \Theta(|V|)$), addition of vertices along with edges allow us to make the graph universally solvable by addition of just 2 edges and 2 vertices. This motivates us to look into a variation of the augmentation problem in which addition of α vertices and β edges is allowed in order to achieve universal solvability.

We formally state this problem and then discuss lower and upper bounds of α and β in general graphs.

Graph Vertex and Edge Augmentation for Universal Solvability (VEAUS)
Input: A connected graph $G(V,E)$, the number of robots p (such that (G,p) is not universally solvable), α and β.
Question: Decide if the graph G can be made universally solvable for the given p by adding a set S of at most α new vertices, and at most β new edges between vertices in $V \cup S$.

Note that the upper bounds established for EAUS also hold for VEAUS if we simply set $\alpha = 0$. As discussed, the lower bound instance for Graph Edge Augmentation for Universal Solvability (i.e., a star with $p-1$ leaves) allows solvability for $\alpha = 2$, $\beta = 2$. We show a stronger lower bound for a different set of graphs.

Definition 6.1. *Define the graph $Z_{\alpha,\beta}$ for some α and β as :*

- *Z contains a vertex v, which we denote as the 'center'.*
- *$(2\beta+1)$-many paths of length $(\alpha+1)$ are attached to the vertex.*

We depict the graph $Z_{\alpha,\beta}$ in Figure 5. Therefore, the number of vertices n of $Z_{\alpha,\beta}$ is equal to $1 + (2\beta+1)\cdot(\alpha+1)$. We will show that $Z_{\alpha,\beta}$ for n robots is a NO-instance of Graph Vertex and Edge Augmentation for Universal Solvability for vertex budget α, and edge budget β.

Theorem 6.3. *$Z_{\alpha,\beta}$ is a NO-instance of* Graph Vertex and Edge Augmentation for Universal Solvability *for vertex budget α, edge budget β and $1 + (2\beta+1)\cdot(\alpha+1)$ robots.*

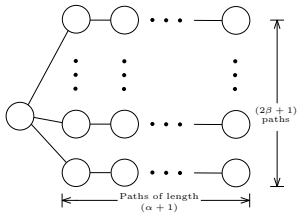

Figure 5: Schematic diagram of the graph $Z_{\alpha,\beta}$

Proof. Our proof exploits the fact that $Z_{\alpha,\beta}$ has large number of long paths. We denote by $n = 1 + (2\beta+1)\cdot(\alpha+1)$, the number of vertices of $Z_{\alpha,\beta}$

Let the paths of length $\alpha+1$ be named as $P_1, P_2, \ldots, P_{2\beta+1}$. By pigeonhole principle, there must exist at least one such P_i for which none among the β newly added edges are incident on any vertex of P_i.

We now consider the leaf vertex u along the path P_i. Consider a configuration S with robot 1 located at u, and consider another configuration T with robot 1 located at the center vertex v. We will show that T is not reachable from S.

We define depth of a vertex as the distance from the center v. Note that the first time instance when robot 1 is at a depth $h \in \{0,1,2,\ldots,\alpha+1\}$, P_i must have at least $(\alpha+1-h)$-many vacant vertices: all vertices between the current location of robot 1 and the leaf vertex u. Therefore, the first time robot 1 reaches the center (i.e., $h=0$), there must be $\alpha+1$ many empty vertices in P_i (entire P_i should be vacant). However this is a contradiction, because the augmented graph has at most $\alpha+n$ vertices and exactly n robots, making it impossible to have more than α vacant vertices. This completes the proof. □

This gives us a lower bound of $\alpha = \Omega\left(\frac{|V|}{\beta}\right)$ for a graph $G(V,E)$ to be a YES-instance, for any number $p \leq |V|$ of robots with vertex budget α and edge budget β. We note down a particularly interesting consequence of this, for $\beta = \Theta(\sqrt{|V|})$.

Corollary 6.1. *There is an infinite family of graphs, such that for some graph $G(V,E)$ in it, for $p = |V|$ robots, and edge budget $\beta = \Theta(\sqrt{|V|})$, universal solvability requires a vertex budget of $\alpha = \Omega(\sqrt{|V|})$.*

7 CONCLUSION

In this paper, we design polynomial time algorithms for USoLR. Moreover we study the augmentation problems EAUS and VEAUS in connected graphs and provide upper and lower bounds on the vertex and edge budgets in order to achieve universal solvability. A promising direction is to generalize USoLR to settings where edges have weights representing costs or distances, and where nodes or edges may have capacity limits. In such models, universal solvability would require not only reachability between all configurations but also consideration of efficiency metrics such as total movement cost or makespan.

REFERENCES

[1] Vincenzo Auletta, Angelo Monti, Mimmo Parente, and Pino Persiano. 1999. A linear-time algorithm for the feasibility of pebble motion on trees. *Algorithmica* 23, 3 (1999), 223–245.

[2] Jacopo Banfi, Nicola Basilico, and Francesco Amigoni. 2017. Intractability of Time-Optimal Multirobot Path Planning on 2D Grid Graphs with Holes. *IEEE Robotics and Automation Letters* 2, 4 (2017), 1941–1947.

[3] Édouard Bonnet, Tillmann Miltzow, and Paweł Rzążewski. 2018. Complexity of token swapping and its variants. *Algorithmica* 80 (2018), 2656–2682.

[4] Soon-Jo Chung, Aditya Avinash Paranjape, Philip Dames, Shaojie Shen, and Vijay Kumar. 2018. A survey on aerial swarm robotics. *IEEE Transactions on Robotics* 34, 4 (2018), 837–855.

[5] Argyrios Deligkas, Eduard Eiben, Robert Ganian, Iyad Kanj, and M. S. Ramanujan. 2024. Parameterized Algorithms for Coordinated Motion Planning: Minimizing Energy. In *51st International Colloquium on Automata, Languages, and Programming (ICALP 2024) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 297)*. 53:1–53:18.

[6] Erik D Demaine, Sándor P Fekete, Phillip Keldenich, Henk Meijer, and Christian Scheffer. 2019. Coordinated motion planning: Reconfiguring a swarm of labeled robots with bounded stretch. *SIAM J. Comput.* 48, 6 (2019), 1727–1762.

[7] Anubhav Dhar, Pranav Nyati, Tanishq Prasad, Ashlesha Hota, and Sudeshna Kolay. 2026. Universal Solvability for Robot Motion Planning on Graphs. arXiv:2506.18755 [cs.CC] https://arxiv.org/abs/2506.18755

[8] Eduard Eiben, Robert Ganian, and Iyad Kanj. 2023. The Parameterized Complexity of Coordinated Motion Planning. In *39th International Symposium on Computational Geometry (SoCG 2023) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 258)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 28:1–28:16.

[9] G. Franchino, G. Buttazzo, and T. Facchinetti. 2005. A distributed architecture for mobile robots coordination. In *2005 IEEE Conference on Emerging Technologies and Factory Automation*, Vol. 2. 8 pp.–156.

[10] Shinya Fujita, Tomoki Nakamigawa, and Tadashi Sakuma. 2015. Pebble exchange on graphs. *Discrete Applied Mathematics* 184 (2015), 139–145.

[11] Tzvika Geft and Dan Halperin. 2022. Refined Hardness of Distance-Optimal Multi-Agent Path Finding. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems* (Virtual Event, New Zealand) *(AAMAS '22)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 481–488.

[12] Gilad Goraly and Refael Hassin. 2010. Multi-color pebble motion on graphs. *Algorithmica* 58 (2010), 610–636.

[13] Zool Hilmi Ismail, Nohaidda Sariff, and E Gorrostieta Hurtado. 2018. A survey and analysis of cooperative multi-agent robot systems: challenges and directions. *Applications of Mobile Robots* 5 (2018), 8–14.

[14] Hadi Jahanshahi and Naeimeh Najafizadeh Sari. 2018. Robot path planning algorithms: a review of theory and experiment. (2018).

[15] Tatsuoki Kato, Tomoki Nakamigawa, and Tadashi Sakuma. 2021. Pebble exchange group of graphs. *European Journal of Combinatorics* 95 (2021), 103325.

[16] D. Kornhauser, G. Miller, and P. Spirakis. 1984. Coordinating Pebble Motion On Graphs, The Diameter Of Permutation Groups, And Applications. In *25th Annual Symposium onFoundations of Computer Science, 1984.* 241–250. https://doi.org/10.1109/SFCS.1984.715921

[17] Sam Loyd. 1959. *Mathematical puzzles*. Vol. 1. Courier Corporation.

[18] Ellips Masehian and Azadeh H Nejad. 2009. Solvability of multi robot motion planning problems on trees. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 5936–5941.

[19] Hongwei Qin, Shiliang Shao, Ting Wang, Xiaotian Yu, Yi Jiang, and Zonghan Cao. 2023. Review of autonomous path planning algorithms for mobile robots. *Drones* 7, 3 (2023), 211.

[20] Pavel Surynek. 2009. An application of pebble motion on graphs to abstract multi-robot path planning. In *2009 21st IEEE International Conference on Tools with Artificial Intelligence*. IEEE, 151–158.

[21] Richard M Wilson. 1974. Graph puzzles, homotopy, and the alternating group. *Journal of Combinatorial Theory, Series B* 16, 1 (1974), 86–96.

[22] Katsuhisa Yamanaka, Erik D Demaine, Takehiro Ito, Jun Kawahara, Masashi Kiyomi, Yoshio Okamoto, Toshiki Saitoh, Akira Suzuki, Kei Uchizawa, and Takeaki Uno. 2015. Swapping labeled tokens on graphs. *Theoretical Computer Science* 586 (2015), 81–94.

[23] Jingjin Yu. 2016. Intractability of Optimal Multirobot Path Planning on Planar Graphs. *IEEE Robotics and Automation Letters* 1, 1 (2016), 33–40.

[24] Jingjin Yu and Daniela Rus. 2014. Pebble Motion on Graphs with Rotations: Efficient Feasibility Tests and Planning Algorithms. arXiv:1205.5263 [cs.DS]