Pebble Motion on Graphs with Rotations: Efficient Feasibility Tests and Planning Algorithms

Jingjin Yu and Daniela Rus

Abstract We study the problem of planning paths for p distinguishable pebbles (robots) residing on the vertices of an n-vertex connected graph with $p \leq n$. A pebble may move from a vertex to an adjacent one in a time step provided that it does not collide with other pebbles. When $p = n$, the only collision free moves are synchronous rotations of pebbles on disjoint cycles of the graph. We show that the feasibility of such problems is intrinsically determined by the diameter of a (unique) permutation group induced by the underlying graph. Roughly speaking, the diameter of a group $\mathbf{G}$ is the minimum length of the generator product required to reach an arbitrary element of $\mathbf{G}$ from the identity element. Through bounding the diameter of this associated permutation group, which assumes a maximum value of $O(n^2)$, we establish a linear time algorithm for deciding the feasibility of such problems and an $O(n^3)$ algorithm for planning complete paths.

1 Introduction

In Sam Loyd's 15-puzzle [10], a player arranges square blocks labeled 1–15, scrambled on a 4×4 board, to achieve a shuffled row major ordering of the blocks using one empty swap cell (see, e.g., Fig. 1). Generalizing the grid-based board to an arbitrary connected graph over n vertices, the 15-puzzle becomes the problem of *pebble motion on graphs* (PMG). Here, up to $n - 1$ uniquely labeled pebbles on the vertices of the graph must be moved to some desired goal configuration, using unoccupied (empty) vertices as swap spaces. Since the initial work by Kornhauser et al. [8], PMG and its optimal variants has received significant attention in robotics [13, 18, 19] and artificial intelligence [9, 14], among others. The connection between PMG and multi-robot path planning is immediately clear, with potential applications towards

J. Yu (✉) · D. Rus
Computer Science and Artificial Intelligence Lab, Massachusetts Institute of Technology, Cambridge, MA, USA
e-mail: jingjin@csail.mit.edu

D. Rus
e-mail: rus@csail.mit.edu

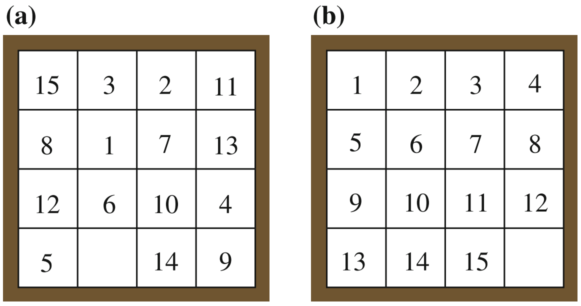


Fig. 1 Two 15-puzzle instances. **a** An unsolved instance. In the next step, one of the blocks 5, 6, 14 may move to the vacant cell, leaving behind it another vacant cell for the next move. **b** The solved instance

micro-fluidics [7], multi-robot path planning [13], and modular robot reconfiguration [12], to name a few.

As early as 1879, Story [15] observed that the parity of a 15-puzzle instance decides whether it is feasible. Wilson [20] generalized this observation by showing that, for 2-connected graphs (other than cycles and one special graph) with n vertices and $n-1$ pebbles, the reachable configurations form an alternating (resp. symmetric) group on $n-1$ letters when the graph is bipartite (resp. non-bipartite). An associated planning algorithm was also provided. Kornhauser et al. [8] improved the potentially exponential time algorithm from [20] by giving an algorithm for PMG that runs in $O(n^3)$ time. Auletta et al. [1] showed that deciding feasibility for PMG requires linear time when the graph is a tree. Recently, the linear-time feasibility result was extended to general graphs [6, 21]. Although not a focus of this paper, we note that computing optimal plans for such problems is generally NP-complete [5, 11, 16, 22].

As evident from the techniques used in [8, 20], pebble motion problems are closely related to the structure of *permutation groups*. Fixing a graph and the number of pebbles, and viewing the pebble moving operations as *generators*, all configurations reachable from an initial configuration form a group that is isomorphic to a subgroup of $\mathbf{S_n}$, the symmetric group on n letters. Deciding whether a problem instance is feasible is then equivalent to deciding whether the final configuration is reachable from the initial configuration via generator products [15, 20]. Another interesting problem in this domain is the study of the *diameter* of such groups, which is the length of the longest minimal generator product required to reach a group element. Driscoll and Furst [3, 4] showed that any group represented by generators that are cycles of bounded degree has a diameter of $O(n^2)$ and such a generator sequence is efficiently computable. For generators of unbounded size, Babai et al. [2] proved that if one of the generators fixes at least 67 % of the domain, then the resulting group has a polynomial diameter. In contrast, groups with super polynomial diameters exist [3].

Somewhat surprisingly, a natural generalization of PMG allowing rotations of the pebbles without empty swap vertices has not received much attention, possibly due to its difficulty. As an example, in Fig. 2a, the pebbles labeled 3, 4, and 5 are allowed to rotate clockwise along the (only) triangle to achieve the configuration in Fig. 2b. We call this generalization the problem of *pebble motion with rotations* (PMR), a formal definition of which will follow shortly. Synchronous rotations are important to have

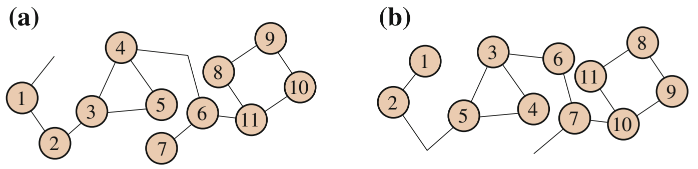


Fig. 2 Two configurations that can be turned into each other in a single synchronized move

in a multi-robot setting for at least two reasons. First, with communication, robots are able to execute synchronous rotational moves easily. Disabling such moves thus wastes robots' capabilities. Second, allowing rotational moves could allow more problem instances to be solved and could also significantly reduce the length of plans (note that the length of a plan can never be increased by adding more modes of motion).

In this paper, we employ a group theoretic approach to derive a linear time algorithm for testing the feasibility of a given PMR instance. The algorithm also implies a cubic time algorithm for computing full plans when a PMR instance is feasible. Thus, we establish that PMR induces similar algorithmic complexity as PMG does in the sense that planning and feasibility test take $O(n^3)$ and linear time, respectively. Nevertheless, the algorithms for solving PMG and PMR have significant differences due to the introduction of synchronous pebble rotations. By delivering these algorithms for PMR, we also bring forth the contribution of providing a now fairly complete landscape over graph-based multi-robot path planning problems.

We formally define PMG and PMR problems in Sect. 2. In Sect. 3, we look at the groups generated by cyclic rotations of labeled pebbles, on graphs fully occupied by pebbles. We show that such groups have $O(n^2)$ diameters. With this intermediate result, we continue to show, in Sect. 4, that the feasibility test of the PMR problem can be performed in $O(|V|+|E|)$ time, which implies an $O(n^3)$ algorithm for computing a feasible solution (the set of movements). We conclude the paper in Sect. 5.[1]

2 Pebble Motion Problems

Let $G = (V, E)$ be a connected undirected graph with $|V| = n$. Let there be a set $p \leq n$ pebbles, numbered $1, \ldots, p$, residing on distinct vertices of G. A *configuration* of these pebbles is a sequence $S = \langle s_1, \ldots, s_p \rangle$, in which s_i denotes the vertex occupied by pebble i. A configuration can also be viewed as a bijective map $S : \{1, \ldots, p\} \rightarrow V(S)$ in which $V(S)$ denotes the set of occupied vertices by S. We allow two types of *moves* of pebbles. In a *simple move*, a pebble may move to

[1]See http://people.csail.mit.edu/jingjin/files/YuRus15STAR.pdf for non-essential proofs and other details that were omitted.

an adjacent empty vertex. In a *rotation*, pebbles occupying all vertices of a cycle can rotate simultaneously (clockwise or counterclockwise) such that each pebble moves to the vertex previously occupied by its (clockwise or counterclockwise) neighbor. Two configurations S and S' are *connected* if there exists a sequence of moves that takes S to S'. Let S and D be two pebble configurations on a given graph G, the problem of *pebble motion on graphs* is defined as follows.

Problem 1 (*Pebble Motion on Graphs (PMG)*) Given (G, S, D), find a sequence of simple moves that take S to D.

When G is a tree, PMG is also referred to as *pebble motion on trees* (PMT). In this case, an instance is usually written as $I = (T, S, D)$ with T being a tree. When both simple moves and rotations are allowed, the resulting variant is the problem of *pebble motion with rotations*.

Problem 2 (*Pebble Motion with Rotation (PMR)*) Given (G, S, D), find a sequence of simple moves and rotations that takes S to D.

If G is a tree, then a PMR is simply a PMT. We note that it may be possible to achieve additional efficiency by allowing multiple simple moves and rotations (along disjoint cycles) to take place concurrently. For example, the configuration in Fig. 2a can be taken to the configuration in Fig. 2b in a single concurrent move. A full discussion of such moves (i.e., the optimality perspective) is beyond the scope of this paper.

3 Graph Induced Group and the Upper Bound on Its Diameter

3.1 Groups Generated by Cyclic Pebble Motions and Their Diameters

A particularly important case of PMR is when $p = n$; we restrict our discussion to this case in this section. When $p = n$, only synchronous rotations are possible. Given two configurations S and S' that are connected, they induce a permutation of the pebbles, which is computable via $\sigma_{S,S'}(i) = S^{-1}(S'(i))$ for each pebble i; $\sigma_{S,S}$ is the identity element. Given an initial configuration S_0, let $\mathscr{S}$ denote the set of all configurations reachable from S_0. It can be verified, using basic definitions of groups, that the permutations σ_{S_0,S_i} over all $S_i \in \mathscr{S}$ form a subgroup of $\mathbf{S_n}$, the symmetric group on n letters. Since this group is determined by the graph G, we denote it $\mathbf{G}$.

Two cycles of G are *disjoint* if their vertex sets have empty intersection. When $p = n$, each synchronous move corresponds to the rotations of pebbles along a set of of disjoint cycles. Let $\mathscr{C}$ be the collection of all sets of disjoint cycles in G; each $C \in \mathscr{C}$ is a unique set of disjoint cycles of G. Since the pebbles may rotate clockwise or counterclockwise along a cycle $c_i \in C$, each set of disjoint cycles C

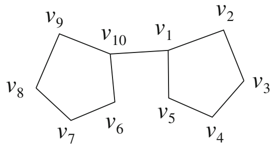


Fig. 3 For the graph above, the collection of sets of cycles are $\mathscr{C} = \{\{v_1v_2v_3v_4v_5\}, \{v_6v_7v_8v_9v_{10}\}, \{v_1v_2v_3v_4v_5, v_6v_7v_8v_9v_{10}\}\}$

can take a configuration to $2^{|C|}$ new configurations with one move. That is, each C yields $2^{|C|}$ generators of $\mathbf{G}$. Let the set of all generators obtained this way be $\mathscr{G}$. As an example, the graph in Fig. 3 has two cycles, with $|\mathscr{C}| = 3$ and $|\mathscr{G}| = 8$ (note that $|\mathscr{G}| = 2^{|\mathscr{C}|}$ does not hold in general). We make the simple observation that these definitions yield a natural bijection between synchronous moves and elements of $\mathscr{G}$. As such, when a configuration S' is reachable from a configuration S, we say that the permutation $\sigma_{S,S'} \in \mathbf{G}$ is *reachable* (from the identity) using products of generators from $\mathscr{G}$ corresponding to the synchronous moves. We frequently invoke this bijection between synchronous moves and generators without explicitly stating so. Lastly, any element $x \in \mathbf{G}$ can be expressed as generator product $g_1g_2 \dots g_k$ in which $g_1, \dots, g_k \in \mathscr{G}$. Let k_x be the minimum k such that $x = g_1g_2 \dots g_k$. The diameter of $\mathbf{G}$, $diam(\mathbf{G})$, is defined as the maximum k_x over all $x \in \mathbf{G}$.

3.2 Upper Bound over Group Diameters

The main result to be established in this section is $diam(\mathbf{G}) = O(n^2)$. To show this, G is divided into classes based on its connectivity. When G is connected (1-connected) but none of its subgraphs are 2-connected (i.e., G has no cycles), it is a tree. In this case, no pebble can move. Another simple case is when G is a cycle, the simplest 2-connected graph. Then, it is clear that all elements of $\mathbf{G}$ are generated by a single rotation.

Lemma 1 (Trees and Cycles) *If G is a tree, then $\mathbf{G} \cong \{1\}$, the trivial group. If G is a cycle, then $\mathbf{G} \cong \mathbb{Z}/n$, the cyclic group of order n.*

When G is connected but the removal of some vertex from G leaves two or more components, it is *separable*. An important case here is when G is a set of cycles sharing vertices so that no edge of G is on more than one cycle. Such graphs form a subset of 2-edge-connected graphs. Figure 4 gives an example with two cycles. Following convention, $\mathbf{A_n}$ denotes the *alternating group* on n letters. For groups, $\mathbf{G}_1 \geq \mathbf{G}_2$ or $\mathbf{G}_2 \leq \mathbf{G}_1$ denotes that $\mathbf{G}_2$ is a subgroup of $\mathbf{G}_1$. For two configurations S and S' over the same set of pebbles on the same graph, we say that they are *cycle similar* if the following property holds. For any pebble a, let the sets of cycles (of the underlying graph G) occupied by a in configurations S and S' be C_S and $C_{S'}$, respectively. Then $C_S \cap C_{S'} \neq \varnothing$.

A key result of this section is the following.

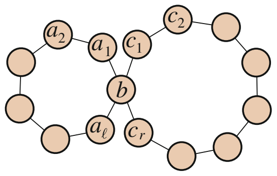


Fig. 4 Two cycles sharing one common vertex. The graph is *separable* at b

Theorem 1 (Cycles, Separable) *If every edge of a separable graph G is on exactly one cycle, then $\mathbf{G} \geq \mathbf{A_n}$ and $diam(\mathbf{G}) = O(n^2)$.*

Proof Given configurations S and D, we claim:

1. In $O(n^2)$ moves, D can be taken to some configuration D' such that S and D' are cycle similar. As an example, in Fig. 4, assuming the given configuration is S, this step ensures that in configuration D', pebbles a_i's are all on the left cycle and pebbles c_i's are all on the right cycle. The pebble b may appear on either one of the two cycles.
2. In $O(n^2)$ moves from D', a configuration D'' can be reached such that either $D'' = S$ or D'' and S differ by a transposition (group action). We require that the transposition is fixed for a fixed S and involves two adjacent pebbles of S. Let S' be the result of letting this transposition act on S.

These claims are proved in lemmas that follow. By these claims, an arbitrary D can reach either S or S'. Therefore, all configurations (and consequently elements of $\mathbf{S_n}$) are partitioned into two equivalence classes based on mutual reachability. Since the only subgroup of $\mathbf{S_n}$ of index 2 is $\mathbf{A_n}$, this implies that $\mathbf{G} \geq \mathbf{A_n}$.

When $\mathbf{G} \cong \mathbf{A_n}$, any element of $\mathbf{G}$ is a product of generators from $\mathscr{G}$ with a length of $O(n^2)$, proving $diam(\mathbf{G}) = O(n^2)$. If $\mathbf{G}$ is not isomorphic to $\mathbf{A_n}$, since the only subgroups of $\mathbf{S_n}$ containing $\mathbf{A_n}$ are $\mathbf{A_n}$ and $\mathbf{S_n}$ itself, $\mathbf{G} \cong \mathbf{S_n}$. This implies that $\mathbf{A_n}$ has at most two cosets in $\mathbf{G}$; denote the other coset of $\mathbf{A_n}$ as $\mathbf{A_n}^c$, which also have a diameter of $O(n^2)$ (to see this, note that any configuration D is reachable from one of S, S' in $O(n^2)$ moves). From the identity, all elements of $\mathbf{A_n}$ are reachable using generator products of length $O(n^2)$. Since elements of $\mathbf{A_n}^c$ are now reachable from elements of $\mathbf{A_n}$, an element of $\mathbf{A_n}^c$ must be reachable from the identity using a generator product of length $O(n^2)$ as well. Therefore, when $\mathbf{G} \cong \mathbf{S_n}$, all elements of $\mathbf{G}$ are reachable using generator products of length $O(n^2)$, yielding $diam(\mathbf{G}) = O(n^2)$. □

Before moving to the lemmas, we note that when G is separable and every edge of G is on exactly one cycle, the edges of G can be partitioned into equivalence classes based on the cycles they belong to. Because G is separable, every cycle must border one or more cycles and at the same time, two cycles can share at most one vertex. Such a graph is also called a *cactus* graph. Moreover, there exists a cycle that only shares one vertex with other cycles. We call such a cycle a *leaf cycle*. An example of a leaf cycle is given in Fig. 5.

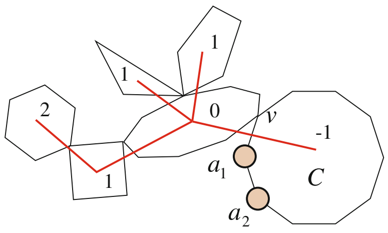


Fig. 5 The dual tree structure in a separable graph G with every edge on exactly one cycle. The numbers represent distances of the cycles to the leaf cycle C, which in fact is the root of the tree

Given a cycle C' on G, it is of *cycle distance* d_c to C if a vertex on C' needs to travel through at least d_c cycles to reach C. A neighboring cycle of C has distance 0 since they share a common vertex. Let C have a cycle distance of -1 by definition. This induces a (dual) tree structure on the cycles when viewing them as vertices joined by edges to neighbors (see, e.g., Fig. 5). Computing such a tree takes time $O(|V|+|E|)$ because obtaining maximal 2-connected components takes linear time [17]. The first claim in the proof of Theorem 1 can be stated as follows.

Lemma 2 (Initial Arrangement) *Given a separable G with each edge on exactly one cycle and configurations S and D, in $O(n^2)$ moves, a configuration that is cycle similar to S is reachable from D.*

Proof Note that a pebble may reside on multiple cycles; this lemma only ensures that each pebble gets moved to one of the cycles it belongs to in S. First we show that a single pebble can be relocated to a cycle it belongs to in S in $O(n)$ rotations, without affecting pebbles that are previously arranged. When G is two cycles joined on a common vertex (e.g., Fig. 4), without loss of generality, assume that we need to move a_i from the left cycle to the right cycle. This implies that some pebble c_j (and possibly b) does not belong to the right cycle in S. We note that the group $\mathbf{G}$ in this case has four generators, $g_\ell = \begin{pmatrix} a_1 & a_2 & \dots & a_\ell & b \\ b & a_1 & \dots & a_{\ell-1} & a_\ell \end{pmatrix}$, $g_r = \begin{pmatrix} c_1 & c_2 & \dots & c_r & b \\ c_2 & c_3 & \dots & b & c_1 \end{pmatrix}$, which correspond to clockwise rotations along the left and right cycles, respectively, and their inverses, g_ℓ^{-1} and g_r^{-1}. One can verify that the generator product $g_\ell^{-i} g_r^{-j} g_\ell^{i}$ exchanges a_i and c_j between the two cycles without affecting the cycle membership of other pebbles (see Fig. 6). For the general case in which a pebble needs to go through some k cycles, denoting the generators as $g_1, \dots, g_k$, it is easy to verify that a product of the form $g_1^{-i_1} g_2^{-i_2} \dots g_k^{i_k} \dots g_2^{i_2} g_1^{i_1}$ achieves what we need, with $i_1 + \dots + i_k < n$. There may be more than these $2k$ basic generators, but we do not need the other generators for this proof. Therefore, at most $2n$ moves are needed to move one pebble to the

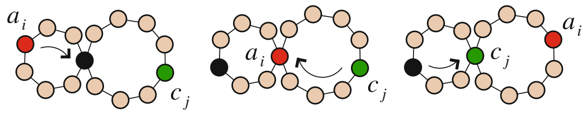


Fig. 6 Illustration of the vertex arrange algorithm for two adjacent cycles

desired cycle. To avoid affecting pebbles that are previously arranged, we may simply fix a leaf cycle C and start with cycles based on their cycle distance to C in decreasing order. At most $2n^2$ moves are required to arrange all n pebbles to the desired cycles. □

Lemma 3 (Rearrangement) *The pebbles arranged according to Lemma 2 can be rearranged such that the resulting configuration is the same as S or differ from S by a fixed transposition of two neighboring pebbles in S. Rearrangement requires $O(n^2)$ moves.*

Proof For a fixed G, let C be a leaf cycle and let C border other cycle(s) via vertex v. In S, let a_1 be the pebble occupying counterclockwise neighboring vertex of v on the cycle C, and let a_2 be the counterclockwise neighbor of a_1 on C (again, see Fig. 5 for an illustration of this setup). The fixed transposition will be $(a_1\, a_2)$.

We rearrange pebbles to match the configuration S starting from cycles with higher cycle distances to the leaf cycle C, using the neighboring cycle with smaller cycle distance (such a cycle is unique). We show that the pebbles on the more distant cycle can always be rearranged to occupy the vertex specified by S. Moreover, this can be achieved using moves that only affect the ordering of two pebbles on the neighboring cycle. Without loss of generality, we use the two cycle example from Fig. 4 and let the right cycle be the more distant one. The generators g_ℓ, g_ℓ^{-1}, g_r, and g_r^{-1} from previous lemma remain the same. To exchange two pebbles on the right cycle, for example c_i, c_j, we may use the following generator product

$$g_\ell^{-2} g_r^{-i} g_\ell g_r^{j-i} g_\ell^{-1} g_r^{-j+i} g_\ell g_r^{-i} g_\ell. \tag{1}$$

Performing such exchanges iteratively, within $2n^2$ moves, all pebbles except those on the leaf cycle C can be rearranged to occupy vertices specified by S. Reversing the process, we can arrange all pebbles on C to occupy vertices specified by S, using a neighboring cycle C', affecting the ordering of at most two pebbles on C'. Repeating this process again with C' using C as the neighboring cycle and a_1, a_2 as the swapping pebbles, all pebbles except possibly a_1, a_2 occupy the vertices specified by S. □

The above two lemmas complete the proof of Theorem 1. At this point, it is easy to see that when G is separable with each edge on a single cycle, $\mathbf{G} \cong \mathbf{S_n}$ if and only if G contains an even cycle, corresponding to the composition of an odd number of transpositions. Otherwise, $\mathbf{G} \cong \mathbf{A_n}$. We are left with the case in which G is 2-connected but not a (single) cycle.

Theorem 2 (2-connected, General) *If G is 2-connected and not a cycle,* $\mathbf{G} \cong \mathbf{S_n}$ *with* $diam(\mathbf{G}) = O(n^2)$.

Combining Theorems 1 and 2 concludes the case for 2-edge-connected graphs that are not single cycles; the case of general graph then follows. Since we will mention "2-edge-connected component" fairly frequently, we abbreviate it to "TECC" except in theorem statements. Also, we call each component of G after deleting all TECCs a *branch*.

Proposition 1 (2-edge-connected) *If G is 2-edge-connected and not a single cycle,* $\mathbf{G} \geq \mathbf{A_n}$ *with* $diam(\mathbf{G}) = O(n^2)$.

Proof A 2-edge-connected graph G can be separated into 2-connected components via splitting at articulating vertices. A (dual) tree structure, similar to that illustrated in Fig. 5, can be built over these components. The two-step algorithm used in the proof of Theorem 1, in combination with Theorem 2, can be applied to show that $\mathbf{G} \geq \mathbf{A_n}$ and $diam(\mathbf{G}) = O(n^2)$. □

After gathering all cases, we obtain the following main result for this section.

Theorem 3 (General Graph) *Given an arbitrary connected, undirected, simple graph G,* $diam(\mathbf{G}) = O(n^2)$.

Proof Pebbles on vertices of G that are not on any cycle are always immobile. Deleting those vertices does not change $\mathbf{G}$. After all such vertices are removed, we are left with the TECCs of G. Denoting the associated groups of these components $\{\mathbf{G}_i\}$, $\mathbf{G}$ is the direct product of the $\mathbf{G}_i$'s. Since all $\mathbf{G}_i$'s have $O(n^2)$ diameter, so does $\mathbf{G}$. □

4 Linear Time Feasibility Test of PMR

We now describe a linear time algorithm for testing the feasibility for PMR, using a proof strategy similar to that from [1] on PMT. We first restate a result form [1].

Theorem 4 (Theorem 3 in [1]) *Given an instance (T, S, D) of PMT, in $O(n)$ steps, an instance (T, S', D) of PMT can be computed such that S', D contain the same set of vertices and (T, S, S') is feasible.*

The following corollary is also obvious.

Corollary 1 *Given an instance (T, S, D) of PMR, let (T, S', D) be the new instance obtained according to Theorem 4. Then (T, S, D) is feasible if and only if (T, S', D) is feasible.*

By Theorem 4 and Corollary 1, reconfiguration can be performed on a PMR instance $I = (G, S, D)$ to get an equivalent instance $I' = (G, S', D)$ so that S', D have the same underlying vertex set (i.e., $V(S') = V(D)$). To do this, find a spanning tree T of G. The $O(n)$ time algorithm guaranteed by Theorem 4 can then compute a desired instance (T, S', D) with S', D having the same set of vertices. Since the moves taking (T, S, S') is feasible, (G, S, S') is feasible; therefore, (G, S, D) is feasible if and only if (G, S', D) is feasible. Given an instance $I = (G, S, D)$ in which S and D have the same underlying set, we call it the *pebble permutation with rotation* problem or PPR. Given a PPR instance, we say that two pebbles are *equivalent* if they can exchange locations with no net effect on the locations of other pebbles. A set of pebbles are equivalent if every pair of pebbles from the set are equivalent.

In testing the feasibility of a PPR instance $I = (G, S, D)$, a simple but special case is when G is a cycle. In this case, S and D induce natural cyclic orderings of the pebbles. The following is then clear.

Lemma 4 *Let $I = (G, S, D)$ be an instance of PPR in which G is a cycle. Then I is feasible if and only if $s_i = d_{(i+k) \bmod p}$ for some fixed natural number k.*

When G is not a cycle, the feasibility test is partitioned into four main cases, depending on the number of pebbles, p, with respect to the number of vertices of G. It is assumed that G contains at least one TECC since otherwise G is a tree and the problem is a PMT problem.

4.1 Feasibility Test of PPR When $p = n$

When $p = n$, all vertices are occupied by pebbles. Clearly, if a pebble is on a vertex that does not belong to any cycle (i.e., a branch vertex), the pebble cannot move. Therefore, $I = (G, S, D)$ is feasible only if for every branch vertex $v \in V(G)$, $S^{-1}(v) = D^{-1}(v)$. Furthermore, given any TECC C of G, $S^{-1}(C) = D^{-1}(C)$ must also hold, since pebbles cannot move out a TECC. If these conditions hold, the feasibility of I is reduced to feasibilities of $\{(C_i, S|_{S^{-1}(C_i)}, D|_{D^{-1}(C_i)})\}$, in which C_i's are the TECCs of G and $S|_{S^{-1}(C_i)}$ denotes S restricted to the domain $S^{-1}(C_i)$; same applies to $D|_{D^{-1}(C_i)}$. More formally,

Proposition 2 *Let $I = (G, S, D)$ be an instance of PPR with $p = n$. Let $\{C_i\}$ be the set of 2-edge-connected components of G. Then I is feasible if and only if the following holds: 1. for all $v \in V(G\backslash(\cup_i C_i))$, $S^{-1}(v) = D^{-1}(v)$, 2. for each C_i, $S^{-1}(C_i) = D^{-1}(C_i)$, and 3. for each C_i, the PPR instance $(C_i, S|_{S^{-1}(C_i)}, D|_{D^{-1}(C_i)})$ is feasible. Moreover, the feasibility test can be performed in linear time.*

Proof Finding TECCs of G can be done in $O(|V|+|E|)$ time [17]. Checking whether condition 1 holds takes linear time. For checking condition 2, for each C_i, we first gather $S^{-1}(C_i)$ and for each pebble in $S^{-1}(C_i)$, mark the pebble as belonging to C_i. We can then check whether the pebbles in $D^{-1}(C_i)$ also belong to C_i in linear time. For condition 3, deciding the feasibility of $(C_i, S|_{S^{-1}(C_i)}, D|_{D^{-1}(C_i)})$ can be done using the results from Sect. 3. This check can performed as follows. 1. Check whether C_i is a cycle, which is true if and only if no vertex of C_i has degree more than two. If this is the case, apply Lemma 4 to test the feasibility on C_i; 2. Check whether C_i is a cactus with no even cycle. We can verify whether C_i is a cactus as follows: Using depth first search (DFS), detecting cycles of C_i. If C_i is a cactus, then it should assume a "tree" structure shown in Fig. 5; the first cycle that is found must be a leaf cycle. Deleting this cycle (without deleting the vertex that joins this cycle to the rest of C_i) from C_i yields another cactus. Repeating the process tells us whether C_i is a cactus. As we are finding the cycles, we can check whether there is an even cycle. If C_i is indeed a cactus with no even cycle, the possible configurations have two equivalence classes. The subproblem is only infeasible if $S|_{S^{-1}(C_i)}, D|_{D^{-1}(C_i)}$ fall into different equivalence classes, which can be checked by computing the parity of the permutation $\sigma_{S,D}$, restricted to C_i, in linear time; 3. For all other types of C_i, the subproblem is feasible. □

4.2 Feasibility Test of PPR When $p = n - 1$

When $p = n - 1$, nearly all PPR instances, in which G are 2-edge-connected graphs, are feasible.

Lemma 5 *Let $I = (G, S, D)$ be an instance of PPR in which G is 2-edge-connected and not a cycle. If $p < n$, then I is feasible.*

Proof By Theorems 1 and 2, $\mathbf{G} \geq \mathbf{A_n}$. That is, there are at most two equivalence classes of configurations, with configurations from different classes differ by a transposition of neighboring pebbles. Since there is at least one empty vertex, viewing that vertex as a "virtual" pebble that can be exchanged with a neighboring pebble in one move, it is then clear that the two configuration classes collapse into a single class. □

Lemma 6 *Let $I = (G, S, D)$ be an instance of PPR in which G, after deleting one (or more) degree* 1 *vertex (vertices), is a 2-edge-connected graph. If $p < n$, then I is feasible.*

Proof Note that by degree 1 vertices, we mean that these vertices have degree 1 in G. Let H be the 2-edge-connected graph after deleting all degree 1 vertices and let $v_1, \ldots, v_k$ be the degree 1 vertices. Let the neighbor of v_i in G be $v_i' \in V(H)$. Since $v \in v_1, \ldots, v_k$ has degree 1, it is attached to H via a single edge. Let H_i be the subgraph of G after deleting all vertices in $v_1, \ldots, v_k$ except v_i. Assume that v_1 is empty initially, we show next that all pebbles occupying H_1 are equivalent. That is, an arbitrary configuration of these pebbles can be achieved.

If H is cycle, the subroutine illustrated in Fig. 7 shows how an arbitrary configuration of pebbles can be achieved for a triangle H, which directly generalizes to an arbitrary sized cycle. This shows that all pebbles on H_1 fall in the same equivalence class. If H is not a cycle, we can move an arbitrary pebble j from H to v_1. Lemma 5

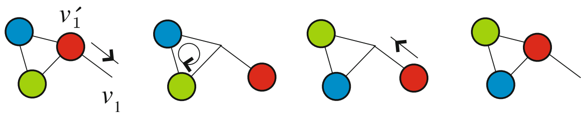


Fig. 7 With one empty vertex, pebbles on a triangle can be arranged to achieve any desired configuration. This generalizes to an arbitrary TECC

implies that all pebbles on H are equivalent. Since j is arbitrary, all pebbles on H_1 are equivalent.

Having shown that all pebbles on H_1 are equivalent, we move an arbitrary pebble j to v_1 and empty vertex v_2 (if there is a v_2). Following the same procedure, all pebbles on H_2 are equivalent. Since j is arbitrary, all pebbles on H, v_1, v_2 are equivalent. Inductively, all pebbles on G are equivalent. Therefore, an arbitrary instance I is feasible. □

When there is a single empty vertex on G, it is clear that pebbles can be moved so that the empty vertex is an arbitrary vertex of G. In particular, for any TECC H of G, we can move the pebbles so that a vertex of H is empty. By Lemma 6, all pebbles on H and its distance one neighboring vertices fall in the same equivalence class. We now show that the feasibility of the case of $p = n - 1$ can be decided in linear time.

Proposition 3 *Let $I = (G, S, D)$ be an instance of PPR in which $p = n - 1$ and G is not a cycle. The feasibility of I can be decided in linear time.*

Proof We start with pebble configuration S and group the pebbles into equivalence classes. Without loss of generality, assume that S leaves a vertex of a TECC, say H, unoccupied. By Lemma 6, all pebbles on H and its distance 1 neighbors belong to the same equivalence class, say $h_{S,1}$. Now, check whether any pebble in $h_{S,1}$ is on some other TECC $H' \neq H$. If that is the case, all pebbles on H' and its distance 1 neighbors are also equivalent and belong to $h_{S,1}$. When no more pebbles can be added to $h_{S,1}$ this way, $h_{S,1}$ is completely defined.

Let v be a vertex neighboring a vertex occupied by a pebble from $h_{S,1}$ (v itself is not occupied by a pebble in $h_{S,1}$), if v is not a TECC vertex, the pebble currently on v cannot be move to a TECC and therefore is not equivalent to any other pebble. The pebble then gets its own equivalence class, say $h_{S,2}$. If v belongs to a TECC, say H_v, then all pebbles on H_v and all H_v's distance 1 neighbors that are not yet classified belong to $h_{S,2}$; $h_{S,2}$ is then expanded similarly to $h_{S,1}$. At this point, the procedures given so far apply to partition all pebbles into equivalence classes. It is not hard to see the algorithm takes linear time to complete using breadth first or depth first search, treating each TECC as a whole. As the start configuration S is being classified, the same is done to D. In particular, if a set of pebbles of S belongs to an equivalence class $h_{S,i}$, then the pebbles of D occupying the same set of vertices get assigned to the class $h_{D,i}$. The instance I is feasible if and only if $h_{S,i} = h_{D,i}$ for all i (this can be done in linear time as we have shown in checking the second condition in Proposition 2). □

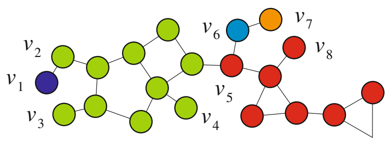


Fig. 8 An example of the case $p = n - 1$. The pebbles are put into 5 different equivalence classes, distinguished by different colors

Figure 8 provides an example of applying the above procedure to a given pebble configuration, which partitions the pebbles into 5 equivalence classes.

4.3 Feasibility Test of PPR When $p < N(TECCs)$

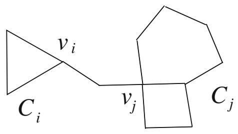
We denote by $N(TECCs)$ the number of vertices of all TECCs of G. An instance is almost always feasible when $p < N(TECCs)$.

Theorem 5 *Let $I = (G, S, D)$ be an instance of PPR in which G is not a cycle. If $p < N(TECCs)$, then I is feasible.*

Proof Since the number of pebbles are not enough to occupy all TECC vertices, we can update configuration S to a new one S' such that all pebbles are on TECC vertices. Repeating the same moves over the configuration D to get D' (i.e., if we move a pebble from v_i to v_j in the initial pebble configuration, we move the corresponding pebble from v_i to v_j in the final pebble configuration). After this process is complete, the updated start and final configurations again occupy the same set of vertices; (G, S, D) is feasible if and only if the (G, S', D') is feasible. In the rest of the proof we show that (G, S', D') is feasible.

Since not all TECC vertices are occupied in S', at least one TECC, say C_i, has an empty vertex. By Lamma 6, all pebbles on C_i are equivalent. Now let C_j be another TECC joined to C_i via a single branch (see Fig. 9 for an example). Since any pebble on C_j can be moved to vertex v_j via a proper sequence of rotations, it is then possible to exchange any pair of pebbles p_1 on C_i and p_2 on C_j: move p_2 to v_j, empty v_i, move p_2 to v_i, rotate p_1 to v_i, and move it to v_j. Via induction, any pair of pebbles on G can be exchanged, without affecting the current configuration of other pebbles. Given this procedure, we can iteratively arrange each pebble i, starting from pebble 1, by exchanging pebble i with some other pebble occupying i's vertex in D'. With up to $p - 1$ exchanges, all pebbles can be arranged to their desired final configurations. □

Fig. 9 A graph with two TECCs



4.4 Feasibility Test of PPR When $N(TECCs) \le p < n - 1$

For this last case, given a PPR instance, (G, S, D), we first move pebbles in S and D so that vertices of all TECCs are occupied. To perform this in linear time, a "fake" goal configuration D_f is created with p pebbles such that all TECCs are full occupied, in an arbitrary order. This is possible because $N(TECCs) \le p < n - 1$. Using a spanning tree T of G and apply Theorem 4 to (T, S, D_f), (T, D, D_f), we get two new instances (T, S', D_f), (T, D', D_f) with the property that S', D', and D_f all occupy the same set of vertices and (T, S, S'), (T, D, D') are both feasible. Thus, we obtain a new PPR instance (G, S', D'), which is feasible if and only if (G, S, D) is, with the additional property that vertices of all TECCs are occupied. For convenience, we call an instance (G, S, D) of PPR in which all TECC vertices are occupied a *rearranged pebble permutation* problem, or RPP. Note that this implies $p \ge N(TECCs)$.

Next, we contract G to get a *skeleton tree*, T_G, by collapsing each TECC into a *composite vertex*; other vertices and edges are left intact. For example, the graph from Fig. 8 have the skeleton tree shown in Fig. 10. This procedure induces a natural map f_T that takes any subgraph H of G to $f_T(H)$ as a subgraph of T_G (via mapping all vertices belonging to the same TECC of G to a composite vertex of T_G and non-composite vertices of G to non-composite vertices of T). Given an instance (G, S, D) of RPP with $p < n - 1$ pebbles, all pebbles on the same TECC are equivalent by Lemma 6. This induces a problem instance (T_G, S', D') in which all pebbles (in S and D) on the same TECC of G are combined into a *composite* pebble (in S' and D'). Given two vertices u and v in a graph, $u \rightsquigarrow v$ denotes a (shortest) path between u and v. Such a path is unique when the graph is a tree. By all vertices on (resp. in) $u \rightsquigarrow v$, we mean vertices of $u \rightsquigarrow v$ including (resp. excluding) u and v. Lemma 6 from [1] can be extended to RPP as follows.

Lemma 7 *Let (G, S, D) be an instance of RPP in which G is not a cycle and $N(TECCs) \le p < n - 1$. Let u, v, and w be vertices of G such that the path between u and v and the path between v and w are not edge disjoint. Assume u and v are occupied by pebbles and moves exist that take S to a new configuration in which pebble $S^{-1}(u)$ is moved to v and $S^{-1}(v)$ is moved to w. Then S can be taken to an configuration S' in which S and S' are the same except pebbles on u and v are exchanged.*

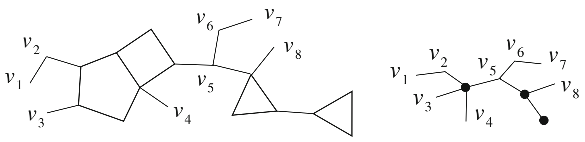


Fig. 10 The skeleton tree (on the *right*) after contracting the graph on the *left* (from Fig. 8); the *black dots* are the composite vertices

Lemma 7 leads to a generalized version of Theorem 4 from [1] to RPP, given below. We omit the proof since it is nearly identical (we need extended versions of Corollary 1 and 2 from [1], which can be easily proved in the same way Lemma 7 is proved).

Theorem 6 *An RPP instance, (G, S, D), in which G is not a cycle and $N(TECCs) \leq p < n-1$, is feasible if and only if the individual exchanges between pebble i and $S^{-1}(D(i))$, $1 \leq i \leq p$, can be performed using moves without affecting the configurations of any other pebble.*

By Theorem 6, if an instance of RPP, $I = (G, S, D)$, is feasible, then pebbles i and $\sigma_{S,D}(i) = S^{-1}(D(i))$ can be exchanged with no net effect on other pebbles. This enables a feasibility test of RPP problems (and therefore, PMR problems): vertices occupied by pebbles are partitioned into equivalence classes such that two pebbles can be exchanged if and only if the vertices occupied by them belong to the same equivalence class. In fact, we apply the $Mark$ algorithm from [1] on the skeleton tree T_G without any change at the pseudocode level (see [1] for the simple algorithm description); the main difference is how to check whether two adjacent pebbles are equivalent (Lemma 8 from [1]).

Before stating our version of the lemma, some notations are in order. We work with an arbitrary RPP instance $I = (G, S, D)$ in which G is not a cycle and $N(TECCs) \leq p < n-1$. Let $I' = (T_G, S', D')$ be the induced instance described earlier in which T_G is G's skeleton tree. A *fork* vertex of T_G is a vertex of degree at least 3 that is not a composite vertex. $F(u)$ is the set of connected components of T_G after deleting the vertex u. $T(u, v)$ is the tree of $F(u)$ containing the vertex v; $\overline{T}(u, v)$ is the rest of $F(u)$. For two vertices $u, v \in V(T_G)$, $d(u, v)$ is the length of $u \rightsquigarrow v$. In the lemmas that follow, only start configuration S' is operated on; same procedure can be applied to D. First we need a version of Corollary 3 from [1] to account for composite vertices; we omit the essentially same proof but point out that although both fork and composite vertices can help two pebbles switch locations, a composite vertex can do so with one fewer empty vertex.

Lemma 8 *Let $p_1 := S'^{-1}(u)$, $p_2 := S'^{-1}(v)$ for $u, v \in V(T_G)$ such that $u \rightsquigarrow v$ contains no other pebbles; all vertices on $u \rightsquigarrow v$ are of degree 2. Let w be a composite or fork vertex such that u is in $w \rightsquigarrow v$. The tree $T(u, w)$ has no more than $d(w, u)$ (resp. $d(w, u) + 1$) empty vertices when w is a composite (resp. fork) vertex. Let w' be the closest composite or fork vertex to v such that v is in $w' \rightsquigarrow u$ satisfying similar properties as w. Then u and v are not equivalent.*

Lemma 9 *Let $p_1 := S'^{-1}(u)$, $p_2 := S'^{-1}(v)$ for some $u, v \in V(T_G)$ such that $u \rightsquigarrow v$ contains no other pebbles. Then p_1, p_2 are equivalent with respect to S' if and only if at least one of the following conditions holds:*

1. *There exists a fork vertex w in $u \rightsquigarrow v$ such that both $T(w, u)$, $T(w, v)$ are not full or at least one other tree of $F(w)$ is not full.*
2. *Let w be a composite vertex such that u is in $w \rightsquigarrow v$ and no other fork vertex or composite vertex is in $w \rightsquigarrow u$. There exists such a w that $T(u, w)$ has $d(w, u) + 1$ empty vertices.*

3. *Symmetric to 2 with u and v switched.*
4. *Let w be a fork vertex such that u is in $w \rightsquigarrow v$ and no other fork vertex or composite vertex is in $w \rightsquigarrow u$. There exists such a w that $T(u, w)$ has $d(w, u) + 2$ empty vertices.*
5. *Symmetric to 4 with u and v switched.*
6. *Vertex u is a fork vertex. Then at least two trees of $F(u)$ has empty vertices or there are at least two empty vertices outside $T(u, v)$.*
7. *Symmetric to 6 with u and v switched.*
8. *Vertex u is a composite vertex. Then at least one tree of $\overline{T}(u, v)$ has an empty vertex.*
9. *Symmetric to 8 with u and v switched.*

Proof The proof is adopted from that of Lemma 8 from [1] with some repetitive details omitted. Since the sufficiency of the conditions can be easily checked by constructing plans that exchange p_1, p_2, only necessity is shown here via contradiction. Assume that u and v are exchangeable without configuration S satisfying any of the conditions 1–9. First consider the case in which there is no fork vertex in $u \rightsquigarrow v$ and u and v are not fork or composite vertices; these assumptions forbids conditions 1 and 6–9. If conditions 2–5 do not hold, the condition from Lemma 8 is true, thus u and v cannot be equivalent.

For the case in which no fork vertex exists in $u \rightsquigarrow v$ but u or v (possibly both) is a fork or composite vertex, the proof from Lemma 8 from [1] applies with little change to show that u and v are not equivalent unless one of conditions 2–9 holds: If conditions 2–5 do not hold, this means that p_1, p_2 must use u or v as a "hub" for switching locations; traveling beyond distance 1 from $u \rightsquigarrow v$ will not help u and v to switch. On the other hand, if conditions 6–9 do not hold, u or v cannot serve as the hub that enables u and v to switch. Furthermore, if conditions 6–9 do not hold, reconfiguration of pebbles will not make conditions 2–5, previously invalid, become valid.

This leaves the case in which conditions 2–9 do not hold, which means that u and v cannot switch on $\overline{T}(u, v)$ nor $\overline{T}(v, u)$. Since there is no pebble in $u \rightsquigarrow v$, the vertices in $u \rightsquigarrow v$ cannot be composite vertices. The same proof from Lemma 8 from [1] then shows that unless condition 1 is met, u and v cannot be equivalent. □

With Lemma 9, all criteria needed for the *Mark* algorithm from [1], in particular Observations 1–4, continue to hold on T_G without change. Since *Mark* is not changed, its running time is linear if deciding whether two adjacent pebbles are equivalent can be performed in (amortized) constant time. For this to hold, for an arbitrary tree $T(u, w)$, we need to know whether $T(u, w)$ has 0, 1, 2 holes and whether the fork or composite vertex of $T(u, w)$ closest to u allows u and another vertex v in $T(u, w)$ to exchange (i.e., $T(u, w)$ should have enough empty vertices). These data can be precomputed in $O(|V| + |E|)$ time using two depth firth traversals over the tree T_G. At this point, it is not hard to see that this linear decision algorithm easily turns into an algorithm that computes a feasible solution to a PPR instance. Our complexity analysis shows that a feasible solution can be computed in $O(|E|)$ if a high level plan is required (computes a corresponding RPP instance, checks feasibility, and outputs the permutation pairs for exchanges) and $O(n^3)$ if step by step output is required (each exchange can be done in $O(n^2)$ moves produced by a fixed formula). We summarize the main result of this section with the following theorem.

Theorem 7 *The feasibility of PMR problems can be decided in linear time. Moreover, a plan for a feasible instance can be computed in $O(n^3)$ time.*

5 Conclusion

In this paper, we proposed the problem of *pebble motion on graphs with rotations* (PMR), a graph-based multi-robot path planning problem. Our formulation takes into account natural, synchronous rotations of pebbles along fully occupied cycles of the underlying graph. The inclusion of this important case, in conjunction with previous studies of the problem that only allow pebbles to move to unoccupied vertices, paints a fairly complete picture of graph-based multi-robot path planning problems. In our systematic analysis of PMR, we show that, even for the fully constrained case in which the number of pebbles equals the number of vertices, deciding the feasibility of a PMR instance can be completed in linear time with respect to the size of the underlying graph. Moreover, computing a full plan for all moving all pebbles requires $O(n^3)$ time.

Acknowledgments This work was supported in part by NSF grant 0904501 and ONR projects N00014-12-1-1000, N00014-09-1-1051, and N00014-09-1-1052.

References

1. Auletta, V., Monti, A., Parente, M., Persiano, P.: A linear-time algorithm for the feasibility of pebble motion on trees. Algorithmica **23**, 223–245 (1999)
2. Babai, L., Beals, R., Seress, Á.: On the diameter of the symmetric group: polynomial bounds. In: Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1108–1112 (2004)
3. Driscoll J.R., Furst M.L.: On the diameter of permutation groups. In: Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing, pp. 152–160 (1983)
4. Driscoll, J.R., Furst, M.L.: Computing short generator sequences. Inf. Comput. **72**(2), 117–132 (1987)
5. Goldreich, O.: Finding the shortest move-sequence in the graph-generalized 15-puzzle is np-hard. Laboratory Computer Science Massachusetts Institute of Technology, unpublished manuscript (1984)
6. Goraly, G., Hassin, R.: Multi-color pebble motion on graph. Algorithmica **58**, 610–636 (2010)
7. Griffith, E.J., Akella, S.: Coordinating multiple droplets in planar array digital microfluidic systems. Int. J. Robot. Res. **24**(11), 933–949 (2005)
8. Kornhauser, D., Miller, G., Spirakis, P.: Coordinating pebble motion on graphs, the diameter of permutation groups, and applications. In: Proceedings of the 25th Annual Symposium on Foundations of Computer Science, pp. 241–250 (1984)

9. Krontiris, A., Luna, R., Bekris, K.E.: From feasibility tests to path planners for multi-agent pathfinding. In: Symposium on Combinatorial Search (2013)
10. Loyd, S.: Mathematical Puzzles of Sam Loyd. Dover, New York (1959)
11. Ratner, D., Warmuth, M.: The $(n^2 - 1)$-puzzle and related relocation problems. J. Symb. Comput. **10**, 111–137 (1990)
12. Reif, J.H., Slee, S.: Asymptotically optimal kinodynamic motion planning for self-reconfigurable robots. In: The Seventh International Workshop on Algorithmic Foundations of Robotics (2006)
13. Solovey, K., Halperin, D.: k-color multi-robot motion planning. In: The Tenth International Workshop on Algorithmic Foundations of Robotics (2012)
14. Standley, T., Korf R.: Complete algorithms for cooperative pathfinding problems. In: Twenty-Second International Joint Conference on Artificial Intelligence, pp. 668–673 (2011)
15. Story, E.W.: Note on the '15' puzzle. Am. J. Math. **2**, 399–404 (1879)
16. Surynek, P.: An optimization variant of multi-robot path planning is intractable. In: The Twenty-Fourth AAAI Conference on Artificial Intelligence, pp. 1261–1263 (2010)
17. Tarjan, R.E.: Depth-first search and linear graph algorithms. SIAM J. Comput. **1**(2), 140–160 (1972)
18. van den Berg, J., Snoeyink, J., Lin, M., Manocha, D.: Centralized path planning for multiple robots: optimal decoupling into sequential plans. In: Proceedings Robotics Science and Systems (2009)
19. Wagner, G., Choset. H.: M*: a complete multirobot path planning algorithm with performance bounds. In: Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3260–3267 (2011)
20. Wilson, R.M.: Graph puzzles, homotopy, and the alternating group. J. Comb. Theory (B) **16**, 86–96 (1974)
21. Yu, J.: A linear time algorithm for the feasibility of pebble motion on graphs. arXiv:1301.2342 (2013)
22. Yu, J., LaValle S.M.: Structure and intractability of optimal multi-robot path planning on graphs. In: Proceedings AAAI National Conference on Artificial Intelligence, pp. 1444–1449 (2013)