Multi-Agent Pathfinding with n Agents on Graphs with n Vertices: Combinatorial Classification and Tight Algorithmic Bounds

Klaus-Tycho Foerster†, Linus Groner‡, Torsten Hoefler‡,
Michael Koenig‡, Sascha Schmid‡, and Roger Wattenhofer‡

ktfoerster@cs.aau.dk, gronerl@ethz.ch, htor@inf.ethz.ch,
mikoenig@ethz.ch, saschmi@ethz.ch, and wattenhofer@ethz.ch

‡ETH Zurich, 8092 Zurich, Switzerland
†Aalborg University, 9220 Aalborg, Denmark

Abstract. We investigate the multi-agent pathfinding (MAPF) problem with n agents on graphs with n vertices: Each agent has a unique start and goal vertex, with the objective of moving all agents in parallel movements to their goal s.t. each vertex and each edge may only be used by one agent at a time. We give a combinatorial classification of all graphs where this problem is solvable in general, including cases where the solvability depends on the initial agent placement. Furthermore, we present an algorithm solving the MAPF problem in our setting, requiring $\mathcal{O}(n^2)$ rounds, or $\mathcal{O}(n^3)$ moves of individual agents. Complementing these results, we show that there are graphs where $\Omega(n^2)$ rounds and $\Omega(n^3)$ moves are required for any algorithm.

1 Introduction

Pathfinding for single agents on a graph is a well studied problem. *Dijkstra's algorithm* provided a solid foundation in 1959 [1] and since then, several more specialized adaptations have been conceived, such as the A^* algorithm [2] for grids and hierarchical pathfinding using the ability to pre-process maps. The applications for *multi*-agent pathfinding have grown numerous in the recent decades:

Movies such as *The Lord of the Rings* want to display huge armies clashing, but without paying an actor for each combatant [3]. Real-time strategy games incorporate larger and larger amounts of units and players expect predictable and efficient unit movement [4]. Building safety researchers can predict the movement and behaviour of human crowds during an emergency evacuation through simulation [5]. Pathfinding on graphs has also drawn attention in robotics, where it is applied to the problem of multi-robot path planning [6]. Another related field is routing in networks, where deadlock-free forwarding (pathfinding) of packets (agents) is of interest [7].

In this paper, we focus our attention on the most congested pathfinding case, where n agents are to be routed on n-vertex graphs, advancing the work of [8,9]. Motivated by real-world capacity constraints, but also following classical pathfinding research [10], we allow each edge and vertex to be used by only one agent at a time. A precise problem definition is given in Section 2, where we formalize the multi-agent pathfinding (MAPF) problem in the form of a labeling problem. Notwithstanding, we invite the reader to first study the background Section 1.1.

A main interest of this article is on classifying graphs where the MAPF problem is generally solvable with combinatorial criteria: That is, for any two initial and desired placements of agents, is there a valid sequence of moves solving the corresponding MAPF problem?

In Section 3, we give a clear-cut combinatorial classification of all graphs where this problem is solvable in general, including cases where the solvability depends on the initial agent placement. In the subsequent Section 4, we then give an algorithm[1] solving the MAPF problem in $\mathcal{O}(n^2)$ rounds and $\mathcal{O}(n^3)$ agent moves. Furthermore, we provide a class of graphs where any algorithm will require $\Omega(n^2)$ rounds and $\Omega(n^3)$ agent movements, matching our upper bounds. We conclude with a summary in Section 5.

1.1 Background

One of the earliest scientific works on multi-agent pathfinding on graphs is by Johnson and Story [11]: They studied the famous *15-puzzle*, where 15 agents $1, 2, \dots, 15$ are placed on a $4x4$-grid, and only one agent may move at a time to a currently unoccupied neighboring vertex. The authors showed that exactly half of the starting positions are not solvable, if the goal is to order the agents in an increasing pattern from 1 to 15, with the lower right vertex being unoccupied, and also studied larger grids – with Wilson showing the connection to alternating groups [12]. In more recent times, it was shown that finding the fastest solution for feasible problems is NP-hard already on grids, cf. [13], [14].

The model of the 15-puzzle, where one agent moves at a time to an unoccupied neighboring vertex, has been studied by numerous people in various communities. One such piece of work that this article draws foundations and techniques from, in particular for lower bounds, is *Coordinating Pebble Motion On Graphs, The Diameter Of Permutation Groups, And Applications* by Kornhauser, Miller, and Spirakis. Two versions exist, one is the Master's Thesis of Kornhauser which is available as a technical report [15]. A more compact version was published at *FOCS* in 1984 [10], omitting some proofs. Even though Kornhauser uses a different model where no rotations are allowed and enforcing one unoccupied node, we arrived at the same upper and lower bounds of $\mathcal{O}(n^3)$, respectively $\Omega(n^3)$ agent moves. Our proof of the $\Omega(n^3)$ lower bound in our model is very similar to that of Kornhauser, as noted in Section 4.4. While their results are from the 1980's, Röger and Helmert [16] pointed out in 2012 that these findings solve some open problems in the robotics community and are still relevant in current research.

The same model as in this article was previously studied by Yu and Rus in *Pebble Motion on Graphs with Rotations: Efficient Feasibility Tests and Planning Algorithms* [8]. The authors provided an algorithm to check if a graph instance is solvable, but did not give combinatorial criteria for feasibility as provided by us. Hence, they could also not provide statements about when exactly half of the MAPF problems are solvable, as we did in Section 3.3. Yu and Rus also give a MAPF algorithm, differing from our

[1] Yu and Rus [8] also give a MAPF algorithm, cf. second to last paragraph of Section 1.1.

methods, for which they prove an upper bound that is equivalent to our $\mathcal{O}(n^2)$ upper bound on the number of rotations. However, they did not show the lower bounds.

Lastly, Driscoll and Furst published a paper [9] in 1983 that gives a $\mathcal{O}(n^2)$ upper bound on the diameter of a class of permutation groups. While Driscoll and Furst's paper does not relate permutations to multi-agent pathfinding, our problem is in said class of permutation problems, and Driscoll and Furst's upper bound directly applies to the number of rotations in our problem. Driscoll and Furst also provide a generating set that leads to a tight lower bound, however this generating set can not be related to MAPF problems in the model discussed in this article, since it relies on two-cycles as generators.

2 Model

In this section we will first formally introduce the problem of multi-agent pathfinding, before providing some mathematical preliminaries for the concepts of permutations and permutation groups. We then use these tools to reformulate the MAPF problem as a labeling problem in Section 2.1. Multi-agent pathfinding (MAPF) on a graph describes a problem where k agents are distributed on vertices of a graph $G(V, E)$ with n vertices. Each agent has a destination, its goal vertex. Agents can move over edges to neighboring vertices. The problem is to find a sequence of moves, such that eventually all agents are on their goal vertex. In the problems studied here, there is always exactly one agent on each vertex, i.e., $k = n$. The movement of the agents is constrained by the following rules:

- At any given time, no more than one agent can be on any vertex.
- Any edge can only be used by one agent at a time, i.e., neighboring agents may not swap places.

The only permitted moves are thus *rotations* on *graph cycles*.

Definition 1 (rotation). *In a* rotation *on a graph cycle $v_1, \ldots, v_m$, the agent on a vertex v_i moves to the vertex v_{i+1} if $i \in \{1, \ldots, m-1\}$ or the vertex v_1 if $i = m$.*

To keep the terminology consistent with other works in Computer Science and Mathematics, we will be dealing with *labeled graphs* instead of *agents on graphs*:

Definition 2 (labeling). *Let $L = \{1, 2, 3, \ldots, |V|\}$ be the* set of labels. *A* labeling *of a graph $G(V, E)$ is a bijective function $l : V \to L$.*

Problems where objects are reordered are typically associated with the mathematical theory of *permutations* and *permutation groups*. In the following, we will give some of the basic definitions and results from those fields.

Definition 3 (permutation). *Let $X = 1, \ldots, n$. A* permutation *is a bijective function $\pi : X \to X$.*

There are multiple established notations for permutations. In the *two-line notation* one writes for each element x in the first row its image $\pi(x)$ in the second row:

$$\pi = \begin{pmatrix} l_1 & l_2 & l_3 & \ldots & l_{n-1} & l_n \\ \pi(l_1) & \pi(l_2) & \pi(l_3) & \ldots & \pi(l_{n-1}) & \pi(l_n) \end{pmatrix}$$

The second notation used here is the *cycle notation*: Starting from some element $x \in X$, one writes the sequence $\big(x \ \pi(x) \ \pi(\pi(x)) \ \dots\big)$ of successive images under π. The sequence is continued until x would appear again. Starting at a new element not observed yet, we do the same, and write it in a new pair of parentheses. This is repeated until every element is written down once.

Example 1. $\begin{pmatrix} 1\,2\,3\,4\,5\,6\,7 \\ 1\,5\,7\,2\,4\,3\,6 \end{pmatrix}$ could be written as $\big(1\big)\ \big(2\ 5\ 4\big)\ \big(3\ 7\ 6\big)$

Cycles of length one are omitted, the above permutation then reads as $\big(2\ 5\ 4\big)\ \big(3\ 7\ 6\big)$. Next, a pair of labels is called an inversion, if the order of said labels is changed by the permutation.

Definition 4 (inversion). *(l_i, l_j) is an* inversion *of π, if $l_i > l_j$ and $\pi(l_i) < \pi(l_j)$.*

Definition 5 (parity of a permutation). *The* parity of a permutation *is the parity (odd or even) of the number of inversions it contains.*

Definition 6 (composition of permutations). *Two (and, iteratively, any number of) permutations can be* composed*: $\pi_1 \circ \pi_2 = \pi_1 \pi_2 = \pi_2(\pi_1(x)) \quad \forall x \in X$.*

The set of all permutations on $1, \dots, n$ with operation $\circ$ form the group S_n. An important subgroup of S_n is the *alternating group A_n*. It is the subgroup of S_n which contains all even permutations. It contains exactly half of the $n!$ elements of S_n. We need the following lemma to see that closure is satisfied for A_n, which is proven in numerous textbooks on the subject, such as [17]:

Lemma 1. *The composition of even permutations is even.*

We can conclude by recursion, that the composition of any number of even permutations will again result in an even permutation. The set of even permutations is thus closed under composition.

2.1 Reformulation of the MAPF problem

The permitted operations in our model are rotations. They can be interpreted as an element of S_n. We refer to Figure 1 for an introductory case explained in Example 2.

Example 2. If we label the bow-tie graph with labels as in Figure 1, we can write the permutations corresponding to the rotations in cycle notation, e.g.:

- clockwise rotation in the left cycle: $\pi_{L^-} = \big(1\ 3\ 2\big)\ \big(4\big)\ \big(5\big) = \big(1\ 3\ 2\big)$,
- counterclockwise rotation in the right cycle: $\pi_{R^+} = \big(1\big)\ \big(2\big)\ \big(3\ 4\ 5\big) = \big(3\ 4\ 5\big)$,

In fact, rotations in our model always correspond to permutation cycles. (But not all permutation cycles correspond to a valid move.) It is thus justified to reformulate the MAPF problem:

Main Idea. *Let π_{goal} be the permutation that represents the goal labeling and let P_G be the set of permutations that correspond to a valid rotation.*

Find a sequence $\pi_{r_1}, \ldots, \pi_{r_m}$, where$\pi_{r_i} \in P_G$ such that

$$\pi_{r_1} \circ \pi_{r_2} \circ \ldots \circ \pi_{r_m} = \pi_{\text{goal}} \tag{1}$$

This problem has a solution if and only if π_{goal} is an element of the group generated by P_G.

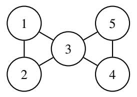


Fig. 1: Labeled bow-tie graph, consisting of two odd cycles of length three.

Lemma 2. *Rotations on graph cycles with even length correspond to odd permutations. Rotations on odd-length graph cycles correspond to even permutations.*

Proof. Rotations on graph cycles with length i correspond to permutation cycles of the same length i. It is known, cf. [17], that cyclic permutations of even length correspond to odd permutations and vice-versa. Therefore odd i give rise to even permutations, even i to odd ones. □

3 Necessary and Sufficient Combinatorial Criteria for Solvability

We will begin this section with Theorem 1, where we specify necessary and sufficient combinatorial criteria for graphs on which the MAPF problem can be generally solved.

Definition 7. *The MAPF problem is* generally solvable *on a graph G, if the MAPF problem is solvable on G for any combination of an initial labeling with a goal labeling.*

Theorem 1. *The MAPF-problem on a graph G with $n \geq 2$ vertices is generally solvable, if and only if the following conditions hold:*

1. *G is 2-edge-connected,*
2. *G contains at least two cycles,*
3. *G contains a cycle of even length.*

In the following Section 3.1, we address the necessity of these criteria. Then, in Section 3.2, we point out that graphs fulfilling the criteria are indeed solvable, i.e., the conditions are sufficient.

Lastly in this section, we address in Section 3.3 that half of the MAPF problems are still solvable if the particular requirement that a graph must contain an even-length cycle is not satisfied.

3.1 The Combinatorial Conditions in Theorem 1 are Necessary

We defer the proof of conditions 1 and 2 to the full version of this article. Condition 3 is proven in the following lemma.

Lemma 3. *The MAPF-problem on a graph G is not generally solvable, if the graph does not contain an even-length cycle.*

Proof. Assume Graph G contains only odd-length cycles. Lemma 2 then implies that all π_{r_i} of Equation 1 are even. Using Lemma 1, we see that the permutation problem can not be solved for odd π_{goal} and we will always stay in A_n. □

In fact, as we will see in Section 3.3, all problems corresponding to even π_{goal} are solvable, when this last constraint is not satisfied. That is, exactly half of all problems are still solvable in that case.

3.2 The Combinatorial Conditions in Theorem 1 are Sufficient

In this section, we show that the MAPF problem on the graphs specified in Theorem 1 are indeed generally solvable. We will show that on such graphs it is possible to exchange any two labels while leaving all other labels unaffected. In terms of permutations this amounts to being able to express 2-cycles as a sequence of the permutations corresponding to the permitted rotations. (cf. our main idea). Since the set of all 2-cycles generates S_n (cf. [17]), this will conclude the proof of Theorem 1.

3.2.1 Swapping two Labels in a Generally Solvable Graph

Lemma 4. *Let*

$$\pi_{a\times b}(x) = \begin{cases} b & \text{if } x = a \\ a & \text{if } x = b \\ x & \text{otherwise} \end{cases} \qquad \pi_{(l_1,l_2)\to(s_1,s_2)}(x) = \begin{cases} s_1 & \text{if } x = l_1 \\ s_2 & \text{if } x = l_2 \\ x' & \text{otherwise} \end{cases}$$

where x' in $\pi_{(l_1,l_2)\to(s_1,s_2)}$ is arbitrary, with the constraint that $\pi_{(l_1,l_2)\to(s_1,s_2)}$ is bijective. Then,

$$\pi_{l_1\times l_2} = \pi_{(l_1,l_2)\to(s_1,s_2)}\pi_{s_1\times s_2}\pi^{-1}_{(l_1,l_2)\to(s_1,s_2)} \tag{2}$$

Proof. To make notation less cumbersome, we will denote $\pi_{(l_1,l_2)\to(s_1,s_2)}$ by π. The right hand side of Equation 2 can be rewritten as $\pi\pi_{s_1\times s_2}\pi^{-1} = \pi^{-1}(\pi_{s_1\times s_2}(\pi(x)))$. We distinguish cases:

Case $x \neq l_1 \wedge x \neq l_2$: Then, $\pi(x) = x'$. Since $x' \neq s_1$ and $x' \neq s_2$ we have $\pi_{s_1\times s_2}(x') = x'$. We can then plug these values in as follows:

$$\pi^{-1}(\pi_{s_1\times s_2}(\pi(x))) = \pi^{-1}(\pi_{s_1\times s_2}(x')) = \pi^{-1}(x') = x$$

Case $x = l_1$: $\pi(l_1) = s_1$ and $\pi_{s_1\times s_2}(s_1) = s_2$:

$$\pi^{-1}(\pi_{s_1\times s_2}(\pi(x))) = \pi^{-1}(\pi_{s_1\times s_2}(s_1)) = \pi^{-1}(s_2) = l_2$$

Case $x = l_2$: analogously

In all cases, we have $\pi^{-1}(\pi_{s_1\times s_2}(\pi(x))) = \pi_{l_1\times l_2}(x)$, concluding the proof. □

In other words, if we can swap a specific pair of labels (s_1 and s_2) without affecting other labels, and we are able to move any pair of labels (l_1 and l_2) to the position of the aforementioned labels, we can effectively swap any two labels by means of Equation 2. It remains to prove that we can express some $\pi_{(l_1,l_2)\to(s_1,s_2)}$ and the suitable $\pi_{s_1\times s_2}$ for any l_1 and l_2 by means of the permitted rotations.

Lemma 5. *For any cycle c_1 in a graph that is 2-edge-connected with at least two cycles, one of the following two options holds:*

1. *There is a cycle c_2 with which it shares exactly one vertex or*
2. *There are 2 vertices in c_1 with 3 vertex-disjoint paths between them.*

The proof of this lemma is deferred to the full version of this article. According to this lemma, finding $\pi_{s_1 \times s_2}$ for all 2-edge-connected graphs can be done by finding $\pi_{s_1 \times s_2}$ in each of the stated cases. We will now demonstrate how swapping is possible in either case.

3.2.2 Swapping Labels in Cycles Sharing Exactly One Vertex Let C_{n_l,n_r} denote a graph consisting of two cycles, with sizes n_l and n_r, respectively, that share exactly one vertex. As there are two cycles, four operations are permitted, namely rotations in both directions on either cycle. π_{L^-} denotes the permutation associated with a clockwise rotation in the left cycle, π_{L^+} the permutation associated with a counterclockwise rotation in the left cycle. π_{R^+} and π_{R^-} are the analogous counterparts in the right cycle. Algorithm 1 describes a procedure to swap the labels l_1 and m in C_{n_l,n_r}.

Lemma 6. *If n_l is even, Algorithm 1 terminates.*

Proof. The permutations associated with the basic rotations that we use can readily be written down in cycle notation:

$$\pi_{L^-} = \left(l_1\ m\ l_{n_l-1} \dots l_2\right) \quad \pi_{R^+} = \left(r_1\ r_2 \dots r_{n_r-1}\ m\right) \quad \pi_{R^-} = \left(r_1\ m\ r_{n_r-1} \dots r_2\right)$$

Building on these, we can write down the composed permutations of the algorithm:

$$\begin{aligned}\pi_{\text{init}} &= \pi_{R^-}\pi_{L^-}\pi_{L^-}\pi_{R^+}\pi_{L^-} \\ &= \begin{pmatrix} l_1 & l_2 & l_3 & l_4 & l_5 & l_6 & l_7 & \dots & l_{n_l-3} & l_{n_l-2} & l_{n_l-1} & m & r_1 & r_2 & r_3 & \dots & r_{n_r-2} & r_{n_r-1} \\ l_{n_l-2} & r_1 & m & l_1 & l_2 & l_3 & l_4 & \dots & l_{n_l-6} & l_{n_l-5} & l_{n_l-4} & l_{n_l-1} & l_{n_l-3} & r_2 & r_3 & \dots & r_{n_r-2} & r_{n_r-1} \end{pmatrix}\end{aligned}$$

$$\begin{aligned}\pi_{\text{step}} &= \pi_{R^-}\pi_{L^-}\pi_{R^+}\pi_{L^-} \\ &= \begin{pmatrix} l_{n_l-2} & r_1 & m & l_1 & l_2 & l_3 & l_4 & \dots & l_{n_l-6} & l_{n_l-5} & l_{n_l-4} & l_{n_l-1} & l_{n_l-3} & r_2 & r_3 & \dots & r_{n_r-2} & r_{n_r-1} \\ l_{n_l-4} & l_{n_l-2} & l_{n_l-1} & r_1 & m & l_1 & l_2 & \dots & l_{n_l-8} & l_{n_l-7} & l_{n_l-6} & l_{n_l-3} & l_{n_l-5} & r_2 & r_3 & \dots & r_{n_r-2} & r_{n_r-1} \end{pmatrix}\end{aligned}$$

Assuming n_l is even, π_{step} reads in cycle notation:

$$\pi_{\text{step}} = \left(l_1\ r_1\ l_{n_l-2}\ l_{n_l-4} \dots l_2\ m\ l_{n_l-1}\ \ l_{n_l-3} \dots l_3\right)$$

We left out out $\frac{n_l}{2} - 4$ labels with each "…", namely l_i's with even i in the left case and with odd i in the the right. Note that the labels $l_1, \dots, l_{n_l-3}$ always take the place of the label with an index that is larger by 2. If n_l was odd, $n_l - 1$ would be even and l_{n_l-1} would be in the cycle much earlier, such that not all labels would be in the same cycle.

Applying a cyclic permutation k-fold has step the labels k steps forward in the order of the cycle. For each label x in the permutation cycle we can count k positions to the right in the cyclic representation of π_{step} to find $\pi^{k}_{\text{step}}(x)$. In this way, we find $\pi_{\text{step}}^{\frac{n_l}{2}-1}$:

$$\pi_{\text{step}}^{\frac{n_l}{2}-1} = \begin{pmatrix} l_{n_l-2} & r_1 & m & l_1 & l_2 & l_3 & l_4 & \dots & l_{n_l-6} & l_{n_l-5} & l_{n_l-4} & l_{n_l-1} & l_{n_l-3} & r_2 & r_3 & \dots & r_{n_r-2} & r_{n_r-1} \\ m & l_2 & l_3 & l_4 & l_5 & l_6 & l_7 & \dots & l_{n_l-3} & l_{n_l-2} & l_{n_l-1} & l_1 & r_1 & r_2 & r_3 & \dots & r_{n_r-2} & r_{n_r-1} \end{pmatrix}$$

We've written down $\pi_{\text{step}}^{\frac{n_l}{2}-1}$ such that it is easy to see that

$$\pi_{\text{init}}\pi_{\text{step}}^{\frac{n_l}{2}-1} = \begin{pmatrix} l_1 & l_2 & l_3 & l_4 & l_5 & l_6 & l_7 & \dots & l_{n_l-3} & l_{n_l-2} & l_{n_l-1} & m & r_1 & r_2 & r_3 & \dots & r_{n_r-2} & r_{n_r-1} \\ m & l_2 & l_3 & l_4 & l_5 & l_6 & l_7 & \dots & l_{n_l-3} & l_{n_l-2} & l_{n_l-1} & l_1 & r_1 & r_2 & r_3 & \dots & r_{n_r-2} & r_{n_r-1} \end{pmatrix}$$

Which is our goal permutation.That is, after $\frac{n_l}{2} - 1$ repetitions of the loop in Algorithm 1, we are at the desired configuration, and the algorithm terminates. □

Algorithm 1: Swapping Two Labels in C_{n_l,n_r}

```
$\pi := \pi_{R-}\pi_{L-}\pi_{L-}\pi_{R+}\pi_{L-}$
$\pi_{\text{step}} := \pi_{R-}\pi_{L-}\pi_{R+}\pi_{L-}$
while $\pi \neq \pi_{\text{goal}}$ do
    $\pi := \pi\pi_{\text{step}}$
```

3.2.3 Swapping Labels in a Cycle Containing 2 Vertices with Three Paths Between Them In the case when there are two vertices with three vertex-disjoint paths between them, swapping two labels is simpler, and possible with just 3 rotations. One possibility of performing such a swap is illustrated in Figure 2.

3.2.4 Travelling to Swapspot It remains to express $\pi_{(l_1,l_2)\to(s_1,s_2)}$ for any l_1, l_2, s_1 and s_2, where the initial vertices of s_1 and s_2 are neighbors. We will do this in two phases. First, we move l_1 and l_2 such that they are neighbors. Then, these neighbors are moved to the place where they can be swapped. For details, we refer to the full version.

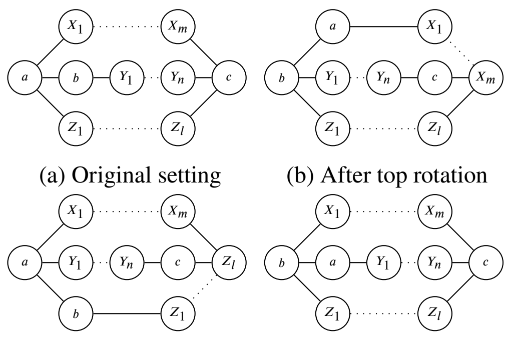
3.3 Solvable Problems on not Generally Solvable Graphs

We have now specified the class of graphs on which the MAPF problem is generally solvable. On those that are not generally solvable, some problems are still solvable. In the cases where a graph is not 2-edge connected, one can consider each 2-edge connected component separately, as no label can cross bridges. The solvable problems are then those where the labels only travel within subgraphs that fulfill the constraints of Theorem 1. Another case is when there is only one cycle present, where the solvable problems are exactly those obtained by rotations on this cycle.

However, if a graph is still 2-edge connected and contains at least two cycles, but only contains cycles of odd length, a more interesting observation can be made. In fact, exactly half of the problems can still be solved. In Section 3.2 we presented a method to express 2-cycles as a sequence of the permitted rotations. Without the presence of even cycles, it is possible to express 3-cycles with a very similar method. Recall that 3-cycles are a generating set of the alternating group A_n, which contains half of the elements of S_n. The details of 3-cycling are deferred to the full version.

4 Algorithms, Lower and Upper Bounds

In this section, we use the mechanisms studied so far to construct an algorithm that solves the MAPF problem in $\mathcal{O}(n^3)$ label movements and $\mathcal{O}(n^2)$ rotations. We will also present a class of graphs, on which the MAPF problem cannot be solved with less than $\Omega(n^3)$ label movements and $\Omega(n^2)$ rotations, meaning that our algorithm is optimal in terms of the asymptotic number of operations in the worst case.



(a) Original setting (b) After top rotation

(c) After outside rotation (d) After bottom rotation

Fig. 2: Swapping in graphs with two vertices with three vertex-disjoint paths between them.

4.1 Complexity Measures

The complexity of a solution to the problem can be described in different ways. In this chapter we will investigate the complexity with respect to three related measures. An upper bound on the length of the sequence of permutations found by the algorithm is given in all three measures, and and a class of graphs is given on which these upper bounds for all three measures are tight in an asymptotic sense.

One way of describing the complexity of a solution is that of the total number of *rotations*. In this case, every rotation increases the complexity by one. This is effectively the length of the sequence found in our main idea. For some problem instances, rotations can be performed in parallel. On these problems, measuring the complexity with the number of rotations might not give a good representation of the running time. The number of *rounds* thus can be used as a second measure. However, the algorithms used in this paper never use the possibility of parallel rotations. Therefore here, the number of rotations equals the number of rounds. As we will see, our lower bounds are tight regardless. Lastly, the number of *label movements* is studied. That is, the number of rotations a label was involved in, summed up over all labels.

4.2 The Algorithm

We have established the notions of swapping and 3-cycling labels. Using these mechanisms we can directly build an algorithm:

1. As long as there are labels in the wrong place, pick one wrongly placed label, say a. Then, pick the label $b := \pi_{\text{goal}}(a)$.
2. Set c to be an arbitrary incorrectly placed label such that $a \neq c$ and $\pi_{\text{goal}}(a) \neq c$. If no such c can be found, a and b are the only wrongly placed labels left, and we swap them. If swapping is not possible, the problem is not solvable. (Since the solution is only one swap away, π_{goal} is not in A_n.)
3. If a c is found, 3-cycle a, b and c. Since this only moves wrongly placed labels, and fixes the position of a, this decreases the overall number of wrongly placed labels by at least one.

4. Repeat until all labels are at the right place.

Note that by better choices of a,b and c, we can fix at least two labels with every 3-cycle. However, this leads to a sequence of operations of the same asymptotic length.

4.3 Upper Bound on Number of Operations

Lemma 7. *Swapping two labels and 3-cycling three labels without affecting any other labels both take $\mathcal{O}(n)$ rotations.*

For a full proof, we refer to the full article. The gist is that there is a constant number of steps involved, each with a complexity in $\mathcal{O}(n)$ rotations. These complexities are mainly determined by the lenght of paths and cycles in the graph.

Theorem 2. *The Algorithm described in Section 4.2 terminates in $\mathcal{O}(n^2)$ rotations, $\mathcal{O}(n^2)$ rounds and $\mathcal{O}(n^3)$ label movements.*

Proof. Since on n labels and n vertices, there can be at most n wrongly placed labels, and we fix at least one with every 3-cycle and every swap, we will need at most n such operations. In other words, the added number of 3-cycles and swaps performed is in $\mathcal{O}(n)$.

We have seen in Lemma 7 that both swapping and cycling take $\mathcal{O}(n)$ rotations. Having $\mathcal{O}(n)$ swaps or cycling operations costing $\mathcal{O}(n)$ rotations each, we get the claimed overall bounds of $\mathcal{O}(n^2)$ rotations. Clearly, each rotation moves at most n labels, which directly implies the upper bound of $\mathcal{O}(n^3)$ label movements.

The worst case in terms of number of rounds, is when all rotations are done sequentially. Therefore, an upper bound on the number of rotations is also an upper bound on the number of rounds. I.e., the upper bound of $\mathcal{O}(n^2)$ rotations directly implies an upper bound of $\mathcal{O}(n^2)$ rounds. □

4.4 Lower Bound on Number of Operations

We will now give a class of graphs and a MAPF problem on which any algorithm takes at least $\Omega(n^2)$ rotations, $\Omega(n^2)$ rounds and $\Omega(n^3)$ label movements, providing lower bounds that are asymptotically tight. The class of graphs is the same as Kornhauser et al. [10] used for their model.

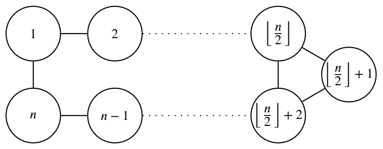

1
2
$\lfloor \frac{n}{2} \rfloor$
$\lfloor \frac{n}{2} \rfloor + 1$
n
$n-1$
$\lfloor \frac{n}{2} \rfloor + 2$

Fig. 3: Graph LB_n for the proof of the lower bound

Consider the graph of Figure 3, that is the cyclic graph on n vertices with an added edge between the $\lfloor \frac{n}{2} \rfloor$-th and the $\lfloor \frac{n}{2} + 2 \rfloor$-th vertex. We denote this graph by LB_n.

4.4.1 Rotations and Rounds

Lemma 8. *There is a MAPF problem on LB_n for which any solution requires $\Omega(n^2)$ rotations.*

Proof. Assume n to be odd. We define d_i to be the *semi-circular distance* between label i and label $i+1$. The semi-circular distance is the shortest path between the labels on the cyclic graph, that does not use the added edge. The d_i are maximal for $d_i = \lfloor \frac{n}{2} \rfloor$, and are at least 1.

Following Kornhauser [10], we define the notion of *entropy* as $E = \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} d_i$. We chose an initial labeling, for which $E = \lfloor \frac{n}{2} \rfloor^2$, with our goal configuration having $E = \lfloor \frac{n}{2} \rfloor$. There are six permitted operations on LB_n: A rotation on the outer cycle, denoted by A, a rotation on the cycle $\left(\lfloor \frac{n}{2} \rfloor \;\; \lfloor \frac{n}{2} \rfloor + 1 \;\; \lfloor \frac{n}{2} \rfloor + 2\right)$, denoted by B and a rotation on the cycle not including $\lfloor \frac{n}{2} \rfloor + 1$, denoted by C, as well as their respective inverses A^{-1}, B^{-1} and C^{-1}. We can study the effect of the three operations on the entropy. Clearly, A and A^{-1} do not change the entropy. Rotating B or C can only change the d_i that include the labels on vertices $\lfloor \frac{n}{2} \rfloor$, $\lfloor \frac{n}{2} \rfloor + 1$ and $\lfloor \frac{n}{2} \rfloor + 2$, and by at most 2 each. Each rotation thus decreases E by at most 12. Having $E = \lfloor \frac{n}{2} \rfloor^2$ at the beginning and $E = \lfloor \frac{n}{2} \rfloor$ at the goal configuration, we can say that we need at least $\frac{\lfloor \frac{n}{2} \rfloor^2 - \lfloor \frac{n}{2} \rfloor}{12} \in \Omega(n^2)$ operations. □

Lemma 9. *There is a MAPF problem on LB_n for which any solution requires $\Omega(n^2)$ rounds.*

Proof. Since all three cycles in LB_n pairwise share vertices, only one rotation can be performed at a time. Therefore, the lower bound on the number of rotations from Lemma 8 is also a lower bound for the number of rounds. □

4.4.2 Label Movements

We now look at the number of label movements.

Lemma 10. *There is a MAPF problem on LB_n where any solution requires $\Omega(n^3)$ label movements.*

Proof. We can assume that in an optimal solution, no more than one consecutive operation is performed on cycle B. (Since, e.g., BB can be replaced by B^{-1}, $B^{-1}B$ by doing nothing at all, consecutive operations on B indicate non-optimal solutions.) We thus know, that after each operation on B, there will be one on either A or C. Thus, if there are m operations, at least $\lfloor \frac{m}{2} \rfloor$ operations are performed on A and C. Those require at least $n-1$ label movements. (Namely, if C is moved.) As any solution will use at least $\Omega(n^2)$ rotations, so will a solution that is optimal with respect to label movements. Hence, $(n-1)\lfloor \frac{\Omega(n^2)}{2} \rfloor \in \Omega(n^3)$ is a lower bound for the number of label movements. □

5 Conclusion

We studied combinatorial classifications and algorithms for the multi-agent pathfinding (MAPF) problem on graphs G with n agents. We proved that the MAPF problem is only generally solvable, if the graphs G are 2-edge-connected, contain at least two cycles, and contain at least one cycle of even length. Should the last of these three combinatorial conditions be violated, we showed that exactly half of the MAPF problems on these graphs are solvable.

Furthermore, we specified an algorithm that solves feasible MAPF problems in $\mathcal{O}(n^2)$ operations or $\mathcal{O}(n^3)$ agent-movements. We also specified a class of graphs, where at least $\Omega(n^2)$ operations or $\Omega(n^3)$ agent-movements are required, meaning that on general graphs, our algorithms are asymptotically optimal.

Acknowledgements We would like to thank the anonymous reviewers for their helpful comments. Klaus-Tycho Foerster is supported by the Danish Villum Foundation.

References

1. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numerische Mathematik **1**(1) (1959) 269–271
2. Hart, P.E., Nilsson, N.J., Raphael, B.: A Formal Basis for the Heuristic Determination of Minimum Cost Paths. IEEE Transactions on Systems Science and Cybernetics **4**(2) (1968) 100–107
3. Scott, R.: Sparking life: Notes on the performance capture sessions for the lord of the rings: The two towers. SIGGRAPH Comput. Graph. **37**(4) (2003) 17–21
4. Silver, D.: Cooperative pathfinding. In: Artificial Intelligence and Interactive Digital Entertainment Conference. (2005)
5. Pelechano, N., Malkawi, A.: Evacuation simulation models: Challenges in modeling high rise building evacuation with cellular automata approaches. Automation in Construction **17**(4) (2008) 377 – 385
6. Svestka, P., Overmars, M.H.: Coordinated path planning for multiple robots. Robotics and Autonomous Systems **23**(3) (1998) 125 – 152
7. Domke, J., Hoefler, T., Matsuoka, S.: Routing on the Dependency Graph: A New Approach to Deadlock-Free High-Performance Routing. In: Symposium on High-Performance Parallel and Distributed Computing. (2016)
8. Yu, J., Rus, D.: Pebble motion on graphs with rotations: Efficient feasibility tests and planning algorithms. In: Selected Contributions of the Eleventh International Workshop on the Algorithmic Foundations of Robotics. (2015)
9. Driscoll, J.R., Furst, M.L.: On the diameter of permutation groups. In: Symposium on Theory of Computing. (1983)
10. Kornhauser, D., Miller, G., Spirakis, P.: Coordinating pebble motion on graphs, the diameter of permutation groups, and applications. In: Symposium on Foundations of Computer Science. (1984)
11. Wm. Woolsey Johnson, W.E.S.: Notes on the "15" Puzzle. American Journal of Mathematics **2**(4) (1879) 397–404
12. Wilson, R.M.: Graph puzzles, homotopy, and the alternating group. Journal of Combinatorial Theory, Series B **16**(1) (1974) 86 – 96
13. Ratner, D., Warmuth, M.: the $n^2 - 1$ puzzle and related relocation problems. Journal of Symbolic Computation **10**(2) (1990) 111 – 137
14. Goldreich, O.: Finding the Shortest Move-sequence in the Graph-generalized 15-puzzle is NP-hard. In: Studies in Complexity and Cryptography. Springer-Verlag, Berlin, Heidelberg (2011) 1–5
15. Kornhauser, D.: Coordinating pebble motion on graphs, the diameter of permutation groups, and applications. Master's Thesis MIT/LCS/TR-320, Massachusetts Institute of Technology (1984)
16. Röger, G., Helmert, M.: Non-optimal multi-agent pathfinding is solved (since 1984). In: Symposium on Combinatorial Search. (2012)
17. Jacobson, N.: Basic algebra. San Francisco - Calif. : Freeman (1974)