Multi-Agent Path Finding on Strongly Connected Digraphs

S. Ardizzoni1, I. Saccani 1, L. Consolini 1, M. Locatelli 1

1Dipartimento di Ingegneria e Architettura, Università di Parma, Parco Area delle Scienze, 181/A, Parma, Italy

Abstract—On an assigned graph, the problem of Multi-Agent Pathfinding (MAPF) consists in finding paths for multiple agents, avoiding collisions. Finding the minimum-length solution is known to be NP-hard, and computation times grows exponentially with the number of agents. However, in industrial applications, it is important to find feasible, suboptimal solutions, in a time that grows polynomially with the number of agents. Such algorithms exist for undirected and biconnected directed graphs. Our main contribution is to generalize these algorithms to the more general case of strongly connected directed graphs. In particular, given a MAPF problem with at least two holes, we present an algorithm that checks the problem feasibility in linear time with respect to the number of nodes, and provides a feasible solution in polynomial time.

I. INTRODUCTION

We consider a graph and a set of agents. Each agent occupies a different node and may move to unoccupied positions. The Multi-Agent Path Finding (MAPF) problem consists in computing a sequence of movements that repositions all agents to assigned target nodes, avoiding collisions. In this paper, we deal with strongly connected digraphs, directed graphs in which it is possible to reach any node starting from any other node. The main motivation comes from the management of fleets of automated guided vehicles (AGVs). AGVs move items between different locations in a warehouse. Each AGV follows predefined paths, that connect the locations in which items are stored or processed. We associate the paths’ layout to a directed graph. The nodes represent positions in which items are picked up and delivered, together with additional locations used for routing. The directed arcs represent the precomputed paths that connect these locations. If various AGVs move in a small scenario, each AGV represents an obstacle for the other ones. In some cases, the fleet can reach a deadlock situation, in which every vehicle is unable to reach its target. Hence, it is important to find a feasible solution to MAPF, even in crowded configurations.

Literature review. Various works address the problem of finding the optimal solution of MAPF (i.e., the solution with the minimum number of moves). For instance, Conflict Based Search (CBS) is a two-level algorithm which uses a search tree, based on conflicts between individual agents (see [10]). However, finding the optimal solution of MAPF is NP-hard (see [9]). Search-based suboptimal solvers aim to provide a high quality solution, but are not complete (i.e., they are not always able to return a solution). An example is Hierarchical Cooperative A* (HCA*) [11], in which agents are planned one at a time according to some predefined order. Instead, rule-based approaches define specific movements for different scenarios. They favor completeness at low computational cost over solution quality. Two important rule-based algorithms are TASS [3] and Push and Rotate [12] [13].

The literature cited so far concerns exclusively undirected graphs. Fewer results are related to directed graphs. Reference [15] proves that finding a feasible solution of MAPF on a general directed graph (digraph) is NP-hard. However, in some special cases this problem can be solved in polynomial time. One relevant reference is [5], which solves MAPF on the specific class of biconnected digraphs, i.e., strongly connected digraphs where the undirected graphs obtained by ignoring the edge orientations have no cutting vertices. The proposed algorithm has polynomial complexity with respect to the number of nodes.

Statement of contribution. We consider MAPF on strongly connected digraphs, a class that is more general than biconnected digraphs, already addressed in [5]. To our knowledge, this is the first work that considers this specific problem. Essentially, our approach generalizes the method presented in [4] to digraphs. Namely, we decompose the graph into biconnected components, and use some of the methods presented in [5] to reconfigure the agents in each biconnected component. We present a procedure, based on [2], that checks the problem feasibility in linear time with respect to the number of nodes. Also, we present diSC (digraph Strongly Connected) algorithm that finds a solution for all admissible problems with at least two holes, extending the method in [4].

II. PROBLEM DEFINITION

Let $G = (V, E)$ be a digraph, with vertices $V$ and directed edges $E$. We assign a unique label to each pebble and hole. Sets $P$ and $H$ contain the labels of the pebbles and, respectively, the holes. Each vertex of $G$ is occupied by either a pebble or a hole, so that $|V| = |P| + |H|$. A configuration is a function $\mathcal{A} : P \cup H \to V$ that assigns the occupied vertex to each pebble or hole. A configuration is valid if it is one-to-one (i.e., each vertex is occupied by only one pebble or hole). Set $C \subset \{P \cup H \to V\}$ represents all valid configurations.

Given a configuration $\mathcal{A}$ and $u, v \in V$, we denote by $\mathcal{A}[u, v]$ the configuration obtained from $\mathcal{A}$ by exchanging the pebbles (or holes) placed at $u$ and $v$:

$$\mathcal{A}[u, v](q) := \begin{cases} v, & \text{if } \mathcal{A}(q) = u; \\ u, & \text{if } \mathcal{A}(q) = v; \\ \mathcal{A}(q), & \text{otherwise}. \end{cases} \quad (1)$$

Function $\rho : C \times E \to C$ is a partially defined transition function such that $\rho(\mathcal{A}, u \to v)$ is defined if and only if $v$ is empty (i.e., occupied by a hole). In this case $\rho(\mathcal{A}, u \to v)$ is the configuration obtained by exchanging the pebble or the hole in $u$ with the hole in $v$. Notation $\rho(\mathcal{A}, u \to v)!$ means that the function is well-defined. In other words $\rho(\mathcal{A}, u \to v)!$ if and only if $(u, v) \in E$ and $A^{-1}(v) \in H$, and, if $\rho(\mathcal{A}, u \to v)!$, $\rho(\mathcal{A}, u \to v) = \mathcal{A}[u, v]$. Note that the hole in $v$ moves along edge $u \to w$ in reverse direction, while pebble or hole on $u$ moves on $v$.

We represent plans as ordered sequences of directed edges. It is convenient to view the elements of $E$ as the symbols of a language. We denote by $E^*$ the Kleene star of $E$, that is the set of ordered sequences of elements of $E$ with arbitrary
length, together with the empty string $\epsilon$:

$$E^* = \bigcup_{i=1}^{\infty} E^i \cup \{\epsilon\}.$$ 

We extend function $\rho : C \times E \to C$ to $\rho : C \times E^* \to C$ if $G(D)$ is connected. Proposition 4.2 leads to the following result about the feasibility of MAPF on digraphs:

Figure 1 shows an undirected graph and its corresponding biconnected component tree. $G$ and $T(G)$ have the same number of pebbles and the same number of holes, since trans-shipment vertices are not considered as free. Building $T(G)$ from $G$ takes a linear time with respect to $|E|$ [14].

Figure 1: Undirected graph and corresponding biconnected component tree. $A$, $B$, and $C$ are the trans-shipment vertices. Let $A : P \cup H \to V$ be a configuration on $G$. We associate it to a configuration on $T(G)$, $\tilde{A} : P \cup H \to V$ such that $(\forall q \in P \cup H) \tilde{A}(q) = A(q)$. Note that the codomain of $\tilde{A}$ is $V$, not $V_T$, since trans-shipment nodes are not present in $G$. In this way, we associate every MAPF instance on $G$ to a PMT instance on $T$. Reference [2] proves the following important result.

**Theorem 3.1.** [2] MAPF on graph $G$ is feasible if and only if PMT on tree $T(G)$ is feasible.

Since feasibility of PMT on a tree $T = (V_T, E_T)$ is decidable in $O(|V_T|)$ time (see [1]), it follows that:

**Theorem 3.2.** The feasibility of a MAPF instance on an undirected graph $G = (V_G, E_G)$ is decidable in $O(|V_G|)$ time.

**B. Solving PMT**

Since solving MAPF is equivalent to solving PMT, we recall the algorithm which solves PMT presented in [4], inspired by the feasibility test presented in [1]. The idea is to reduce the PMT problem to PPT by moving each pebble into one of the target positions. This reduction can be achieved in linear time with respect to $|V_T|$. Then, PPT is solvable if for every pebble $p$ there exists an exchange plan $f_{A^*(p)A^i(p)}$, which swaps $p$ with the pebble occupying its target position. Feasibility of the swap between two pebbles can be checked in constant time. We can solve PPT with TASS, proposed in [3].

**IV. STRONGLY CONNECTED DIGRAPHS**

As said, we consider MAPF for strongly connected digraphs.

**Definition 4.1.** A digraph $D = (V, E)$ is strongly connected if for each $v, w \in V$, $v \neq w$, there exist a directed path from $v$ to $w$, and a directed path from $w$ to $v$ in $D$.

As shown in Proposition 13 of [6], in strongly connected digraphs each move is reversible. From this, a more general result follows:

**Proposition 4.2.** In a strongly connected digraph each plan has a reverse plan.

Given a digraph $D$, we indicate with $G(D)$ its underlying graph, that is the undirected graph obtained by ignoring the orientations of the edges. Note that $D$ is strongly connected only if $G(D)$ is connected. Proposition 4.2 leads to the following result about the feasibility of MAPF on digraphs:
Theorem 4.3. Let $D = (V_D, E_D)$ be a strongly connected digraph. Then,

1) any MAPF instance on $D$ is feasible if and only if it is feasible on the underlying graph $G = G(D)$;
2) feasibility of any MAPF instance on $D$ is decidable in linear time with respect to $|V_D|$.

Proof. 1) The necessity is obvious. To prove sufficiency, let $f'$ be a plan which solves a MAPF instance on $G(D)$. Then we can define a plan $f$ on $D$ in the following way. For each pebble move $u \to v$ in $f'$, if $(u, v) \in E_D$, we perform move $u \to v$ on $D$. Otherwise, since $(v, u) \in E_D$, we execute a reverse plan for $v \to u$, $(v \to u)^{-1}$, that exists by Proposition 4.2.

2) It follows from Theorem 3.2.

A direct consequence of Theorem 4.3 and Theorem 3.1 is the following important result:

Corollary 4.4. MAPF on strongly connected digraph $D$ is feasible if and only if PMT on tree $T(G(D))$ (i.e., the biconnected component tree of the underlying graph of $D$) is feasible.

The proof of Theorem 4.3 leverages the reversibility of each pebble motion in strongly connected digraphs. It presents a simple algorithm that reduces MAPF for strongly connected digraphs to the undirected graphs case. However, this approach leads to very redundant solutions, since it does not exploit the directed graph structure. This fact is illustrated in Fig. 2, that shows a digraph $D$ and its associated underlying graph $G(D)$.

![Diagram](attachment:image.png)

Figure 2: A digraph $D$ and its underlying graph $G(D)$.

Example 4.5. A pebble $p$ is placed at node 2, while all other nodes are free. We want to move $p$ to 5. Plan $f' = (2 \to 1)(1 \to 5)$ is a solution of the corresponding problem on $G(D)$. We convert this to a plan on $D$ by applying the method in Theorem 4.3. Since $(2, 1) \notin E_D$, move $(2 \to 1)$ is converted into plan $(2, 3)(3, 4)(4, 5).$ Similarly, move $(1 \to 5)$ is converted into $(1, 2)(2, 3)(3, 4)(4, 5).$ This solution is redundant, since shorter plan $f = (2 \to 3)(3 \to 4)(4 \to 5)$ solves the overall problem.

To find shorter solutions, we avoid using the method described in Theorem 4.3, and present a method that takes into account the structure of the directed graph. In particular, we will exploit the fact that strongly connected digraphs can be decomposed in strongly biconnected components. In each component, we will use the method presented in [5]. First, we recall the following definition.

Definition 4.6. A digraph $D$ is said to be strongly biconnected if $D$ is strongly connected and $G(D)$ is biconnected.

We recall that an undirected graph $G$ is biconnected if it is connected and there are no cut vertices, i.e., the graph remains connected after removing any single vertex. The partially-bidirectional cycle is a simple example of a strongly biconnected digraph.

Definition 4.7. A digraph is a partially-bidirectional cycle if it consists of a simple cycle $C$, plus zero or more edges of the type $(u, v)$, where $(v, u) \in C$ (i.e., edges obtained by swapping the direction of an edge from $C$).

Reference [6] shows that strongly biconnected (respectively, strongly connected) digraphs have an open (respectively, closed) ear decompositions. We recall the definitions of open and closed ear decompositions. Given a graph $D = (V_D, E_D)$ and a sub-digraph $H = (V_H, E_H)$, a path $\pi$ in $D$ is a $H$-path if it is such that its startpoint and its endpoint are in $V_H$, no internal vertex is in $V_H$, and no edge of the path is in $E_H$. Moreover, a cycle $C$ in $D$ is a $H$-cycle if there is exactly one vertex of $C$ in $V_H$.

Definition 4.8. Let $D = (V_D, E_D)$ be a digraph and $L = [L_0, L_1, \ldots, L_n]$ an ordered sequence of sub-digraphs of $D$, where $L_i = (V_{L_i}, E_{L_i})$. We say that $L$ is:
1) a closed ear decomposition, if:
   - $L_0$ is a cycle,
   - for all $0 < i < r$, $L_i$ is a $D_i$-path or a $D_i$-cycle, where $D_i = (V_{D_i}, E_{D_i})$ with $V_{D_i} = \bigcup_{0 \leq j < i} V_{L_j}$ and $E_{D_i} = \bigcup_{0 \leq j < i} E_{L_j}$.
   - $V_D = \bigcup_{0 \leq j \leq r} V_{L_j}$, $E_D = \bigcup_{0 \leq j \leq r} E_{L_j}$
2) an open ear decomposition (oed), if it is a closed ear decomposition such that for all $0 < i \leq r$, $L_i$ is a $D_i$-path, i.e., it is not a $D_i$-cycle.

In Definition 4.8, each $L_i$ is called an ear. In particular, $L_0$ is the basic cycle and the other ears are derived ears. An ear is trivial if it has only one edge.

Definition 4.9. We say that an open ear decomposition of a strongly biconnected digraph is regular (r-oed) if the basic cycle $L_0$ has three or more vertices, and there exists a non-trivial derived ear with both ends attached to the basic cycle.

Figure 3: Digraph with an open ear decomposition.

Observation 4.10. Let $D = (V, E)$ be a digraph with an oed $L = [L_0, L_1, \ldots, L_n]$. For each pair $v, w \in V$, there exists a sequence of cycles $C = [C_1, \ldots, C_n]$ such that:
   - $v \in V_{C_1}$ and $w \in V_{C_n}$;
   - for all $j = 1, \ldots, n - 1$, $\exists a_j, b_j \in V_{C_j} \cap V_{C_{j+1}}$ such that $(a_j, b_j) \in E$.

Figure 3 shows a digraph with an oed $[L_0, L_1, L_2]$. The sequence of cycles associated to pair $v = 2$, $w = 10$ is $C = [C_0, C_2]$, where $C_0 = L_0$ and $C_2$ is the subgraph induced by $\{1, 8, 9, 10, 7, 4, 5\}$. Note that $(4, 5) \in C_0 \cap C_2$. The sequence associate to pair $v = 1$, $w = 6$ is simply $C = [C_1]$, where $C_1$ is the subgraph induced by $\{1, 2, 3, 6, 7, 4, 5\}$. In fact, nodes 1 and 6 belong to the same cycle.

We recall the following results, that characterize strongly biconnected and strongly connected digraphs:

Theorem 4.11. Let $D$ be a non-trivial digraph.
   - $D$ is strongly biconnected if and only if $D$ has an oed.
   - Any cycle can be the starting point of an oed [8].
• D is strongly biconnected if and only if exactly one of the following holds [5]:
  1) D is a partially-bidirectional cycle;
  2) D has a r-oed.

**Theorem 4.12.** [7] Let D be a non-trivial digraph. D is strongly connected if and only if D has a closed ear decomposition.

**Observation 4.13.** Roughly speaking, this last result means that a strongly connected digraph is composed of non-trivial strongly biconnected components connected by corridors, or articulation points. A corridor is a sequence of adjacent vertices \(u_1, \ldots, u_n\) such that \((u_i, u_{i+1}), (u_{i+1}, u_i) \in E\) for each \(i = 1, \ldots, n - 1\). For example, in Fig. 4 the subgraph induced by nodes 3, 5 and 6 is a corridor. Given a digraph \(D = (V, E)\), vertex \(v \in V\) is an articulation point if its removal increases the number of connected components of the underlying graph \(G(D)\). In Fig. 4 nodes 3, 6 and 11 are articulation points.

**A. Solving MAPF on strongly biconnected digraphs**

Reference [5] shows that all MAPF instances on strongly biconnected digraphs with at least two holes can be solved (or proven to be unsolvable) in polynomial time. It also presents Algorithm diBOX, that solves MAPF in the two possible cases of a partially-bidirectional cycle and of a digraph with a r-oed.

**Partially-bidirectional cycle.** This is the easy case. As no swapping between agents is possible, an instance is solvable if and only if the agents come in the right order in the first place. In this case, only one hole is needed in the digraph. Computing the solution can be performed by diBOX with a time complexity of \(O(|V_D|^2)\).

**Regular open-ear decomposition.** This is a more complex case.

**Proposition 4.14.** [5] Let D be a strongly biconnected digraph with a r-oed, with pebbles P and holes H, with \(|H| \geq 2\). For any configurations pair \(A^s, A^t\), there exists a plan \(f\) such that \(A^t(P) = \rho(A^s, f)(P)\) (i.e., all MAPF instances with at least two holes have a solution).

In particular, diBOX solves any MAPF instance with at least two holes, and finds a solution in \(O(|V_D|^3)\) time.

**V. PATH PLANNER FOR STRONGLY CONNECTED DIGRAPH**

As in literature, MAPF has been studied only on connected undirected graphs or on biconnected digraphs. In this section, we consider the more general case of strongly connected digraphs. In particular, we discuss the feasibility of MAPF and present an algorithm (diSC) to find solutions in polynomial time. For proofs of the main results please refer to [16]. We will need some results on the motion planning problem. We recall its definition.

**Definition 5.1.** Let \(D = (V, E)\) be a digraph, \(P\) a set of pebbles. Given a pebble \(p \in P\), an initial configuration \(A\), and \(v \in V\), the motion planning problem (MPP) consists in finding a plan \(f\) such that \(A = \rho(A, f)\) satisfies \(A(p) = v\). We indicate such a plan with notation \(A(p) \Rightarrow v\).

Reference [6] discusses the feasibility of the motion planning problem and proves the following:

**Theorem 5.2.** (Theorem 14 of [6]) Let D be a strongly biconnected digraph, \(P\) a set of pebbles and \(H\) a set of holes.

Then any MPP on D is feasible if and only if \(|H| \geq 1\).

For connected undirected graphs, in Section III, we mentioned that the feasibility of MAPF is decidable in linear time with respect to the number of nodes. Indeed, MAPF can be reduced to PMT. In the following, we show that the same result holds for strongly connected digraphs. In fact, it is possible to define a biconnected component tree \(T\) and a corresponding PMT problem such that MAPF on D is solvable if and only if PMT on \(T\) is solvable.

The biconnected component tree of a digraph D is the biconnected component tree \(T(G)\) of the underlying graph \(G = G(D)\). By Theorem 4.3 and Theorem 3.1, it follows that MAPF on D is feasible if and only if the corresponding PMT on \(T(G)\) is feasible.

**Figure 4:** Example of strongly connected digraph: the corresponding underlying graph is shown in Figure 1. Note that each star subgraph of \(T(G(D))\) represents a biconnected component of \(G(D)\), which corresponds to a strongly biconnected component of D. Indeed, Theorem 9 of [6] defines a one-to-one correspondence between strongly biconnected components of D and biconnected components of \(G(D)\). We will use the following definition adapted from [2]:

**Definition 5.3.** Let \(B = (V, E)\) be a strongly biconnected digraph and \(v \notin V\) be an external node. We consider a digraph \(G = (V \cup \{v\}, E)\) with \(E \subseteq E\). We say that G is:

- a strongly biconnected digraph with an entry-attached edge, if there exists \(z \in V\) such that \(E = \{(v, z)\} \cup E\);
- a strongly biconnected digraph with an attached edge, if there exists \(z \in V\) such that \(E = \{(v, z), (z, v)\} \cup E\).

First, we define some basic plans, that we will use to move pebbles and holes.

**BRING HOLE FROM \(v\) TO \(w\).** Let \(A\) be an initial configuration, such that \(v \in A(H)\) (i.e., \(v\) is an unoccupied vertex). Let \(\pi = u_1 = w, \ldots, u_n = v\) be a shortest path from w to v. We define the plan BRING HOLE FROM \(v\) TO \(w\) as:

\[
h_{v,w} = (u_{n-1} \rightarrow u_n, \ldots, u_1 \rightarrow u_2).
\]

In other words, for each \(j\) from \(n - 1\) to 1, if there is a pebble on \(u_j\), we move it on \(u_{j+1}\). The new configuration \(\bar{A}\) is defined as follows:

\[
\bar{A}(q) = \begin{cases} 
  u_{j+1}, & \text{if } A(q) = u_j \\
  w, & \text{if } A(q) = v; \\
  A(q), & \text{otherwise},
\end{cases}
\]

which means that only pebbles and holes along path \(\pi\) change positions.

**BRING BACK HOLE FROM \(w\) TO \(v\).** Let \(h_{v,w}\) be a plan BRING HOLE FROM \(v\) TO \(w\). Since the graph is strongly connected, by Proposition 4.2 there exists a reverse plan \(h_{w,v}^{-1}\), which returns pebbles and holes to their initial positions. We call BRING BACK HOLE FROM \(w\) TO \(v\) the plan \(h_{v,w}^{-1}\).

**BRING HOLE FROM \(v\) TO A SUCCESSOR OF \(w\).** Let \(A\) be an initial configuration, such that \(v \in A(H)\). Let \(\pi = u_1 = w, \ldots, u_n = v\) be a shortest path from \(w\) to \(v\), where \(u_2\) is the successor of \(w\) along \(\pi\). Then, BRING HOLE FROM \(v\)
Lemma 5.6. Entry Lemma. Let $P$ be a set of pebbles and $H$, with $|H| \geq 2$, a set of holes on $G = (V \cup \{v\}, E)$, where $G$ is a strongly biconnected digraph with an entry-attached edge $(v, y)$ (see Definition 5.3). Let $A$ be a configuration, $p \in P$ such that $A(p) = v$, and $w \in A(H)$. Let $A[v, w]$ be the configuration defined in (1). Then, there exists a plan $f_{vw}$ such that $A[v, w] = \rho(A, f_{vw})$, i.e., that moves $p$ from $v$ to $w$, without altering the locations of the other pebbles. In particular, we can write this plan as

$$f_{vw} = h_{hs[w]}(R^*_h(v \to y))(R^*_h)^{ch_h^{-1}}w,$$

where $C = [C_1, \ldots, C_n]$ is a sequence of cycles, $h$ a hole, and $k \in \mathbb{N}^n$.

Figure 5: $f_{vw} = r_2C_1r_3C_2r_1C_3(v \to y)r_3C_3r_2C_2r_1C_1$.

Lemma 5.7. Stay in Lemma. Let $P$ be a set of pebbles and $H$, with $|H| \geq 2$, a set of holes on $D = (V,E)$, a strongly biconnected digraph with a $r$-oed. Let $A$ be a configuration, $p \in P$ such that $A(p) = v$, and $w \in A(H)$. Let $A[v, w]$ be a configuration defined as in (1), then there exists a plan $f_{vw}$ such that $A[v, w] = \rho(A, f_{vw})$.

Lemma 5.8. Attached-Edge. Let $P$ be a set of pebbles and $H$, with $|H| \geq 2$, a set of holes on $D = (V \cup \{v\}, E)$, a strongly biconnected digraph with an attached edge such that $|H| \geq 2$. Let $A$ be a configuration, $p \in P$ such that $A(p) = u$, and $v \in A(H)$. Let $A[u, v]$ be a final configuration defined as in (1), then there exists a plan $f_{uw}$ such that $A[u, v] = \rho(A, f_{uw})$.

Next lemma deals with the case of two biconnected components joined by an articulation point, like, e.g., $\{6, 7, 8, 9, 10, 11\} \text{ and } \{11, 12, 13\}$ in Figure 4, where the articulation point is node 11.

Figure 6: Cycle with an attached-edge.

Lemma 5.9. Two Biconnected Components. Let $P$ be a set of pebbles and $H$, with $|H| \geq 2$, a set of holes on $D = (V,E)$, a strongly biconnected digraph, composed of two biconnected components joined by an articulation point. Let $A$ be a configuration, $p \in P$ be such that $A(p) = a$, and $b \in A(H)$. Let $A[a, b]$ be a final configuration defined as in (1). Then, there exists a plan $f_{ab}$ such that $A[a, b] = \rho(A, f_{ab})$.

These results allow us to prove that feasibility of a MAPF on a strongly connected digraph with at least two holes is equivalent to feasibility of the corresponding PMT. This is a consequence of the following theorem, adapted from a similar theorem on undirected graphs (see 2).

Theorem 5.10. Let $P$ be a set of pebbles and $H$ a set of
holes on a strongly connected digraph $D = (V, E)$, which is not a partially-bidirectional cycle, and let $T = (V_T, E_T)$ be the corresponding biconnected component tree. Let $A$ be an initial configuration on $D$ and $\tilde{A}$ the corresponding configuration on $T$. Let $a, b \in V$ and $p \in P$ be a pebble on $a$. Then, if $|H| \geq 2$, there is a plan $f_{ab}$ on $D$ such that $\mathcal{A}[a, b] = \rho(\mathcal{A}, f_{ab})$ if and only if there is a plan $f'_{ab}$ on $T$ such that $\mathcal{A}[a, b] = \rho(\mathcal{A}, f'_{ab})$.

A. Algorithm diSC

The idea of this algorithm is to use the same strategy to solve MAPF on undirected graphs, described in Section III. The main steps are the following ones:
1) Convert the digraph $D$ into a tree $T$ and consider the corresponding PMT problem.
2) Convert the PMT problem into a PPT one and solve it.
3) Convert solution plans on $T$ into plans on $D$ by a function CONVERT-PATH, which is based on Theorem 5.10.

Theorem 5.11. diSC finds the solution of a MAPF instance with at least two holes on $D = (V, E)$ in polynomial time with respect to $|V|$.

Proof. Convert $D$ into $T(D)$ takes $O(|V|)$ time [14]. Solve PMT takes polynomial time with respect to $|V|$ [3]. CONVERT-PATH uses diBOX to move pebbles within a strongly biconnected component of the digraph, which takes $O(|V|^3)$ time [5].

VI. EXPERIMENTAL RESULTS

We implemented the diSC algorithm in Matlab. To evaluate its behaviour, we generated random graphs with a number of nodes that ranges from 20 to 100, with increments of 5 nodes. For every number of nodes, we generated a set of 200 graphs. In doing so we used a procedure in order to generate test graphs with multiple biconnected components. First, we ran the algorithm varying the number of nodes: for every generated graph (200 for every different number of nodes), we created a MAPF problem instance, with 10 agents and random initial and final positions. We then ran the algorithm for MAPF problem instances on a 397 nodes graph associated to the layout a real warehouse, varying the number of agents from 1 to 10. We used an Intel(R) Core(TM) i7-4510U CPU @ 2.60 GHz processor with 16 GB of RAM. For each obtained solution, we recorded the overall number of moves and the computation time.

Fig. 7 shows the medians of the number of steps and computational time as a function of the number of nodes. Roughly, both increase quadratically. In these figures, the trendlines are the least squares approximations with second or third order polynomials.

Fig. 8 shows the medians of the number of steps and computational time as a function of the number of agents in a warehouse. Also in this case both the number of steps and the running time seem to be increasing in a polynomial way.

Thus, our simulations confirm the complexity result presented in Theorem 5.11.

VII. CONCLUSIONS AND FUTURE WORK

We proved that the feasibility of MAPF problems on strongly connected digraphs is decidable in linear time (Theorem 4.3). Moreover, we show that a MAPF problem on a strongly connected digraph is feasible if and only if the corresponding PMT problem on the biconnected component tree is feasible (Corollary 4.4). Finally, we presented an algorithm (diSC) for solving MAPF problems on strongly connected digraphs in polynomial time with respect to the number of both nodes and agents (Theorem 5.11). As already said, diSC algorithm finds a solution that has often a much larger number of steps than the shortest one. Our next step will be to shorten the solution, for instance by a local search.

REFERENCES

[1] V. Auletta, A. Monti, P. Persiano, M. Parente, A Linear Time Algorithm for the Feasibility of the Pebble Motion on Trees, Algorithmica 23(3): 223-245 (1999)
[2] G. Goraly, R. Hassin, Multi-Color Pebble Motion on Graphs, Algorithmica, 58(3): 93-106 (2010)
[3] M. Khorshid , R.C. Holte, N.R. Sturtevan, A polynomial-time algorithm for non-optimal multi-agent path finding, The Fourth Annual Symposium on Combinatorial Search, 76-83 (2011)
[4] A. Krontiris, R. Luna, K.E. Bekris, From Feasibility Tests to Path Planners for Multi-Agent Pathfinding, Symposium on Combinatorial Search (SoCS) (2013)
[5] A. Botea, P. Surynek, Multi-agent path finding on strongly biconnected digraphs, Journal of Artificial Intelligence Research 62:273–314 (2018)
[6] Z. Wu, S. Grumbach, Feasibility of motion planning on acyclic and strongly connected directed graphs, manuscript (2008)
[7] J. Bang-Jensen, G. Gutin, Digraph Theory. Algorithms and Applications, Springer Monographs in Mathematics (2000)
[8] D. B. West, Introduction to Graph Theory, 2nd ed., Prentice Hall (2000)
[9] J. Yu, S. M. LaValle, Structure and Intractability of optimal multi-robot path planning on graphs, AAAI (2013)
[10] G. Sharon, R. Stern, A. Felner, N.R. Sturtevant, Conflict-based search for optimal multi-agent pathfinding, Artificial Intelligence 219 40-46 (2015)
[11] D. Silver, Cooperative pathfinding, Artificial Intelligence and Interactive Digital Entertainment (AIIDE) 117-122 (2005)
[12] B. de Wilde, A. W. ter Mors, C. Witteveen, Push and rotate: cooperative multi-agent path planning, AAMAS 87-94 (2013)
[13] E. T. S. Alotaibi and H. Al-Rawi, Push and spin: A complete multi-robot path planning algorithm, 2016 14th International Conference on Control, Automation, Robotics and Vision (ICARCV), 2016, pp. 1-8, doi: 10.1109/ICARCV.2016.7838836.
[14] M. H. Karaata, A Stabilizing Algorithm for Finding Bi-connected Components, Journal of Parallel and Distributed Computing 62 982-999 (2002)
[15] B. Nebel, On the Computational Complexity of Multi-Agent Pathfinding on Directed Graphs, Proceedings of the International Conference on Automated Planning and Scheduling, vol. 30, no. 1 212–216 (2020)
[16] S. Ardizzone, I. Saccani, L. Consolini, M. Locatelli, Multi-Agent Pathfinding on Strongly Connected Digraphs: Feasibility and Solution Algorithms, Arxiv (2022)