Discrete Applied Mathematics 158 (2010) 1017–1028

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

Feasibility of motion planning on acyclic and strongly connected directed graphs

Zhilin Wu[a,*], Stéphane Grumbach[b]

[a] CASIA-LIAMA, Zhongguancun East Road #95, 100190, Beijing, China
[b] INRIA-LIAMA, Zhongguancun East Road #95, 100190, Beijing, China

A R T I C L E   I N F O

Article history:
Received 31 July 2008
Received in revised form 17 July 2009
Accepted 1 February 2010
Available online 7 March 2010

Keywords:
Motion planning
Feasibility
Acyclic directed graphs
Strongly connected directed graphs

A B S T R A C T

Motion planning is a fundamental problem of robotics with applications in many areas of computer science and beyond. Its restriction to graphs has been investigated in the literature, for it allows one to concentrate on the combinatorial problem abstracting from geometric considerations. In this paper, we consider motion planning over directed graphs, which are of interest for asymmetric communication networks. Directed graphs generalize undirected graphs, while introducing a new source of complexity to the motion planning problem: moves are not reversible. We first consider the class of acyclic directed graphs and show that the feasibility can be solved in time linear in the product of the number of vertices and the number of arcs. We then turn to strongly connected directed graphs. We first prove a structural theorem for decomposing strongly connected directed graphs into strongly biconnected components. Based on the structural decomposition, we show that the feasibility of motion planning on strongly connected directed graphs can be decided in linear time.

1. Introduction

Motion planning is a fundamental problem of robotics. It has been extensively studied [10], and has numerous practical applications beyond robotics, such as in manufacturing, animation, games [12] as well as in computational biology [15,5]. The complexity of motion planning, which is intrinsically PSPACE-hard [9,10], has received a lot of attention. The study of motion planning on graphs was proposed by Papadimitriou et al. [14] to strip away the geometric considerations of the general motion planning problem and concentrate on the combinatorial problem.

In this paper, we consider the feasibility of motion planning over directed graphs. Our results generalize results on undirected graphs, which can be shown as a subclass of directed graphs. Directed graphs are of great importance in several fields such as communication networks, which are frequently asymmetric [7]. But technically, directed graphs differ from undirected graphs, for movements in the graph are not reversible.

Papadimitriou et al. [14] first introduced the problem of motion planning on graphs. They defined the Graph Motion Planning with 1 Robot problem (GMP1R) as follows: Suppose we are given a graph $G = (V, E)$ with n vertices, there is one robot in a vertex s and some of the other vertices contain a movable obstacle. The objective of GMP1R is to move the robot from the source vertex s to a destination vertex t with the smallest number of moves, where a move consists in moving a robot or an obstacle from one vertex to an adjacent vertex that does not contain an object (robot or obstacle). It may be impossible to move the robot from s to t, for instance, if all the vertices other than s are occupied by obstacles. The feasibility

[*] Corresponding address: Universite Bordeaux, LaBRI, 351 Cours de la liberation, F-33405 Talence cedex, France. Tel.: +33 0 5 40 00 69 08; fax: +33 0 5 40 00 66 69.
E-mail addresses: zlwu@liama.ia.ac.cn, zlwu@labri.fr (Z. Wu), stephane.grumbach@inria.fr (S. Grumbach).

0166-218X/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2010.02.001

problem of GMP1R is to decide whether it is possible or not to move the robot from the source vertex to the destination vertex.

In [14], it was shown that the feasibility of GMP1R can be decided in polynomial time, and the optimization of GMP1R is NP-complete (even on planar graphs). They also gave a $O(n^6)$ exact algorithm as well as a fast 7-approximation algorithm for GMP1R on trees, a $O(\sqrt{n})$-approximation algorithm for GMP1R on general graphs. Auletta et al. proposed more efficient algorithms for the feasibility and optimization of GMP1R on trees in [1,2].

Motion planning on graphs has wide practical applications. Track transportation system [13] constitutes a typical example: Vehicles move on a system of tracks such that each track connects two distinct stations. There is a distinguished vehicle which moves from a source station to a destination station. There are other vehicles (obstacles) on the non-source stations. The vehicles are only able to stop at the stations and not able to stop in the middle of tracks. They coordinate with each other to let the distinguished vehicle move from the source station to the destination station. A variant of the previous example is packet transfer in communication buffers. Graphs are regarded as (bidirectional) communication networks and objects as indivisible packets of data. If there is a distinguished packet which moves from a source node to a destination node, and there are already some other packets stored in the communication buffers of nodes in the network, the objective is to move the distinguished packet from the source node to the destination node without exceeding the capacities of the communication buffers of each node.

In practice, in the two previous examples, the tracks (links) between the stations (nodes) might be asymmetric. This motivates the study of motion planning on directed versus undirected graphs.

Let us consider first the track transportation system. Some of the tracks may be unidirectional. For instance, if the two stations are not at the same altitude, the track connecting them might be too steep, and the vehicles not strong enough to climb up the track. There may also be unidirectional tracks as a result of security considerations. The vehicle movement from a source station to a destination station, on a track-transportation system containing unidirectional tracks, leads to motion planning on directed graphs.

Now consider the packet transfer in communication networks. There might be unidirectional links in communication networks. For instance, in wireless ad hoc networks, unidirectional links can result from factors such as heterogeneity of receiver and transmitter hardware, power control algorithms, or topology control algorithms. Unidirectional links may also result from interference around a node that prevents it from receiving packets even though the other nodes are able to receive packets from it [11,7]. Networks with unidirectional links can be modeled as directed graphs. The problem of transferring a distinguished packet in networks with unidirectional links without exceeding the capacities of the communication buffers amounts to solving motion planning on directed graphs.

Directed graphs generalize undirected graphs, while introducing a new source of complexity to the motion planning problem: moves are not reversible, and motion planning might become infeasible after inappropriate moves.

In this paper, we first give a motivating example to illustrate that motion planning on directed graphs is much more intricate than motion planning on graphs. Then, we consider the class of acyclic directed graphs. We show that the feasibility of motion planning on acyclic directed graphs can be decided in time linear in the product of the number of vertices and the number of arcs (Theorem 3). We then turn to strongly connected directed graphs. We first consider their structure. We start with a subclass of them, strongly biconnected directed graphs. We obtain an interesting characterization of strongly biconnected directed graphs by showing that a directed graph is strongly biconnected iff it has an open ear decomposition (Theorem 8). This characterization can be seen as a generalization of the classical open-ear-decomposition characterization of biconnected graphs. We then prove a structural theorem for decomposing strongly connected directed graphs into strongly biconnected components (Theorem 9). Based on the open-ear-decomposition characterization, we show that motion planning on strongly biconnected directed graphs is feasible iff there is at least one vertex occupied neither by robot nor by obstacle (Theorem 14). With the structural decomposition, we show that the feasibility of motion planning on strongly connected directed graphs can be decided in linear time (Theorem 21).

The paper is organized as follows. A motivating example is presented in the next section. In Section 3, we recall classical definitions from graph theory. We consider acyclic directed graphs in Section 4, and give an algorithm to decide the feasibility of motion planning on such a class of directed graphs. In Section 5, we consider strongly biconnected directed graphs, and prove a structural theorem on their decomposition into strongly biconnected components. The feasibility for strongly connected directed graphs is considered in Section 6.

2. A motivating example

In the sequel, for brevity, we use "digraph" to denote "directed graph".

Let us consider a simple example to illustrate the motion planning on digraphs. Vertices can contain either an object (obstacle or the robot) or nothing. If there is no object on a vertex, we say that there is a *hole* in that vertex. For an arc (v, w) from v to w, with an object on v, and a hole on w, the object can be moved from v to w, and we say equivalently that the hole can be moved (backwards) from w to v.

Consider the strongly connected component C in the graph of Fig. 1 which contains vertices $v_1, \dots, v_5$, s and t. The initial positions of the robot and obstacles are shown in Fig. 1(a).

We can move the robot from s to t as follows: Move the hole in v_1 to v_2, move the robot from s to v_2, then move the two holes in v_7, v_8 into C through s, without moving the robot in v_2 (Fig. 1(b)). Now move the obstacle in v_4 to v_5, and move the

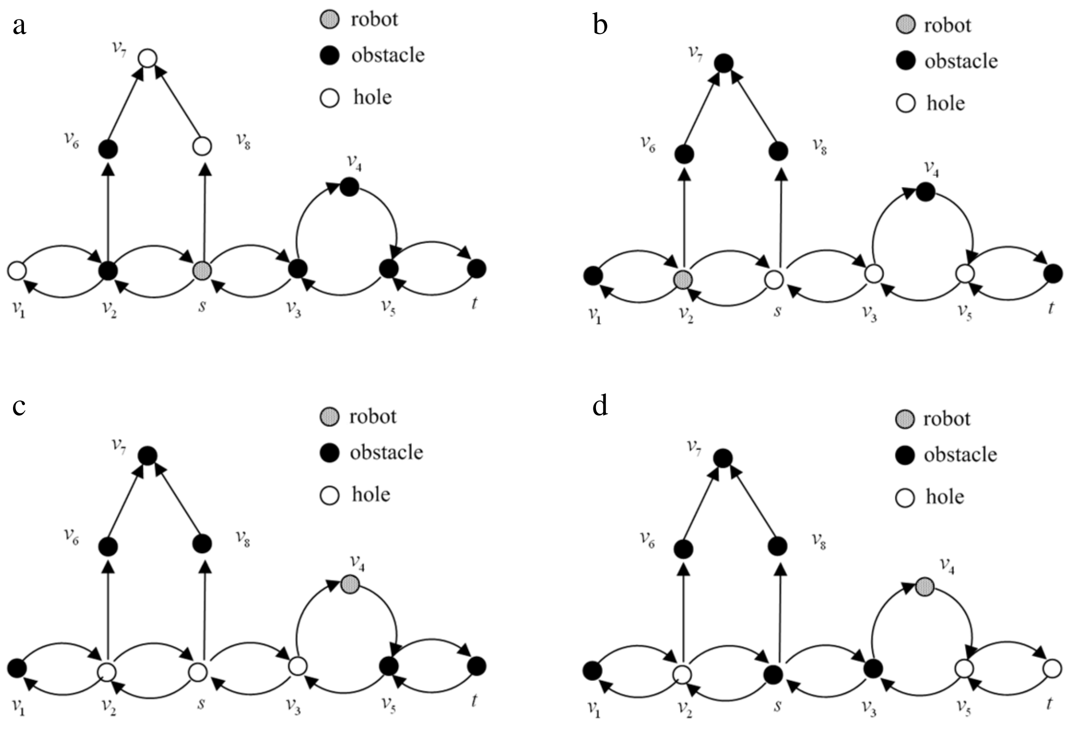

Fig. 1. Motion planning on digraphs.

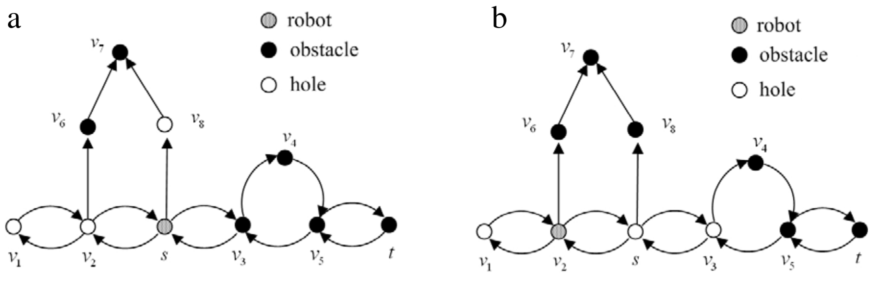

Fig. 2. Object moves that do not preserve feasibility.

robot to v_4 (see Fig. 1(c)). Move the two obstacles in v_5 and t to v_3 and s (Fig. 1(d)), then move the robot from v_4 to v_5, and finally to t. The main idea of these moves is to move the robot to v_4 in order to free the way for the moves of the holes from s and v_3 to v_5 and t.

If the robot is in s and we move the hole in v_7 to v_2 (Fig. 2(a)), then the problem becomes infeasible. We can move the robot from s to v_2 and the hole in v_8 to s (Fig. 2(b)), but it is then impossible to move the robot from v_2 to v_4.

As illustrated in the above example, the intricacy of motion planning on digraphs follows from the non-reversibility of moves in the digraphs.

In the sequel, we propose algorithms which take as input, a digraph $D = (V, E)$ encoded by its adjacency lists, a source and destination vertex $s, t \in V$, a function f mapping each vertex to an element of the set {"robot","obstacle", "hole"}, and produces a Boolean value (true or false) indicating whether it is feasible to move the robot from s to t in D.

3. Preliminaries

A *digraph* D is a binary tuple (V, E) such that $E \subseteq V^2$. Elements of V and E are called respectively *vertices* and *arcs* of D. We assume that $(v, v) \notin E$ for all $v \in V$ (there are no self-loops).

For a vertex v of a digraph $D = (V, E)$, the *indegree* of v, denoted $in(v)$, is defined as $|\{w \in V | (w, v) \in E\}|$, and the *outdegree* of v, denoted $out(v)$, is defined as $|\{w \in V | (v, w) \in E\}|$.

A *graph* G is a binary tuple (V, E) such that $E \subseteq V^{[2]}$, where $V^{[2]}$ contains exactly all two-element subsets of V, namely $V^{[2]} = \{\{v, w\} | v, w \in V, v \neq w\}$. Elements of E are called *edges* of G.

For a vertex v of a graph $G = (V, E)$, the *degree* of v, denoted $deg(v)$, is defined as $|\{w \in V | \{v, w\} \in E\}|$.

If $D = (V, E)$ (resp. $G = (V, E)$), and $e = (v, w) \in E$ (resp. $e = \{v, w\} \in E$), then e is said to be *incident* to v and w in D (resp. G).

A digraph (resp. graph) containing exactly one vertex is said to be *trivial*, otherwise it is said to be *nontrivial*.

Given a digraph $D = (V, E)$ (resp. graph $G = (V, E)$), the digraph (resp. graph) $H = (V_H, E_H)$ such that $V_H \subseteq V$ and $E_H \subseteq E$ is called a *sub-digraph* of D (resp. *subgraph* of G). Let $X \subseteq V$, the sub-digraph (resp. subgraph) *induced by* X, denoted $D[X]$ (resp. $G[X]$), is the sub-digraph (resp. subgraph) $(X, E \cap X \times X)$ (resp. $(X, E \cap X^{[2]})$).

Suppose $D = (V, E)$ (resp. $G = (V, E)$) is a digraph (resp. graph) and $X \subseteq V$, let $D - X$ (resp. $G - X$) denote the digraph (resp. graph) obtained from D (resp. G) by deleting all the vertices in X and all the arcs (resp. edges) incident to at least one element of X. If $X = \{v\}$, then $D - \{v\}$ (resp. $G - \{v\}$) is written as $D - v$ (resp. $G - v$) for simplicity.

Given a digraph $D = (V, E)$, the *underlying graph* of D, denoted by $\mathcal{G}(D)$, is the graph obtained from D by omitting the directions of arcs, namely $\mathcal{G}(D) = (V, \{\{v, w\} | (v, w) \in E\})$.

A *path* of a digraph $D = (V, E)$ (resp. graph $G = (V, E)$) is an alternating sequence of vertices and arcs (resp. edges) $v_0 e_1 v_1 \dots v_{k-1} e_k v_k$ ($k \geq 1$) such that for all $1 \leq i \leq k$, $e_i = (v_{i-1}, v_i) \in E$ (resp. $e_i = \{v_{i-1}, v_i\} \in E$), and for all $0 \leq i < j \leq k$, $v_i \neq v_j$. v_0 and v_k are called the *tail* and *head endpoint* of the path respectively, and the other vertices are called the *internal vertices* of the path. In particular, an arc or an edge is a path without internal vertices.

A *cycle* of a digraph $D = (V, E)$ is a sequence of vertices $v_0 v_1 \dots v_k$ such that for all $0 \leq i \leq k$, $(v_i, v_{i+1}) \in E$ (v_{k+1} interpreted as v_0), and for all $0 \leq i < j \leq k$, $v_i \neq v_j$. Cycles of graphs can be defined similarly, but we have the additional restriction that $k \geq 2$. So cycles of graphs contain at least three vertices.

A digraph D is *acyclic* if there are no cycles in D.

Suppose $H = (V_H, E_H)$ is a sub-digraph of $D = (V, E)$ (resp. subgraph of $G = (V, E)$). A path P of D (resp. G) is an *H-path* if the two endpoints of P are in H, no internal vertices of P are in H, and no arcs (resp. edges) of P are in H. In particular, an arc $(v, w) \in E \setminus E_H$ (resp. an edge $\{v, w\} \in E \setminus E_H$) with $v, w \in V_H$ is an H-path. A cycle C is an *H-cycle* if there is exactly one vertex of C in H.

Let $H_1 = (V_1, E_1)$ and $H_2 = (V_2, E_2)$ be two sub-digraphs of a digraph $D = (V, E)$, then the *union* of H_1 and H_2, $H_1 \cup H_2$, is defined as $(V_1 \cup V_2, E_1 \cup E_2)$. The union of subgraphs can be defined similarly.

A digraph $D = (V, E)$ is *strongly connected* if for any two distinct vertices v and w, there are both a path from v to w and a path from w to v in D. The digraph containing exactly one vertex and no arcs is the minimal strongly connected digraph.

Let $D = (V, E)$ be a digraph. The *strongly connected components* of D are the maximal strongly connected sub-digraphs of D.

A graph $G = (V, E)$ is *connected* if, for any two distinct vertices v and w of G, there is a path of G with endpoint v and w. The *connected components* of a graph G are the maximal connected subgraphs of G.

If $G = (V, E)$ is a graph, $v \in V$, and the number of connected components of $G - v$ is more than that of G, then v is said to be a *cut vertex* of G.

A graph G is *biconnected* if G is connected and there are no cut vertices in G. In particular, the graph containing exactly one vertex is the minimal biconnected graph. The *biconnected components* of a graph G are the maximal biconnected subgraphs of G.

Without loss of generality, we assume that for each digraph D, (i) the underlying graph of D, $\mathcal{G}(D)$, is connected, (ii) the source vertex s and the destination vertex t are distinct (thus all the digraphs considered from now on are nontrivial), (iii) there is at least one path from s to t in D.

4. Motion planning on acyclic digraphs

In this section we assume that $D = (V, E)$ is an acyclic digraph.

We first recall a result about topological orderings of acyclic digraphs.

A *topological ordering* of an acyclic digraph $D = (V, E)$ is an ordering of all vertices of D, say $v_1, \dots, v_k$, such that $(v_i, v_j) \in E$ implies $i < j$. From [3], we know that a topological ordering of a given acyclic digraph can be computed in linear time by a depth-first-search.[1]

Theorem 1 ([3]). *Given an acyclic digraph $D = (V, E)$, a topological ordering of D can be computed in time $O(n + m)$, where n is the number of vertices and m is the number of arcs of D.*

We introduce some notations in the following.

Let V' denote the set of vertices from which there is a path to t, and to which there is a path from s. In particular, $s, t \in V'$.

For each $v \in V'$, let $h(v)$ denote the number of holes that can be moved to v.

For each $v \in V'$, define $h_t(v)$ as follows: Suppose that the robot is in v.

- If the robot can be moved from v to t in D, then there may be different paths (from v to t) along which the robot can be moved from v to t, let $h_t(v)$ be the minimal length (number of arcs) of such paths.

[1] In [3], topological orderings are called acyclic orderings.

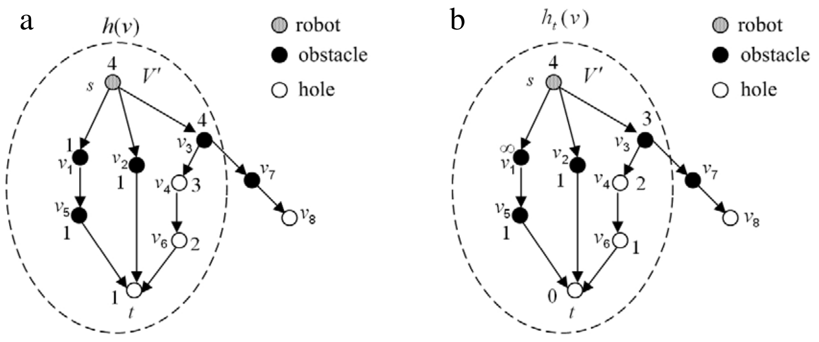

Fig. 3. Computation of $h(v)$ and $h_t(v)$.

- If it is impossible to move the robot from v to t, let $h_t(v) = \infty$.

The $h(v)$'s can be computed by solving the reachability problems. The $h_t(v)$'s can be computed as follows:

(i) Compute a topological ordering of $D[V']$, say $v_1, \ldots, v_k$, such that $v_1 = s$, $v_k = t$;
(ii) Compute $h_t(v)$'s by a backward induction,
  - $h_t(v_k) := 0$,
  - For $i < k$: If $\exists j : i < j \le k$ such that $(v_i, v_j) \in E$ and $h(v_j) \ge h_t(v_j) + 1$, then $h_t(v_i) := \min\{h_t(v_j) + 1 | i < j \le k, (v_i, v_j) \in E, h(v_j) \ge h_t(v_j) + 1\}$. Otherwise, $h_t(v_i) := \infty$.

The problem is feasible iff $h_t(s) < \infty$, which is justified by the argument in the following two paragraphs.

If the problem is feasible, the robot can be moved from s to t. Let P be the trace of the robot during this movement (namely the sequence of nodes and arcs reached by the robot). As a result of acyclicity of D, P is a path of D. Let $P = v_0e_1v_1 \cdots v_{k-1}e_kv_k$ be such that $v_0 = s$ and $v_k = t$. During the movement, when the robot is moved to $v_{i-1} (1 \le i \le k)$, in order to move the robot from v_{i-1} to v_i, a hole should be moved to v_i. Since D is acyclic, the hole moved to v_i cannot be moved to v_j for any j such that $i < j \le k$ (holes are moved along the reverse direction of arcs). So these holes moved to the v_i's are distinct from each other, and can be moved to occupy all the vertices on P except s. By induction, we can show that for all vertices v on P, the inductively computed $h_t(v)$ satisfies that $h_t(v) < \infty$.

On the other hand, if $h_t(s) < \infty$, by induction, we can show that there is a path P from s to t such that for each vertex $v \ne s$ on P, we have $h_t(v) < \infty$ and $h(v) \ge h_t(v) + 1$, and for each arc (v, w) on P, $h_t(v) = h_t(w) + 1$. By induction again, we can show that the holes in D can be moved to occupy all the vertices on P except s. Then the robot can be moved to t along P, and the problem is feasible.

Example 2. The computation of $h(v)$ and $h_t(v)$ on an acyclic digraph is illustrated in Fig. 3. For instance, $h_t(t) = 0$, $h_t(v_5) = h_t(t) + 1 = 1$, since $(v_5, t) \in E$ and $h(t) = 1 \ge h_t(t) + 1 = 1$; on the other hand, $h_t(v_1) = \infty$, since v_5 is the only successor of v_1, but $h(v_5) = 1 < h_t(v_5) + 1 = 2$.

Theorem 3. *Feasibility of motion planning on acyclic digraphs can be decided in $O(nm)$ time, where n is the number of vertices, and m is the number of arcs.*

Proof. Let D be an acyclic digraph, n and m be the number of vertices and number of arcs of D respectively.

The computation of V' takes $O(m)$ time since it can be done by solving the reachability problem twice.

The computation of $h(v)$'s takes $O(nm)$ time because the computation of each $h(v)$ takes $O(m)$ time and there are at most $O(n)$ such computations.

The computation of a topological ordering of $D[V']$ takes $O(n + m)$ time from Theorem 1.

The computation of $h_t(v)$'s takes $O\left(\sum_{v \in V'} out(v)\right) = O(m)$ time.

Since $n \le m$, we conclude that the overall time complexity is $O(m + nm + n + m + m) = O(nm)$. □

5. Structure of strongly connected digraphs

In this section, we consider the structure of strongly connected digraphs. We first recall some definitions and theorems.

An *open ear decomposition* of a digraph $D = (V, E)$ (resp. graph $G = (V, E)$) is a sequence of sub-digraphs of D (resp. subgraphs of G), say $P_0, \ldots, P_r$, such that

- P_0 is a cycle;
- P_{i+1} is a D_i-path (resp. G_i-path), where D_i (resp. G_i) is $\bigcup_{0 \le j \le i} P_j$ for all $0 \le i < r$;
- $D = \bigcup_{0 \le i \le r} P_i$ (resp. $G = \bigcup_{0 \le i \le r} P_i$).

A *closed ear decomposition* of a digraph $D = (V, E)$ (resp. graph $G = (V, E)$) is a sequence of sub-digraphs of D (resp. subgraphs of G), say $P_0, \ldots, P_r$, such that

- P_0 is a cycle;
- P_{i+1} is a D_i-path or a D_i-cycle (resp. a G_i-path or a G_i-cycle), where D_i (resp. G_i) is $\bigcup_{0 \le j \le i} P_j$ for all $0 \le i < r$;
- $D = \bigcup_{0 \le i \le r} P_i$ (resp. $G = \bigcup_{0 \le i \le r} P_i$).

Theorem 4 (*[16]*). *Let G be a graph containing at least three vertices. G is biconnected iff G has an open ear decomposition. Moreover, any cycle can be the starting point of an open ear decomposition.*

Theorem 5 (*[3]*). *Let D be a nontrivial digraph. D is strongly connected iff D has a closed ear decomposition. Moreover, any cycle can be the starting point of a closed ear decomposition.*

Let $G = (V, E)$ be a graph. The *biconnected-component graph* of G, denoted $\mathcal{G}_{bc}(G)$, is a bipartite graph (V_{bc}, W_{bc}, E_{bc}) defined by

- V_{bc}: biconnected components of G;
- W_{bc}: vertices of G shared by at least two distinct biconnected components of G;
- E_{bc}: let $B \in V_{bc}$ and $w \in W_{bc}$, then $(B, w) \in E_{bc}$ iff $w \in V(B)$.

Theorem 6 (*[16]*). *Let $G = (V, E)$ be a connected graph. Then $\mathcal{G}_{bc}(G)$ is a tree.*

Now we introduce a class of digraphs, strongly biconnected digraphs.

Definition 7. Let D be a digraph. D is said to be strongly biconnected if D is strongly connected and $\mathcal{G}(D)$ is biconnected. The strongly biconnected components of D are the maximal strongly biconnected sub-digraphs of D.

In particular, the digraph containing exactly one vertex and no arcs is strongly biconnected.

The following result demonstrates that strongly biconnected digraphs also admit a similar characterization.

Theorem 8. *Let D be a nontrivial digraph. D is strongly biconnected iff D has an open ear decomposition. Moreover, any cycle can be the starting point of an open ear decomposition.*

Theorem 8 can be proved along the same line as Theorem 7.2.2 and Corollary 7.2.7 in [3].

We can prove the following structural theorem for strongly connected digraphs.

Theorem 9. *Let $D = (V, E)$ be a strongly connected digraph. Then the strongly biconnected components of D are those $D[V(B)]$, namely the sub-digraph of D induced by $V(B)$, where B is a biconnected component of $\mathcal{G}(D)$.*

Proof. Let $D = (V, E)$ be a strongly connected digraph.

If D is trivial, then the result is obvious.

Otherwise, D is nontrivial, let B be a biconnected component of $\mathcal{G}(D)$.

It is sufficient to show that $D[V(B)]$ is strongly connected. If this holds, then $D[V(B)]$ is strongly biconnected. Because all the vertices of a strongly biconnected sub-digraph of D are in some biconnected component of $\mathcal{G}(D)$ and B is a biconnected component of $\mathcal{G}(D)$, $D[V(B)]$ is a maximal strongly biconnected sub-digraph of D, i.e. a strongly biconnected component of D. Since the union of all biconnected components of $\mathcal{G}(D)$ is $\mathcal{G}(D)$ itself, the theorem holds.

Now we show that $D[V(B)]$ is strongly connected.

Let $v, w \in V(B)$ such that $v \neq w$. Since D is strongly connected, there must be a path P from v to w in D. Now we show that P is in $D[V(B)]$ as a matter of fact.

To the contrary, suppose that there is a vertex on P not in $D[V(B)]$.

Let v' be the first vertex on P (starting from v) not in $D[V(B)]$. Then there is $w' \in V(B)$ on P such that $(w', v') \in E$. Because B is a biconnected component of $\mathcal{G}(D)$, and two distinct biconnected components contain at most one vertex in common according to Theorem 6, it follows that v' is in a biconnected component $B' \neq B$ of $\mathcal{G}(D)$, and w' is the unique vertex shared by B' and B. Since P is a path, we have that $w' \neq w$, otherwise we have reached w before v' on P, a contradiction. Because $w \in V(B)$ and $w \neq w'$, we have that $w \in V(B) \setminus V(B')$. Since w' is the unique vertex shared by B and B', any path from v' to $w \in V(B) \setminus V(B')$ has to visit w', so P must visit w' again after visiting v', contradicting the fact that P is a path and there should be no vertices visited twice on a path.

Consequently for any $v, w \in V(B)$, $v \neq w$, there is a path in $D[V(B)]$ from v to w, $D[V(B)]$ is strongly connected. □

From Theorem 9, we have the following definition for strongly-biconnected-component graph of a strongly connected digraph.

Definition 10. Let D be a strongly connected digraph, the strongly-biconnected-component graph of D, denoted $\mathcal{G}_{sbc}(D) = (V_{sbc}, W_{sbc}, E_{sbc})$, is $\mathcal{G}_{bc}(\mathcal{G}(D))$, namely the biconnected-component graph of the underlying graph of D.

From the above definition and Theorem 6, we have the following corollary.

Corollary 11. *Let D be a strongly connected digraph. Then $\mathcal{G}_{sbc}(D)$ is a tree.*

Example 12 (*Strongly-biconnected-component Graph*). A strongly connected digraph D (Fig. 4(a)) and its strongly-biconnected-component graph $\mathcal{G}_{sbc}(D)$ (Fig. 4(b)).

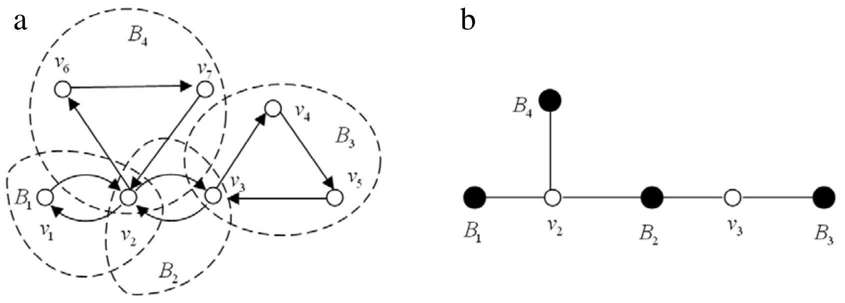

Fig. 4. Example: strongly-biconnected-component graph.

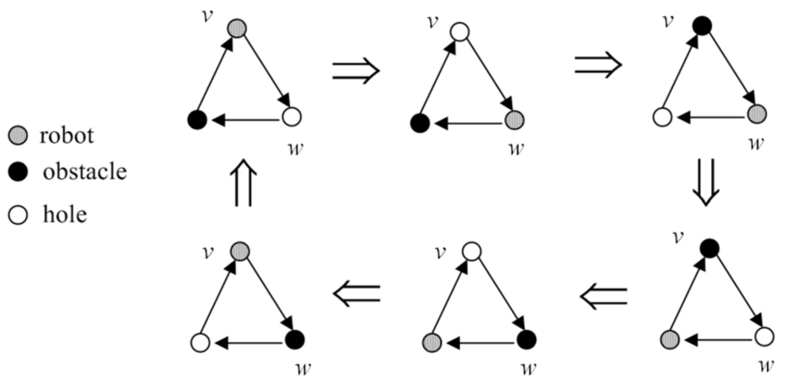

Fig. 5. Restoration by rotating the objects in a cycle.

6. Motion planning on strongly connected digraphs

At first, we make the following observation about motion planning on strongly connected digraphs.

Proposition 13. *Let $D = (V, E)$ be a strongly connected digraph. Then*

(i) *If the robot and a hole are in the same cycle C of D, then the robot can be moved to any vertex of C.*
(ii) *The movement of objects (robot or obstacles) in D preserves the feasibility of motion planning on D.*

Proof. (i) it is obvious since the hole can be moved along the reverse direction of the arcs in C and the objects can be rotated to any vertex in C.
(ii) Suppose we move an object from v to w along the arc $(v, w) \in E$. We prove that the motion planning problem is feasible before the movement iff it is feasible after the movement.

Since $(v, w) \in E$ and D is strongly connected, there is a path P from w to v in D, let C denote the cycle $P \cup \{(v, w)\}$.
Suppose the motion planning problem is feasible before the movement. Because after the movement, there is a hole in v, we can move the hole along the reverse direction of C, rotate the objects along C, and restore the situation before the movement, namely all the objects return to the positions before the movement. An example of this restoration is given in Fig. 5. So the motion planning problem is also feasible after the movement.
The other direction is obvious. □

From [14], we know that if a graph is biconnected, then one hole is sufficient to move the robot from the source vertex to the destination vertex, which is also the case for strongly biconnected digraphs.

Theorem 14. *Let D be a strongly biconnected digraph. Then the motion planning problem on D is feasible iff there is at least one hole in D.*

Proof. "Only if" part: obvious.
"If" part:
Suppose D is strongly biconnected, there is exactly one hole in D (the case that there are more than one hole is similar), the source vertex is s and the destination vertex is t.
From Theorem 8, we know that there is an open ear decomposition $P_0, \ldots, P_r$ of D.
Let j_0 be the minimal j such that s, t and the hole are all in D_j, where $D_j = \bigcup_{0 \le j' \le j} P_{j'}$.
Induction on j_0.

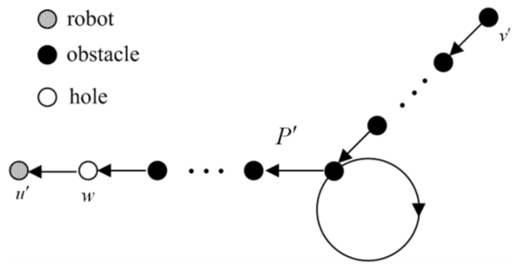

Fig. 6. The case that the hole and t are in P_{j_0} different from u' and v', $s \in D_{j_0-1}$, and $s \neq u'$.

Induction base $j_0 = 0$: s, t and the hole are all in the cycle P_0. Then move the hole along the reverse direction of the cycle and move the robot to t.

Induction step $j_0 > 0$.

Let the tail and head endpoint of P_{j_0} be u' and v' respectively.

Because of minimality of j_0, we have the following three cases.

Case I. s is in P_{j_0}, $s \neq u', v'$:

Select a path P in D_{j_0-1} from v' to u', then $P_{j_0} \cup P$ is a cycle in D.

If the hole is not on $P_{j_0} \cup P$, the hole must be in D_{j_0-1}, we can move it to P in D_{j_0-1} without moving the robot in s.

If t is on $P_{j_0} \cup P$, then move the hole along the reverse direction of $P_{j_0} \cup P$ and move the robot to t.

Otherwise, move the hole along the reverse direction of $P_{j_0} \cup P$ and move the robot to v'. Now the hole is in P_{j_0}, move the hole along the reverse direction of P_{j_0}, until it reaches u'.

Then the position of the robot, v', the destination t and the position of the hole, u', are all in D_{j_0-1}, according to the induction hypothesis, we can move the robot to t.

Case II. s is in D_{j_0-1}, the hole is in some vertex of P_{j_0} different from u' and v':

Select a path P in D_{j_0-1} from v' to u', then $P_{j_0} \cup P$ is a cycle in D.

If t is in D_{j_0-1} and $s \neq u'$, we can move the hole to u' along the reverse direction of P_{j_0} without moving the robot in s, then according to the induction hypothesis, we can move the robot to t.

If t is in D_{j_0-1} and $s = u'$, then move the hole along the reverse direction of $P_{j_0} \cup P$ and move the robot to v'. Now the hole is in P_{j_0}, we can move the hole along the reverse direction of P_{j_0} to u'. Then by the induction hypothesis, we can move the robot to t.

If t is not in D_{j_0-1}, then t is on P_{j_0}.

If $s = u'$, then we can move the hole along the reverse direction of $P_{j_0} \cup P$ and move the robot to t.

Now we consider the case $s \neq u'$.

We can move the hole along the reverse direction of P_{j_0} to u' without moving the robot. Then by the induction hypothesis, we can move the robot from s to v' in D_{j_0-1}.

By the induction hypothesis again, we can move the robot from v' to u' in D_{j_0-1}. Let the trace of the robot during the movement from v' to u' be P'. Note that P' may contain cycles. Suppose the last arc of P' is (w, u') for some w. Then the hole is in w after the movement. Without loss of generality, we assume that during the movement, the robot visits u' only once since u' is the destination. Consequently, the hole can be moved from w to v' along the reverse direction of P' without moving the robot in u' (see Fig. 6).

Since $P_{j_0} \cup P$ is a cycle and t is on P_{j_0}, now we can move the hole along the reverse direction of $P_{j_0} \cup P$ and move the robot to t.

Case III. s and the hole are both in D_{j_0-1}, t is in P_{j_0}, $t \neq u', v'$:

We can move the hole in D_{j_0-1} to v' with possible movements of the robot in D_{j_0-1}. Suppose the new position of the robot is s'.

Now move the hole to some vertex in P_{j_0} different from u' and v', which is possible since P_{j_0} contains at least three vertices. Then we have reduced Case III to Case II. □

We introduce the following notation before giving the algorithm.

Definition 15. Let $D = (V, E)$ be a strongly connected digraph, $u, v, w \in V$ such that $v \neq w$, and $\mathcal{G}_{sbc}(D) = (V_{sbc}, W_{sbc}, E_{sbc})$ be the strongly-biconnected-component graph of D. Then u is said to be on the w-side of v, if $u \neq v$ and one of the following two conditions holds:

(i) $v \in W_{sbc}$, and u, w are in the same connected component of $\mathcal{G}_s(D) - v$.

(ii) $v \notin W_{sbc}$, and either u, w are in the same connected component of $\mathcal{G}_s(D - V(B))$, or $u \in V(B)$, where B is the unique strongly biconnected component of D to which v belongs.

u is said to be on the non-w-side of v if $u \neq v$, and u is not on the w-side of v.

A hole (resp. obstacle) is said to be on the t-side of the robot if the position (vertex) of the hole (resp. obstacle) is on the t-side of the position of the robot, and a hole (resp. obstacle) is said to be on the non-t-side of the robot if the position of the hole is on the non-t-side of the position of the robot.

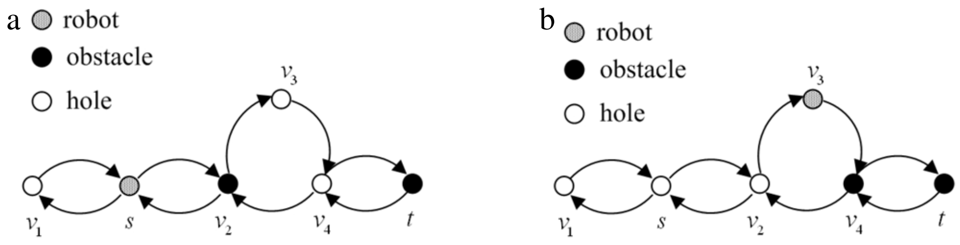

Fig. 7. Example: t-side of the robot.

Note that if $u, v, w \in V$, $v \notin W_{sbc}$, $v \neq w$, $v, w \in V(B)$, where B is the unique strongly biconnected component of D to which v belongs, then u is on the w-side of v iff $u \neq v$ and $u \in V(B)$ according to Definition 15.

Example 16 (*t-side of the Robot*). In Fig. 7(a), the robot is in $s \in W_{sbc}$, two holes in v_3 and v_4 are on the t-side of the robot, and the hole in v_1 is on the non-t-side of the robot. In Fig. 7(b), the robot is in $v_3 \notin W_{sbc}$, the hole in v_2 belongs to the same strongly biconnected component as v_3, so v_2 is on the t-side of the robot, and two holes in v_1 and s are on the non-t-side of the robot.

Since strongly biconnected components of strongly connected digraphs are similar to biconnected components of connected graphs, the feasibility of motion planning on strongly connected digraphs can be decided similar to the motion planning on graphs.

Definition 17 (*Chains*). Let $D = (V, E)$ be a strongly connected digraph, and $\mathcal{G}_{sbc}(D) = (V_{sbc}, W_{sbc}, E_{sbc})$ be the strongly-biconnected-component graph of D. Let $v \in V$, v is called a branching vertex if $v \in W_{sbc}$ and the degree of v is greater than 2 in $\mathcal{G}_{sbc}(D)$, i.e. v is shared by at least three distinct strongly biconnected components. A chain of $\mathcal{G}_{sbc}(D)$ is a path $B_0v_1B_1 \cdots B_{k-1}v_kB_k$ $(k \geq 1)$ in $\mathcal{G}_{sbc}(D)$ such that

(i) for all $1 \leq i \leq k-1$, $|V(B_i)| = 2$;
(ii) for all $1 < i < k$, v_i is not a branching vertex.

The length of a chain is the number of vertices in W_{sbc} on the chain.

Example 18. In Fig. 4, $B_1v_2B_2v_3B_3$ is a chain in $\mathcal{G}_{sbc}(D)$ since $|V(B_2)| = 2$. The length of this chain is 2.

In the following, to decide the feasibility, we first consider the restricted situation that all the holes are on the t-side of the robot, then we consider the more general case.

Let $P := B_0v_1B_1 \ldots B_{p-1}v_pB_p$ be the path in $\mathcal{G}_{sbc}(D) = (V_{sbc}, W_{sbc}, E_{sbc})$, such that $s \in V(B_0)$, $t \in V(B_p)$, $s \neq v_1$ and $t \neq v_p$.

Let l be the maximum length of the chains contained in P.

Case I. All the holes are on the t-side of the robot.

The problem is feasible iff the number of holes is no less than $l+1$, because $l+1$ holes are necessary and sufficient to let the robot go through the chains of length l. For instance, suppose $B_iv_{i+1}B_{i+1} \cdots v_{i+l}B_{i+l}$ is a chain of length l contained in P, $|V(B_i)| \geq 3$ and $|V(B_{i+l})| \geq 3$. In order to move the robot from some vertex $w_1 \in V(B_i)$ such that $w_1 \neq v_{i+1}$ to the other vertex $w_2 \in V(B_{i+l})$ such that $w_2 \neq v_{i+l}$, at first l holes are needed to occupy $v_{i+1}, \ldots, v_{i+l}$ so that the robot can be moved from w_1 to v_{i+l}, then another hole on the t-side of v_{i+l} is needed to move the robot from v_{i+l} to w_2.

Example 19. The strongly connected digraph is given in Fig. 8(a). All the holes are on the t-side of the robot. $P = B_0v_2B_1v_3B_2v_4B_3v_5B_4v_6B_5v_7B_6$. The l, i.e. the maximum length of the chains contained in P, is 3. There are $l+1 = 4$ holes, so the problem is feasible. Now we show how to move the robot from s to t with the four holes: Move the four holes to v_2, v_3, v_4, v_9 respectively (see Fig. 8(b)), then move the robot to v_9 (see Fig. 8(c)), move the obstacle on v_5 to v_4 (see Fig. 8(d)), move the three holes on s, v_2, v_3 to v_6, v_7, t respectively (see Fig. 8(e)), finally move the robot to t (see Fig. 8(f)).

Case II. There are holes not on the t-side of the robot.

If the number of holes on the t-side of the robot is already no less than $l+1$, then the problem is feasible. Otherwise, it is necessary to utilize the holes not on the t-side of the robot in order to move the robot to t.

There are three subcases,

- $|V(B_0)| \geq 3$;
- $|V(B_0)| = 2$ and s is a branching vertex;
- $|V(B_0)| = 2$ and s is not a branching vertex.

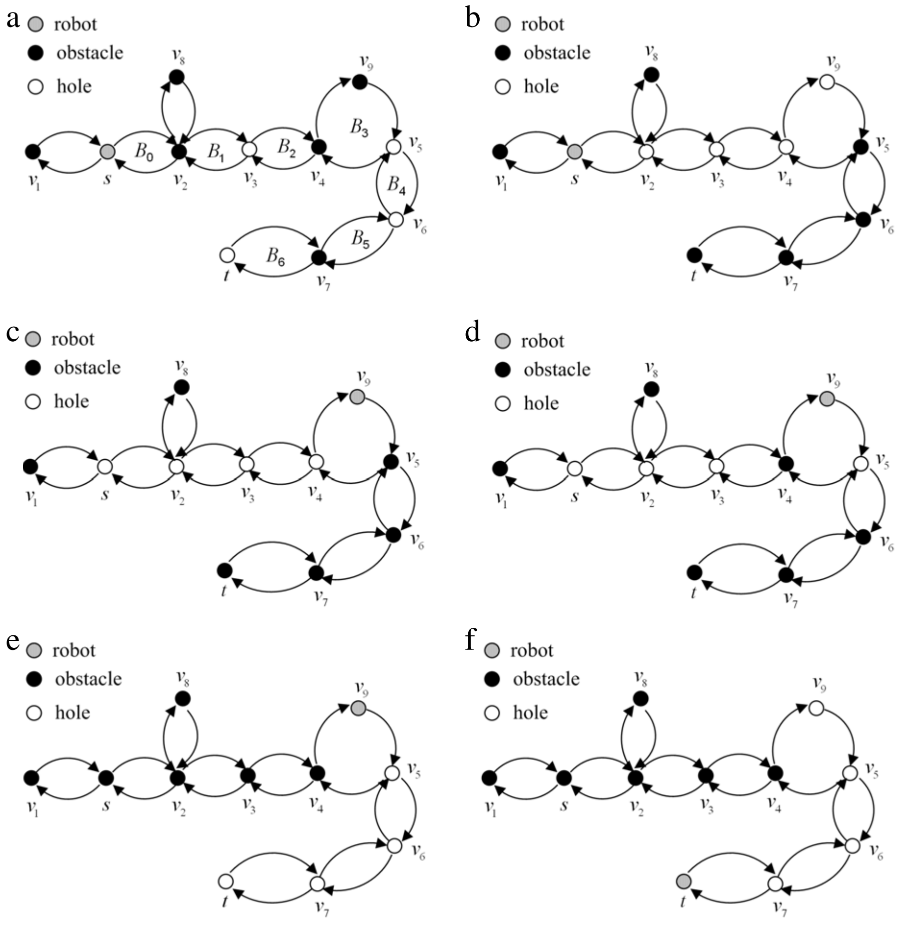
a
robot
obstacle
hole
v_1 s v_2 v_3 v_4 v_5 v_6 v_7 v_8 v_9 t
B_0 B_1 B_2 B_3 B_4 B_5 B_6
b
c
d
e
f

Fig. 8. Example: motion planning on a strongly connected digraph: Case I.

In the following, we only consider the third case to illustrate how to decide the feasibility, the considerations of the first two cases are similar.

Let i_0 be the maximal natural number such that $B_0 v_1 B_1 \cdots B_{i_0-1} v_{i_0} B_{i_0}$ is a chain contained in P.

If the number of holes on the t-side of the robot is no less than $i_0 + 1$, then the problem is feasible iff the (total) number of holes is no less than $l + 1$.

Otherwise, let B be the (unique) strongly biconnected component such that $s \in V(B)$ and $B \neq B_0$ (Recall that s is not a branching vertex.).

If $|V(B)| \geq 3$, then the robot can be moved to some vertex $w \in V(B)$ such that $(s, w) \in E$, and all the holes can be moved to the t-side of w (by moving the robot within B if necessary), the problem is feasible iff the (total) number of holes is no less than $\max(i_0 + 2, l + 1)$.

Otherwise ($|V(B)| = 2$), let $Q := B'_0 v'_1 B'_1 \cdots B'_{q-1} v'_q B'_q$ ($q \geq 1$) be a maximal chain in $\mathcal{G}_{sbc}(D)$ such that $B'_0 = B$ and v'_1 is the (unique) vertex in B different from s.

There are several situations for Q: Either v'_q is a branching vertex, or $|V(B'_q)| \geq 3$, or (v'_q is not a branching vertex, $|V(B'_q)| = 2$, and B'_q is a leaf in $\mathcal{G}_{sbc}(D)$).

If v'_q is a branching vertex or $|V(B'_q)| \geq 3$, then the problem is feasible only if the robot can be moved from s to some $w \in V(B'_q)$ such that $(v'_q, w) \in E$ (in order to move all the holes to the t-side of the robot), and the holes are sufficient to move the robot through the chain $B'_q v'_q B'_{q-1} \cdots B'_1 v'_1 B'_0 s B_0 v_1 \cdots B_{i_0-1} v_{i_0} B_{i_0}$. Therefore, the problem is feasible iff the number of holes not on the t-side of the robot is no less than $q+1$, and the (total) number of holes is no less than $\max(q+i_0+2, l+1)$.

Otherwise (v'_q is not a branching vertex, $|V(B'_q)| = 2$, and B'_q is a leaf in $\mathcal{G}_{sbc}(D)$), the problem is infeasible since it is impossible to move the robot through the chain $B_0 v_1 \cdots B_{i_0-1} v_{i_0} B_{i_0}$.

Example 20. The strongly connected digraph is given in Fig. 9(a). $P = B_3 v_4 B_4 v_5 B_5 v_6 B_6 v_7 B_7 v_8 B_8$, $l = 3$. There are holes not on the t-side of the robot. $|V(B_3)| = 2$ and s is not a branching vertex, so it is the third subcase of Case II. The number of holes on the t-side of the robot is 2, less than $l + 1 = 4$. And $i_0 = 2$, the number of holes on the t-side of the robot is less

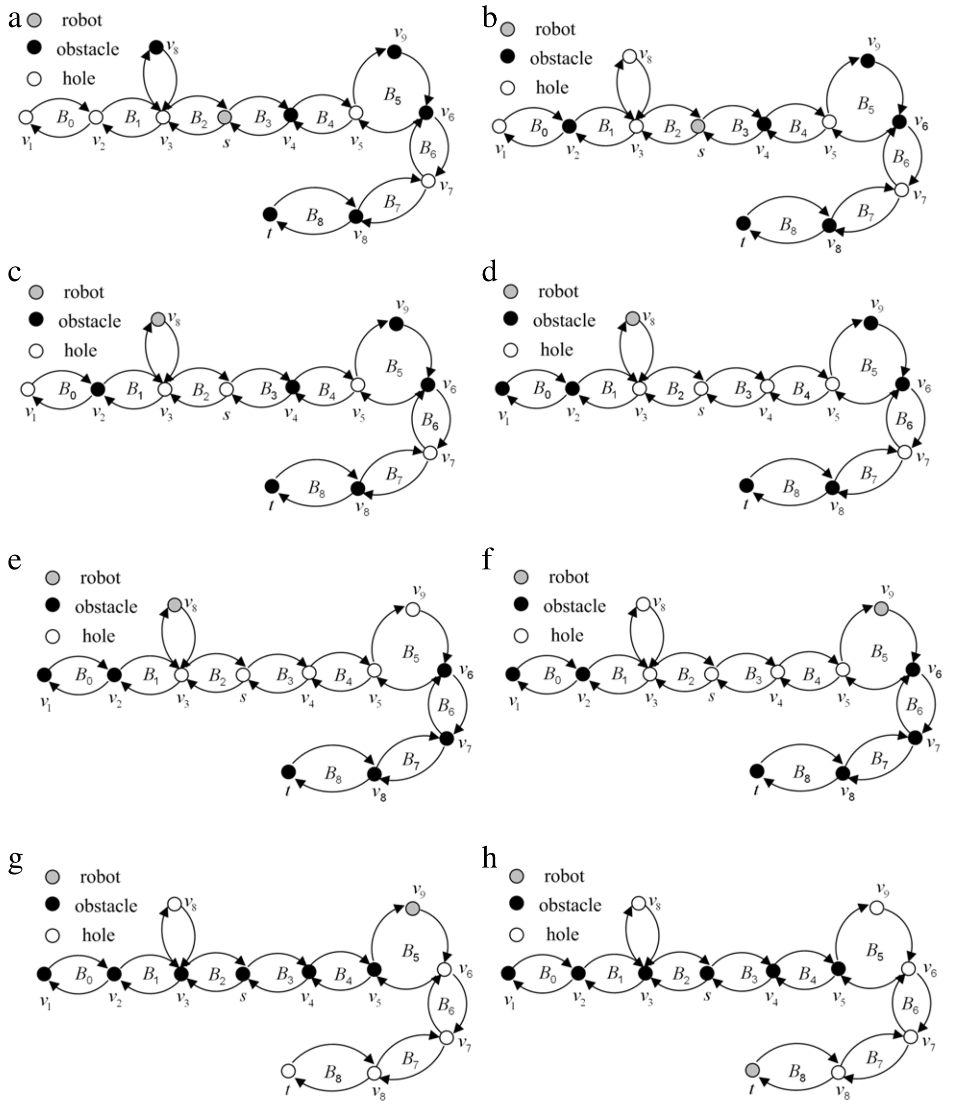

Fig. 9. Example: motion planning on a strongly connected digraph: Case II.

than $i_0 + 1 = 3$. B_2 is the unique strongly biconnected component such that $s \in V(B_2)$ and $B_2 \neq B_3$. $Q = B_2 v_3 B_1$, $q = 1$. The number of holes not on the t-side of the robot is 3, greater than $q + 1 = 2$. Moreover, the total number of holes is 5, no less than $\max(q + i_0 + 2, l + 1) = \max(5, 4) = 5$, so the problem is feasible. Now we show how to move the robot from s to t. Move the hole on v_2 to v_8 (see Fig. 9(b)), move the robot from s to v_8 (see Fig. 9(c)), move the hole on v_1 to v_4 (see Fig. 9(d)), move the hole on v_7 to v_9 (see Fig. 9(e)), move the robot to v_9 (see Fig. 9(f)), move the four holes on v_3, s, v_4, v_5 to v_6, v_7, v_8, t respectively (see Fig. 9(g)), finally move the robot to t.

Theorem 21. *Feasibility of motion planning on strongly connected graphs can be decided in $O(m)$ time.*

Proof. The construction of $\mathcal{G}_{sbc}(D)$ can be done in time $O(m)$ since the biconnected components of a connected graph of m edges can be constructed in $O(m)$ time by a depth-first-search technique [4].

The construction of P, Q takes time $O(m)$.

The total number of holes, the holes on the t-side of the robot and the holes not on the t-side of the robot, can be computed by solving reachability problems, which takes time $O(m)$ as well.

So the overall time is $O(m)$. □

7. Conclusion

In this paper, we considered the feasibility of motion planning on digraphs, we showed that the feasibility of motion planning on acyclic digraphs can be decided in time linear in the product of the number of vertices and the number of arcs, while the feasibility on strongly connected digraphs can be solved in linear time.

The algorithms for the feasibility of motion planning on acyclic digraphs and strongly connected digraphs can be adapted to the case where the capacity of each vertex is more than one (namely, vertices are able to hold several objects simultaneously), the only modification is the computation of the number of holes.

The feasibility of motion planning on digraphs is only partially solved in this paper, since we did not give the algorithm for deciding the feasibility on general digraphs, which, as well as the optimization of the motion of robot and obstacles, is much more intricate than that on graphs because of the irreversibility of the movements on digraphs.

The motion planning on graphs with one robot, GMP1R, has a natural generalization, GMPkR, where there are k robots with their respective destinations. It is also interesting to consider motion planning on digraphs with k robots, since in practice it is more reasonable that a robot shares its workspace with other robots.

GMPkR in general is a very complex problem. A special case of GMPkR, where there are no additional obstacles (thus all the movable objects have their destinations), has been considered. Wilson studied the special case of GMPkR for $k = n-1$ in [17], which is a generalization of the "15-puzzle" problem to general graphs. They gave an efficiently checkable characterization of the solvable instances of the problem. Kornhauser et al. extended this result to $k \leq n - 1$ [8]. Goldreich proved that determining the shortest move sequence for the problem studied by Kornhauser et al. is NP-hard [6]. It seems more realistic to first consider the above special case of GMPkR on digraphs.

Acknowledgements

We thank the anonymous referees for their valuable comments to improve the paper.

References

[1] V. Auletta, A. Monti, D. Parente, G. Persiano, A linear time algorithm for the feasibility of pebble motion on trees, SWAT'96, in: Proceedings of the 5th Scandinavian Workshop on Algorithm Theory, in: LNCS, vol. 1097, Springer-Verlag, 1996, pp. 259–270.
[2] V. Auletta, P. Persiano, Optimal pebble motion on a tree, Informaiton and Computation 165 (1) (2001) 42–68.
[3] J. Bang-Jensen, G. Gutin, Digraphs: Theory, Algorithms and Applications, Springer Monographs in Mathematics, Springer-Verlag, 2000.
[4] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, Introduction to Algorithms, second ed., The MIT Press, 2001.
[5] P.W. Finn, L.E. Kavmkit, Computational approaches to drug design, Algorithmica 25 (1999) 347–371.
[6] O. Goldreich, Finding the shortest move-sequence in the graph-generalized 15-puzzle is NP-hard, manuscript, 1984.
[7] Jorjeta G. Jetcheva, David B. Johnson, Routing characteristics of ad hoc networks with unidirectional links, Ad Hoc Networks 4 (3) (2006) 303–325.
[8] D. Kornhauser, G. Miller, P. Spirakis, Coordinating pebble motion on graphs, the diameter of permutation groups, and applications, in: FOCS'84, 1984, pp. 241–250.
[9] Jean-Claude Latombe, Controllability, recognizability, and complexity issues in robot motion planning, in: FOCS'95, 1995, pp. 484–500.
[10] Steven M. LaValle, Planning Algorithms, Cambridge University Press, 2006.
[11] Mahesh K. Marina, Samir R. Das, Routing performance in the presence of unidirectional links in multihop wireless networks, in: Proc. of ACM MobiHoc, 2002, pp. 12–23.
[12] Motion planning game, website, http://www.download-game.com/Motion_Planning_Game.htm.
[13] Yvonne Perrott, Track transportation systems, European patent, 1988, http://www.freepatentsonline.com/EP0284316.html.
[14] C.H. Papadimitriou, P. Raghavan, M. Sudan, H. Tamaki, Motion planning on a graph, in: FOCS'94, 1994, pp. 511–520.
[15] Guang Song, Nancy M. Amato, Using motion planning to study protein folding pathways, in: RECOMB'01: Proceedings of the Fifth Annual International Conference on Computational Biology, New York, NY, USA, ACM, 2001, pp. 287–296.
[16] Douglas B. West, Introduction to Graph Theory, second ed., Prentice Hall, 2000.
[17] R.M. Wilson, Graph puzzles, homotopy, and the alternating group, Journal of Combinatorial Theory(B) 16 (1974) 86–96.