A Dichotomy Result for Cyclic-Order Traversing Games

Yen-Ting Chen
Department of Computer Science, National Chiao Tung University, Hsinchu City, Taiwan
yentingchen.cs02@nctu.edu.tw

Meng-Tsung Tsai[1]
Department of Computer Science, National Chiao Tung University, Hsinchu City, Taiwan
mtsai@cs.nctu.edu.tw

Shi-Chun Tsai[2]
Department of Computer Science, National Chiao Tung University, Hsinchu City, Taiwan
sctsai@cs.nctu.edu.tw

Abstract

Traversing game is a two-person game played on a connected undirected simple graph with a source node and a destination node. A pebble is placed on the source node initially and then moves autonomously according to some rules. Alice is the player who wants to set up rules for each node to determine where to forward the pebble while the pebble reaches the node, so that the pebble can reach the destination node. Bob is the second player who tries to deter Alice's effort by removing edges. Given access to Alice's rules, Bob can remove as many edges as he likes, while retaining the source and destination nodes connected. Under the guide of Alice's rules, if the pebble arrives at the destination node, then we say Alice wins the traversing game; otherwise the pebble enters an endless loop without passing through the destination node, then Bob wins. We assume that Alice and Bob both play optimally.

We study the problem: When will Alice have a winning strategy? This actually models a routing recovery problem in Software Defined Networking in which some links may be broken. In this paper, we prove a dichotomy result for certain traversing games, called cyclic-order traversing games. We also give a linear-time algorithm to find the corresponding winning strategy, if one exists.

2012 ACM Subject Classification Mathematics of computing → Graph theory, Theory of computation → Design and analysis of algorithms, Networks → Network reliability

Keywords and phrases *st*-planar graphs, biconnectivity, fault-tolerant routing algorithms, software defined network

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2018.29

1 Introduction

Several mathematical models were proposed to forward packets in a routing network G, such as the sink-tree routing model [9, 4, 7], v-acorn routing model [1], and the cyclic-order routing model [17]. The cyclic-order routing model is fault-tolerant in the sense that packets

[1] This research was supported in part by the Ministry of Science and Technology of Taiwan under contract MOST grant 107-2218-E-009- 026-MY3, and the Higher Education Sprout Project of National Chiao Tung University and Ministry of Education (MOE), Taiwan.
[2] This research was supported in part by the Ministry of Science and Technology of Taiwan under contracts MOST 105-2622-8-009-008 and 105-2221-E-009-103-MY3.

Algorithm 1: The pebble-moving algorithm.

Input : A connected undirected simple graph $G = (V \cup \{s, t\}, E)$,
Alice's strategy $\{\pi_{v\to x}, \pi_{x\to v} : (v, x) \in E\}$,
Bob's removal of edges E_B so that $G - E_B$ remains st-connected.

```
pre ← s, cur ← s;
while cur ≠ t do
    foreach (cur, u) in ordered list π_{pre→cur} do
        if (cur, u) ∉ E_B then
            pre ← cur, cur ← u;
            break;
        end
    end
end
return "Alice wins";
```

can be forwarded from the source s to the destination t as long as s and t remain connected under some link and node failures, which is the most reliable one among all routing models. However, as noted in [17], the cyclic-order routing model does not apply to every network G, but no exact graph characterization is known. In this paper, we will show the exact graph class that admits the cyclic-order routing scheme, i.e. a dichotomy result. To simplify the presentation, we formulate the cyclic-order routing scheme as follows.

We define ***traversing game*** to be a two-person game played on a connected undirected simple graph $G = (V \cup \{s, t\}, E)$ with a pebble starting at node s. Alice is the player who wants to set up rules for each node $x \in V \cup \{s, t\}$ that determines where to forward the pebble while the pebble reaches x, so that the pebble can be moved autonomously from the source node s to the destination node t. Bob is the second player who tries to deter Alice's effort by removing edges. Given access to Alice's rules, Bob can remove as many edges as he likes, while retaining s and t connected. We note that removing a node x is equivalent to removing all the edges incident to x, and therefore it suffices to consider edge removals only. More formally, Alice assigns an ordered list $\pi_{v\to x}$ to each ordered pair of nodes $v \to x$ if $(v, x) \in E$, where $\pi_{v\to x}$ is a permutation of the edges incident to x with (x, v) as the last edge. Note that for an undirected edge (x, v), there are two corresponding ordered lists, i.e. $\pi_{v\to x}$ and $\pi_{x\to v}$, indicating the pebble is moving in opposite direction. We say Alice's strategy is the set of ordered lists $X = \{\pi_{v\to x}, \pi_{x\to v} : (v, x) \in E\}$. Given access to X, Bob removes some edges from E. Then the system simulates the tour of the pebble starting from node s. When the pebble reaches node x from node v, the next edge for the pebble to traverse is (x, u) where (x, u) is the first edge in $\pi_{v\to x}$ not removed by Bob. Such an edge must exist because (x, v) is one possible candidate. We assume that there is a self-loop at node s which is the starting edge in the tour of the pebble. This edge is also associated with a permutation $\pi_{s\to s}$ and Bob is not allowed to remove it. If the pebble arrives the destination t at some moment in the tour, then we say Alice wins the traversing game; otherwise the pebble enters an endless loop without passing through t, then Bob wins. The pseudocode for the above pebble moving is described in Algorithm 1. We assume further that Alice and Bob both play optimally.

The traversing game actually models a link failure recovery mechanism of Software Defined Network (SDN) with OpenFlow protocol [18, 19], whose network control plane is separated from the packet forwarding plane, and whose switches have very limited computing capacity

that may only support matching and forwarding packets. Software Defined Networking has been a focus in network and communication research in recent years, since McKeown et al. published their pioneering work [18]. Because an SDN controller needs to monitor multiple OpenFlow switches and constant interaction between controller and switches may slow down the network, some fast failover mechanism is devised [19], in particular the group table used in the OpenFlow protocol. When a packet enters an OpenFlow switch, the flow rules will match related fields in the packet to determine from which port the packet enters and then go to the corresponding group table. Each group table has a bucket list to watch whether the links are up or down. The bucket lists relate to the ordered lists $\pi_{v\to x}$ for each ordered pair of nodes $v \to x$ in the traversing game. When a link is down, the switch can quickly select the next bucket in the link's failover group table with a watch port that is up. It thus can be used to reduce the interaction between controllers and switches when link failures are detected, which is an important issue studied in SDN [19]. The traversing game is an abstraction of the above protocol.

Deciding who wins the traversing game is a problem in $\mathbf{\Sigma}_2^p$ [8, 2], which is the set of all languages L for which there exists a polynomial time Turing machine M and a polynomial q such that $x \in L \Leftrightarrow \exists u \in \{0,1\}^{q(|x|)}\ \forall y \in \{0,1\}^{q(|x|)} M(x,u,y) = 1$. We do not know whether it can be solved in polynomial time, even with a nondeterministic Turing machine. We thus impose a restriction on all the ordered lists $\pi_{v\to x}$ in the traversing game, which makes the traversing game solvable in linear time. Let π_x be a cyclic order of the edges incident to x. The restriction is, for each node $x \in V \cup \{s,t\}$, there exists a cyclic order π_x so that for every ordered pair of nodes $v \to x$, the ordered list $\pi_{v\to x}$ is equal to the segment of π_x that starts from the successor of (x,v) and finishes at (x,v). We say a traversing game with the above restriction ***cyclic-order traversing game***. In [17], the authors show that Alice has a winning strategy for a cyclic-order traversing game if the underlying graph G is comprised of (hierarchical) node-disjoint paths. We will show how to generalize this finding.

We need some notions to state our main result. ***st-planar graphs*** were first introduced by Lempel et al. [16], which are acyclic planar digraphs with exactly one source node s and exactly one sink node t and can be embedded in the plane so that s, t are both on the outer face. This definition was later adapted, for example in [3], to be such undirected graphs that have a planar embedding with s and t on the same face, or equivalently both on the outer face. We use the latter definition of st-planar graphs in this paper.

The ***st-biconnected component*** $B_{st}(G)$ of an undirected graph G is defined to be the subgraph of G induced by the nodes in the biconnected component of $G \cup \{(s,t)\}$ that contains (s,t). Or equivalently, as shown in Lemma 2, $B_{st}(G)$ is the node-induced subgraph of G with the removal of all the nodes that are not on any simple path in G from node s to node t, i.e. ***ignorable nodes***. It is clear that the removal of ignorable nodes cannot make the status of other nodes changed from unignorable to ignorable, so $B_{st}(G)$ is a unique subgraph of G, regardless of the sequence of node removals.

Our main result is:

▶ **Theorem 1.** *For a cyclic-order traversing game with underlying graph $G = (V \cup \{s,t\}, E)$, Alice has a winning strategy if and only if $B_{st}(G)$ is st-planar. In addition, there exists an $O(|V| + |E|)$-time algorithm that either outputs Alice's winning strategy or determines that there is none.*

Related Work

Annexstein et al. [1] also proposed a mathematical model for fault-tolerant routing. In their routing scheme, they need to assign an acyclic orientation to the underlying graph $G = (V \cup \{s, t\}, E)$ so that every node other than t (the sink node) has at least k out-going directed edges and k is the maximum possible among all acyclic orientations. Given the acyclic orientation, packets at node x are forwarded to any available out-going edge of x. As long as t is functioning and fewer than k nodes malfunction, packets can arrive t from nodes other than t. The orientation can be found in linear time.

The routing scheme by Annexstein et al. has no re-routing, so it is efficient to forward packets. It is clear that our routing scheme covers all the cases that Annexstein et al.'s model can handle if the st-biconnected component $B_{st}(G)$ of the underlying graph G is st-planar, so it is more fault-tolerant for such graphs, in tradeoff of the cost to re-route packets. Our routing strategy can be found in linear time as well.

Organization

The rest of the paper is organized as follows. In Section 2, we show some graph properties for st-biconnected components, which are used as building blocks for the proofs in the subsequent sections. In Section 3, we show that Alice has a winning strategy for any cyclic-order traversing game when the $B_{st}(G)$ of the underlying graph $G = (V \cup \{s, t\})$ is st-planar. In Section 4, we show that the graph class studied in Section 3 is the exact graph class that Alice has a winning strategy for cyclic-order traversing games, by studying the situations that Bob has a winning strategy. Finally, in Section 5, a linear-time algorithm is given to compute Alice's winning strategy, if one exists.

2 Properties of st-Biconnected Components

In this section, we show some properties of st-biconnected components, which are used as building blocks for the proofs in subsequent sections. Here are some notations to simplify the presentation. By $G - \{x\}$ (resp. $G - \{(x, y)\}$), we denote to remove node x (resp. edge (x, y)) from G. By $G \cup \{(x, y)\}$, we denote to add an edge (x, y), if not existing, to G. Let ***st*-path** denote an undirected path from s to t. We say a graph is ***st*-connected** if it has an st-path.

We begin with a proof showing that the two definitions of $B_{st}(G)$ are equivalent.

▶ **Lemma 2.** *For every connected undirected simple graph $G = (V \cup \{s, t\}, E)$, removing all nodes that are not on any simple st-path in G yields $B_{st}(G)$.*

Proof. Let C be the graph obtained by removing all nodes that are not on any simple st-path from G. Since G is connected, s and t are on a simple st-path, so $s, t \in C$. On the other hand, $s, t \in B_{st}(G)$ by the definition of st-biconnected component. We need to discuss for those nodes other than s and t.

Let v be a node in $B_{st}(G)$ other than s, t. Since $B_{st}(G) \cup \{(s, t)\}$ is biconnected, there are two node-disjoint paths from $\{v\}$ to $\{s, t\}$ in $B_{st}(G) \cup \{(s, t)\}$ that have only node v in common by Menger's Theorem [14]. Joining these two paths gives a simple path from s to t that passes through v, so v is a node in C.

Let v be a node in C other than s, t. Then there is a simple st-path that passes through v. Together with the edge (s, t), this gives a simple cycle containing s, v, t, so v is a node in $B_{st}(G)$. ◀

By Lemma 2 and the properties of block-cut trees [10, 13], we get:

▶ **Corollary 3.** *For every node v in a connected undirected simple graph $G = (V \cup \{s,t\}, E)$, v is not contained in $B_{st}(G)$ if and only if v can be disconnected from s and t by removing an articulation point x where $x \in B_{st}(G)$ and $x \neq v$.*

▶ **Lemma 4.** *Given a connected undirected simple graph $G = (V \cup \{s,t\}, E)$, let H be any subgraph of $B_{st}(G)$ so that H has at least two nodes and one edge, then there exists a simple st-path in $B_{st}(G)$ that contains at least two nodes in H.*

Proof. If both s and t are in H, then any simple path P in $B_{st}(G)$ from s to t contains at least two nodes in H, e.g. s and t. Such a simple path P must exist because G is connected.

If precisely one of s,t is in H, then H has a node $v \notin \{s,t\}$. By the definition of st-biconnected component, there exists a simple path P in $B_{st}(G)$ from s to t that passes through v. Hence, P contains at least two nodes in H.

The remaining case happens when none of s and t is in H, so H has an edge (u,v) where $u,v \notin \{s,t\}$. Let V_1 be the node set $\{u,v\}$ and V_2 be the node set $\{s,t\}$. By the definition of st-biconnected component, there is a simple path P_u (resp. P_v) in $B_{st}(G)$ from s to t that passes through u (resp. v). In the subgraph $P_u \cup P_v$, to disconnect u (resp. v) from V_2 by removing a single node, u (resp. v) must be the node to be removed. Since $u \neq v$, one cannot disconnect V_1 from V_2 by removing a single node. Thus by Menger's Theorem, there are two node-disjoint paths P_1, P_2 from V_1 to V_2. Joining P_1, P_2 with (u,v) yields the desired path. ◀

▶ **Lemma 5.** *For any cyclic-order traversing game, if the pebble-moving algorithm does not stop, i.e. the pebble does not arrive t, then every possible move $v \to x$ either does not occur or occur more than once.*

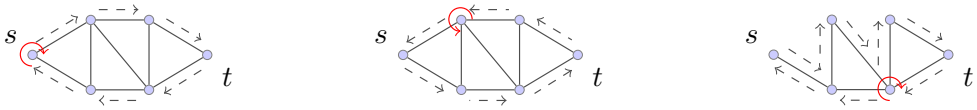
Proof. If the pebble-moving algorithm does not stop, then the tour of the pebble can be represented as an infinite sequence of moves $v_1 \to x_1, v_2 \to x_2, \ldots$ Let S_i be the subsequence of the moves after $v_i \to x_i$, and let j be the smallest $j \geq i$ so that $v_j \to x_j$ occurs more than once in S_i. Such a move $v_j \to x_j$ must exist in S_i because S_i is an infinite sequence of moves and the number of different moves is finite. We claim that $j = i$. Here is why. Suppose $j > i$, then $v_{j-1} \to x_{j-1}$ is also a move repeated in S_i because in a cyclic-order traversing game the predecessor moves of each occurrence of $v_j \to x_j$ are the same, yielding a contradiction. Therefore $v_i \to x_j$ is the first move repeats in S_i. This fact holds for every single $i \geq 1$, so every move in the tour repeats more than once. ◀

We are ready to show that Alice can create a winning strategy by bypassing ignorable nodes.

▶ **Lemma 6.** *Alice has a winning strategy for a cyclic-order traversing game with underlying graph $G = (V \cup \{s,t\}, E)$ if and only if Alice has a winning strategy for a cyclic-order traversing game with underlying graph $B_{st}(G)$.*

Proof. ($\Rightarrow$) Let $X = \{\pi_{v\to x}, \pi_{x\to v} : (v,x) \in E\}$ be Alice's winning strategy on G. Then it works for all st-connected subgraphs, in particular $B_{st}(G)$. Then we let $X' = X$, and remove all the edges in $E(G) - E(B_{st}(G))$ from these ordered lists in X', and therefore X' is a valid strategy on $B_{st}(G)$. Since X' has the same behavior as X on $B_{st}(G)$ and its st-connected subgraphs, X' is a winning strategy on $B_{st}(G)$.

($\Leftarrow$) We prove by induction on $|V(G)| - |V(B_{st}(G))|$. It is clear that the statement is true when $|V(G)| = |V(B_{st}(G))|$. Assume $|V(G)| - |V(B_{st}(G))| = k$ for some $k \geq 1$, and the statement holds up to $k-1$. By Corollary 3, there is an articulation point u separating s,t



(a) clockwise **(b)** counterclockwise **(c)** subgraph of (a)

Figure 1 An illustration of the tour of the pebble, i.e. along the outer face.

from a node not in $B_{st}(G)$. Let C be the subgraph of $G - u$ obtained by removing all the components of $G - u$ that contains s or t. Because of the existence of u, all the nodes in C are not in $B_{st}(G)$ and thus $B_{st}(G) = B_{st}(G - C)$. Then by the induction hypothesis, there is a winning strategy $Y = \{\pi_{v\to x}, \pi_{x\to v} : (v, x) \in E(G - C)\}$ on $G - C$, by which we construct a winning strategy $Y' = \{\pi'_{v\to x}, \pi'_{x\to v} : (v, x) \in E\}$ on G. Let y be any neighbor of u, and z be the neighbor of u connected by $\pi_{y\to u}(1)$, i.e. the first edge in the ordered list $\pi_{y\to u}$. Set $\pi'_{y\to u}$ as any ordered list of those edges connecting u to C followed by $\pi_{y\to u}$. For each neighbor v of u other than y, set $\pi'_{v\to u}$ as a circular shift of $\pi'_{y\to u}$ so that the requirement of cyclic-order strategy is satisfied. For each node x in $B_{st}(G) - \{u\}$ and its neighbor v, set $\pi'_{v\to x}$ as $\pi_{v\to x}$. In this way, the neighbors of u in C are placed together. If the pebble moves from y to u and then C, by Lemma 5 it will eventually leave C and u by traversing the edge (u, z), i.e. it will move from u to z after several steps rather than loop in C endlessly. Therefore, when applying Y' to G, the pebble moves exactly the same as applying Y to $G - C$ if we ignore the tour of the pebble outside $G - C$. Finally, to see that Y' is indeed a winning strategy, consider any st-connected subgraph H of G. Let P be a simple st-path on H. P does not pass through C and therefore $H - C$ is st-connected. Then the pebble moves to t when applying Y to $H - C$, as well as when applying Y' to H. ◀

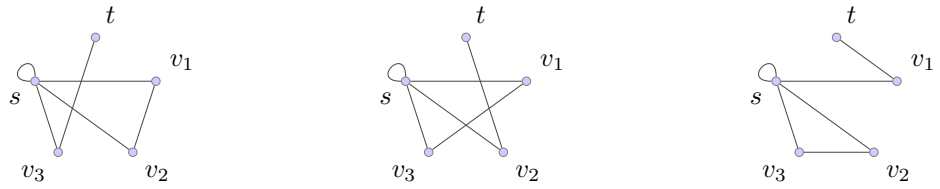
3 Winning Strategies for Alice

In this section, we will show that Alice has a winning strategy for a cyclic-order traversing game with underlying graph $G = (V \cup \{s, t\}, E)$ if $B_{st}(G)$ is st-planar; that is, the direction ($\Leftarrow$) in Theorem 1. Surprisingly, Alice has no winning strategy if the underlying graph is outside the above graph class, as shown in Section 4. Hence we get a dichotomy result for cyclic-order traversing games.

We begin with a proof showing a base case that the underlying graph $G = (V \cup \{s, t\}, E)$ is st-planar.

▶ **Lemma 7.** *Alice has a winning strategy for a cyclic-order traversing game with underlying graph $G = (V \cup \{s, t\}, E)$ if G is st-planar.*

Proof. Since G is st-planar, one can have a planar embedding for G so that s, t, (s, s) are on the outer face. Given the planar embedding, for each node $x \in V \cup \{s, t\}$, order the edges incident to x clockwise with respect to x, which yields a cyclic order c_x. We claim that Alice has a winning strategy by setting $\pi_x = c_x$ for each $x \in V \cup \{s, t\}$. In the pebble-moving algorithm, when the pebble is moved from node x to node y, the algorithm searches for the next available edge in $\pi_{x\to y}$, say (y, z), then the pebble is moved along (y, z). Since we set $\pi_y = c_y$, the transit from (x, y) to (y, z) acts like rotating clockwise with respect to y. As noted in [20], such a sequence of moves makes the pebble traverse all the edges on a single face if G is connected. Since the pebble starts the tour from the edge (s, s), an edge on the outer face, it will visit all the nodes on the outer face, in particular s and t. We depict the tour of the pebble in Figure 1.



(a) H_1 **(b)** H_2 **(c)** H_3

Figure 2 st-connected subgraphs of $K_5 \cup \{(s, s)\}$.

No matter how Bob removes edges from G, s and t still stay on the outer face. To see why, imagine that for every point p on the outer face there is a curve from p to infinity without crossing any node or edge in G. Clearly, removing any subset of edges in G cannot cut the curve, a certificate that p is on the outer face. This yields that, the pebble always visits s and t after any removal of edges, unless s and t are disconnected. In other words, Alice has a winning strategy when the underlying graph is st-planar, as claimed. ◀

Together with Lemma 6, we get:

▶ **Theorem 8.** *Alice has a winning strategy for a cyclic-order traversing game with underlying graph $G = (V \cup \{s, t\}, E)$ if $B_{st}(G)$ is st-planar.*

We remark that, in the proof of Lemma 7, if all nodes in G are on the outer face in the planar embedding, i.e. an ***outerplanar graph*** [5], then the pebble will visit all nodes regardless of Bob's removal of edges. This immediately yields that:

▶ **Corollary 9.** *Alice has a fixed winning strategy for a cyclic-order traversing game with underlying graph $G = (V, E)$, if G is outerplanar and for all choices of $s, t \in V$.*

4 Winning Strategies for Bob

In this section, we will show that Bob has a winning strategy for a cyclic-order traversing game with underlying graph $G = (V \cup \{s, t\}, E)$ if $B_{st}(G)$ is not st-planar; that is, the contraposition of the direction $(\Rightarrow)$ in Theorem 1. Together with the results shown in Section 3, this gives a dichotomy result for cyclic-order traversing games.

We begin with proofs showing base cases where G is K_5, $K_{3,3}$, and their subdivisions.

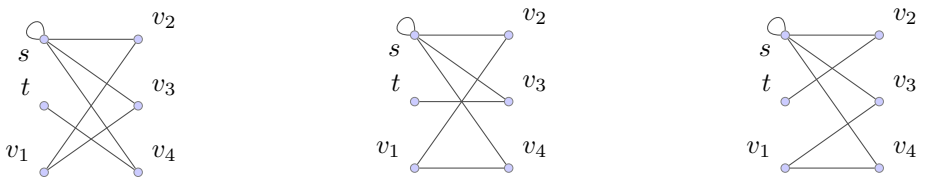
▶ **Lemma 10.** *Bob has a winning strategy for a cyclic-order traversing game with underlying graph $G = (V \cup \{s, t\}, E)$ if $G \cup \{(s, t)\}$ is isomorphic to K_5.*[3]

Proof. We prove the case where $(s, t) \notin E$, then the other case follows. The graphs in Figure 2 are possible subgraphs of G after Bob's removal of edges. We show that Alice cannot assign an ordered list to each $\pi_{v \to x}$ that simultaneously works for H_1, H_2, and H_3. Hence, Bob has a winning strategy on G.

To see why, Alice may set $\pi_{s \to s}$ as any of the following six ordered lists.

***list*$_1$:** $(s, v_1), (s, v_2), (s, v_3), (s, s)$
***list*$_2$:** $(s, v_1), (s, v_3), (s, v_2), (s, s)$
***list*$_3$:** $(s, v_2), (s, v_1), (s, v_3), (s, s)$

[3] In Lemmas 10, 11, and 12, we ignore the self-loop (s, s) while deciding graph isomorphism.



(a) H_1 **(b)** H_2 **(c)** H_3

Figure 3 st-connected subgraphs of $K_{3,3} \cup \{(s,s)\}$.

(a) H_4 **(b)** H_5

Figure 4 st-connected subgraphs of $K_{3,3} \cup \{(s,s)\}$.

$\boldsymbol{list_4}$: $(s,v_2),(s,v_3),(s,v_1),(s,s)$
$\boldsymbol{list_5}$: $(s,v_3),(s,v_1),(s,v_2),(s,s)$
$\boldsymbol{list_6}$: $(s,v_3),(s,v_2),(s,v_1),(s,s)$

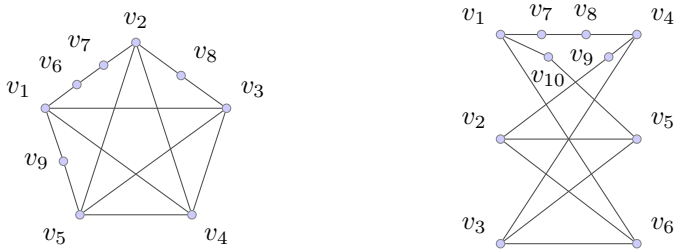
However, if Alice sets $\pi_{s\to s} = list_2$, then it does not work for H_1, because $\pi_{v_2\to s} = (s,s),(s,v_1),(s,v_3),(s,v_2)$ and the pebble moves in the cycle s,s,v_1,v_2,s,s without passing through t. Moreover, if Alice sets $\pi_{s\to s} = list_4$, then it also does not work for H_1, because the pebble moves in the cycle s,s,v_2,v_1,s,s. By the same argument, one can show that setting $\pi_{s\to s} = list_1$ or $list_6$ does not work for H_2, and setting $\pi_{s\to s} = list_3$ or $list_5$ does not work for H_3. This already excludes all possibilities, thus completing the proof. ◀

▶ **Lemma 11.** *Bob has a winning strategy for a cyclic-order traversing game with underlying graph $G = (V \cup \{s,t\}, E)$ if G or $G \cup \{(s,t)\}$ is isomorphic to $K_{3,3}$.*

Proof. First we consider the case where s and t are in the same partition. The graphs in Figure 3 are possible subgraphs of G after Bob's removal of edges. Alice may set $\pi_{s\to s}$ as any of the following six ordered lists.

$\boldsymbol{list_1}$: $(s,v_2),(s,v_3),(s,v_4),(s,s)$
$\boldsymbol{list_2}$: $(s,v_2),(s,v_4),(s,v_3),(s,s)$
$\boldsymbol{list_3}$: $(s,v_3),(s,v_2),(s,v_4),(s,s)$
$\boldsymbol{list_4}$: $(s,v_3),(s,v_4),(s,v_2),(s,s)$
$\boldsymbol{list_5}$: $(s,v_4),(s,v_2),(s,v_3),(s,s)$
$\boldsymbol{list_6}$: $(s,v_4),(s,v_3),(s,v_2),(s,s)$

However, if Alice sets $\pi_{s\to s} = list_2$, then it does not work for H_1, because $\pi_{v_3\to s} = (s,s),(s,v_2),(s,v_4),(s,v_3)$ and the pebble moves in the cycle s,s,v_2,v_1,v_3,s,s without passing through t. Moreover, if Alice sets $\pi_{s\to s} = list_4$, then it also does not work for H_1, because the pebble moves in the cycle s,s,v_3,v_1,v_2,s,s. By the same argument, one can show that setting $\pi_{s\to s} = list_1$ or $list_6$ does not work for H_2, and setting $\pi_{s\to s} = list_3$ or $list_5$ does not work for H_3.



(a) G_1 **(b)** G_2

Figure 5 Subdivisions of K_5 and $K_{3,3}$.

Next we consider the case where s and t are in different partitions, and prove the subcase where $(s,t) \notin E$, then the other subcase follows. Consider the graphs in Figure 4. Alice may set $\pi_{s\to s}$ as any of the following two ordered lists.

$\boldsymbol{list_1}$: $(s,v_3),(s,v_4),(s,s)$
$\boldsymbol{list_2}$: $(s,v_4),(s,v_3),(s,s)$

Alice may also set $\pi_{s\to v_3}$ as any of the following two ordered lists.

$\boldsymbol{list_3}$: $(v_3,v_1),(v_3,v_2),(v_3,s)$
$\boldsymbol{list_4}$: $(v_3,v_2),(v_3,v_1),(v_3,s)$

However, if Alice sets $\pi_{s\to s} = list_1$ and $\pi_{s\to v_3} = list_3$, then it does not work for H_4, because the pebble moves in the cycle s,s,v_3,v_1,v_4,s,s without passing through t. Moreover, if Alice sets $\pi_{s\to s} = list_2$ and $\pi_{s\to v_3} = list_4$, then it also does not work for H_4, because the pebble moves in the cycle s,s,v_4,v_1,v_3,s,s. By the same argument, one can show that setting $\pi_{s\to s} = list_1$ and $\pi_{s\to v_3} = list_4$ does not work for H_5. Setting $\pi_{s\to s} = list_2$ and $\pi_{s\to v_3} = list_3$ also does not work for H_5. This already excludes all possibilities, thus completing the proof. ◀

▶ **Lemma 12.** *Bob has a winning strategy for a cyclic-order traversing game with underlying graph $G = (V \cup \{s,t\}, E)$ if G or $G \cup \{(s,t)\}$ is isomorphic to a subdivision of K_5 or $K_{3,3}$.*

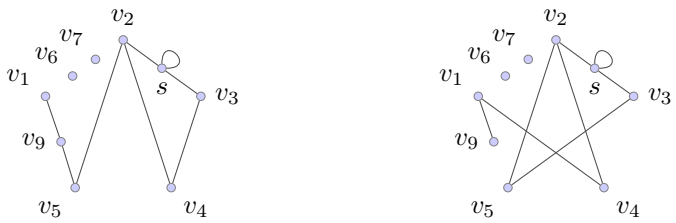
Proof. Bob has a winning strategy if and only if he has one after removing a node with degree one, except s and t. This is also true for smoothing out a node with degree two or subdividing an edge, because removing an incident edge of a node with degree two is equivalent to removing both. Therefore we first transform G or $G \cup \{(s,t)\}$ into one of the graphs in Figure 5.

If both s and t belong to $V(K_5)$ or $V(K_{3,3})$, by Lemma 10 and Lemma 11 the statement is true. In what follows, we consider other choices of s and t.

Case 1: G or $G \cup \{(s,t)\}$ is isomorphic to G_1.

Case 1(a): $\{s,t\} = \{v_1,v_6\}$, or $s = v_6$ and $t = v_7$. Note that (s,t) may or may not belong to $E(G)$. Assume $(s,t) \notin E(G)$, $s = v_6$, and $t = v_1$. By Lemma 10 Alice has no winning strategy to move pebble from v_2 to v_1, and therefore from v_6 to v_1. The remaining cases can be reduced to this one.

Case 1(b): $s = v_1$ and $t = v_8$. Note that G is isomorphic to G_1 in this case. We first smooth out nodes with degree two, i.e. v_6, v_7, and v_9. Let $\mathcal{D}$ be the collection of v_1v_2-connected subgraphs of $K_5 - \{(v_1,v_2)\}$. Let $\mathcal{R}$ be the collection of v_1v_8-connected subgraphs of G that contains (v_2,v_8) but not (v_1,v_2). Define $f : \mathcal{D} \to \mathcal{R}$ to be the



(a) H_1 **(b)** H_2

Figure 6 Subgraphs of $G_1 \cup \{(s, s)\}$.

bijection from $\mathcal{D}$ to $\mathcal{R}$ such that $f(D) \in \mathcal{R}$ is obtained from $D \in \mathcal{D}$ by adding (v_2, v_8) and replacing (v_2, v_3) with (v_8, v_3) if it is in D. For any $D \in \mathcal{D}$, the pebble moves exactly the same on D and $f(D)$. By Lemma 10, for each of Alice's strategies, the pebble cannot move from v_1 to v_2 for some $D \in \mathcal{D}$, and also cannot move from v_1 to v_8 for $f(D)$.

Case 1(c): $s = v_7$ and $t = v_8$. Similar to the proof of Case 1(b), we let $\mathcal{R}$ be the collection of v_7v_8-connected subgraphs of G that contains (v_2, v_8) and (v_7, v_1), but not (v_7, v_2).

Case 1(d): $s = v_8$ and $t = v_1$, or $s = v_8$ and $t = v_9$. Consider the graphs in Figure 6. Alice may set $\pi_{s\to s}$ as any of the following two ordered lists.

***list*$_1$:** $(s, v_2), (s, v_3), (s, s)$

***list*$_2$:** $(s, v_3), (s, v_2), (s, s)$

Alice may also set $\pi_{s\to v_2}$ as any of the following two ordered lists.

***list*$_3$:** $(v_2, v_4), (v_2, v_5), (v_2, s)$

***list*$_4$:** $(v_2, v_5), (v_2, v_4), (v_2, s)$

However, if Alice sets $\pi_{s\to s} = list_1$ and $\pi_{s\to v_2} = list_3$, then it does not work for H_1, because the pebble moves in the cycle $s, s, v_2, v_4, v_3, s, s$ without passing through t. Moreover, if Alice sets $\pi_{s\to s} = list_2$ and $\pi_{s\to v_2} = list_4$, then it also does not work for H_1, because the pebble moves in the cycle $s, s, v_3, v_4, v_2, s, s$. By the same argument, one can show that setting $\pi_{s\to s} = list_1$ and $\pi_{s\to v_2} = list_4$ does not work for H_2, and setting $\pi_{s\to s} = list_2$ and $\pi_{s\to v_2} = list_3$ also does not work for H_2.

Case 2: G or $G \cup \{(s, t)\}$ is isomorphic to G_2.

Case 2(a): $\{s, t\} = \{v_1, v_7\}$ or $s = v_7$ and $t = v_8$. Assume $(s, t) \notin E(G)$, $s = v_7$, and $t = v_1$. By Lemma 11 Alice has no winning strategy to move pebble from v_4 to v_1, and therefore from v_7 to v_1. The remaining cases can be reduced to this one.

Case 2(b): $s = v_1$ and $t = v_9$. Similar to the proof of Case 1(b), we let $\mathcal{R}$ be the collection of v_1v_9-connected subgraphs of G that contains (v_4, v_9) but not (v_1, v_4).

Case 2(c): $s = v_8$ and $t = v_9$. Similar to the proof of Case 1(b), we let $\mathcal{R}$ be the collection of v_8v_9-connected subgraphs of G that contains (v_4, v_9) and (v_1, v_8), but not (v_4, v_8).

Case 2(d): $s = v_9$ and $t = v_1$, or $s = v_9$ and $t = v_{10}$. Consider the graphs in Figure 7. Alice may set $\pi_{s\to s}$ as any of the following two ordered lists.

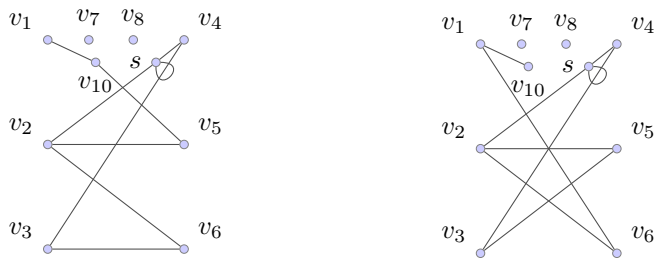
***list*$_1$:** $(s, v_2), (s, v_4), (s, s)$

***list*$_2$:** $(s, v_4), (s, v_2), (s, s)$

Alice may also set $\pi_{s\to v_2}$ as any of the following two ordered lists.

***list*$_3$:** $(v_2, v_5), (v_2, v_6), (v_2, s)$

***list*$_4$:** $(v_2, v_6), (v_2, v_5), (v_2, s)$



(a) H_3 **(b)** H_4

Figure 7 Subgraphs of $G_2 \cup \{(s, s)\}$.

However, if Alice sets $\pi_{s\to s} = list_1$ and $\pi_{s\to v_2} = list_4$, then it does not work for H_3, because the pebble moves in the cycle $s, s, v_2, v_6, v_3, v_4, s, s$ without passing through t. Moreover, if Alice sets $\pi_{s\to s} = list_2$ and $\pi_{s\to v_2} = list_3$, then it also does not work for H_3, because the pebble moves in the cycle $s, s, v_4, v_3, v_6, v_2, s, s$. By the same argument, one can show that setting $\pi_{s\to s} = list_1$ and $\pi_{s\to v_2} = list_3$ does not work for H_4, and setting $\pi_{s\to s} = list_2$ and $\pi_{s\to v_2} = list_4$ also does not work for H_4. ◀

▶ **Theorem 13.** *Bob has a winning strategy for a cyclic-order traversing game with underlying graph $G = (V \cup \{s, t\}, E)$ if $B_{st}(G)$ is not st-planar.*

Proof. $B_{st}(G)$ is not st-planar implies that $B_{st}(G) \cup \{(s, t)\}$ is non-planar. By Kuratowski's Theorem [15], $B_{st}(G) \cup \{(s, t)\}$ has a Kuratowski subgraph H, i.e. a subdivision of K_5 or $K_{3,3}$. By Lemma 4, Bob can find a simple path P from s to t in $B_{st}(G)$ that passes through at least two nodes in H. Let P_1 be the subpath starting from s and finishing at the first node in P that is contained in H. Let P_2 be the subpath starting from the last node in P that is contained in H and finishing at t. Bob's winning strategy is to remove all the edges outside $H - \{(s, t)\} \cup P_1 \cup P_2$. By applying Lemma 12 and Lemma 6, we are done. ◀

5 Linear-Time Algorithm

Finally, we give a linear-time algorithm that either outputs Alice's winning strategy or outputs "Bob wins." This completes the proof of Theorem 1.

▶ **Theorem 14.** *For any cyclic-order traversing game, one can use Algorithm 2 to find a winning strategy for Alice in linear time, if one exists.*

Proof. By Lemma 2, Step 1 is equivalent to finding the biconnected component in $G \cup \{(s, t)\}$ that contains s and t, which can be computed in linear time [11]. By Theorem 8, Step 2, 3, and 4 are equivalent to testing planarity and embedding $B_{st}(G) \cup \{(s, t)\}$ in the plane, which also can be solved in linear time [6, 12, 21]. For Step 5, the conversion can be done by bypassing ignorable nodes as shown in Lemma 6, which also takes linear time. In total, it takes linear time to find a winning strategy for Alice. ◀

6 Conclusion

We identify an interesting traversal problem from a practical network paradigm– software defined networking. We discover that for st-planar graphs we can always find a way in linear time to set up the cyclic-order rules for autonomous re-routing. This can be useful

Algorithm 2: Algorithm to find Alice's winning strategy in linear time.

Input : $G = (V \cup \{s, t\}, E)$
Output : Alice's winning strategy if one exists, or determine that there is none.

Find the st-biconnected component $B_{st}(G)$;
if $B_{st}(G)$ *is st-planar* **then**
  Embed $B_{st}(G) \cup \{(s, t)\}$ in the plane so that $s, t, (s, s)$ are on the same face;
  Find Alice's winning strategy Y on $B_{st}(G)$;
  Convert Y to a winning strategy X on G;
  return X;
else
  return "Bob wins";
end

for designing fault-tolerant network. However, if we allow different type of rules, instead of cyclic ones, then it is not clear when Alice can have a winning strategy. We leave it as an open problem. Meanwhile, we do not know the exact complexity class of the traversing game. We conjecture that it can be $\mathbf{\Sigma}_2^p$-complete.

References

1 Fred S. Annexstein, Kenneth A. Berman, Tsan-Sheng Hsu, and Ram Swaminathan. A multi-tree routing scheme using acyclic orientations. *Theoretical Computer Science*, 240(2):487–494, 2000.

2 Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

3 Giorgio Ausiello, Paolo Giulio Franciosa, Isabella Lari, and Andrea Ribichini. Max flow vitality in general and planar graphs. *CoRR*, abs/1710.01965, 2017. arXiv:1710.01965.

4 Dimitri P. Bertsekas and Robert G. Gallager. *Data Networks, Second Edition*. Prentice Hall, 1992.

5 Gary Chartrand and Frank Harary. Planar permutation graphs. *Annales de l'Institut Henri Poincaré B*, 3(4):433–438, 1967.

6 Norishige Chiba, Takao Nishizeki, Shigenobu Abe, and Takao Ozawa. A linear algorithm for embedding planar graphs using PQ-Trees. *Journal of Computer and System Sciences*, 30(1):54–76, 1985.

7 Reuven Cohen, Baiju V. Patel, Frank Schaffa, and Marc Willebeek-LeMair. The Sink Tree Paradigm: Connectionless Traffic Support on ATM LAN's. *IEEE/ACM Trans. Netw.*, 4(3):363–374, 1996.

8 Ding-Zhu Du and Ker-I Ko. *Theory of Computational Complexity, Second Edition*. Wiley, 2014.

9 Eli M. Gafni and Dimitri P. Bertsekas. Distributed algorithms for generating loop-free routes in networks with frequently changing topology. *IEEE Trans. Comm.*, 29:11–18, 1981.

10 Frank Harary. *Graph theory*. Addison-Wesley, 1991.

11 John Hopcroft and Robert Tarjan. Algorithm 447: Efficient Algorithms for Graph Manipulation. *Commun. ACM*, 16(6):372–378, June 1973. doi:10.1145/362248.362272.

12 John Hopcroft and Robert Tarjan. Efficient planarity testing. *Journal of the Association for Computing Machinery*, 21:549–568, 1974.

13 Tsan-sheng Hsu and Ming-Yang Kao. A Unifying Augmentation Algorithm for Two-Edge Connectivity and Biconnectivity. *J. Comb. Optim.*, 2(3):237–256, 1998.

14 Bernhard Korte and Jens Vygen. *Combinatorial Optimization: Theory and Algorithms*. Springer Publishing Company, Incorporated, 6th edition, 2018.

15 Casimir Kuratowski. Sur le problème des courbes gauches en Topologie. *Fundamenta Mathematicae*, 15(1):271–283, 1930. URL: `http://eudml.org/doc/212352`.

16 A. Lempel, S. Even, and I. Cederbaum. An Algorithm for Planarity Testing of Graphs. In *Theory of Graphs, International Symposium*, pages 215–232, 1966.

17 Y.-Z. Liao and S.-C. Tsai. Fast Failover with Hierarchical Disjoint Paths in SDN. In *IEEE Global Communications Conference (GLOBECOM)*, 2018.

18 Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38:69–74, 2008.

19 SDN. `https://www.opennetworking.org/sdn-resources/sdn-definition`. [Online].

20 Pierre Rosenstiehl and Robert E. Tarjan. Rectilinear Planar Layouts and Bipolar Orientations of Planar Graphs. *Discrete Comput. Geom.*, 1(4):343–353, December 1986.

21 W.-K. Shih and W.-L Hsu. A new planarity test. *Theo. Comput. Sci.*, 223:179–191, 1999.