Independent Set Reconfiguration on Directed Graphs

Takehiro Ito
Graduate School of Information Sciences, Tohoku University, Sendai, Japan

Yuni Iwamasa
Graduate School of Informatics, Kyoto University, Kyoto, Japan

Yasuaki Kobayashi
Graduate School of Information Science and Technology, Hokkaido University, Sapporo, Japan

Yu Nakahata
Division of Information Science, Nara Institute of Science and Technology, Ikoma, Japan

Yota Otachi
Graduate School of Informatics, Nagoya University, Nagoya, Japan

Masahiro Takahashi
Graduate School of Informatics, Kyoto University, Kyoto, Japan

Kunihiro Wasa
Faculty of Science and Engineering, Hosei University, Tokyo, Japan

Abstract

Directed Token Sliding asks, given a directed graph and two sets of pairwise nonadjacent vertices, whether one can reach from one set to the other by repeatedly applying a local operation that exchanges a vertex in the current set with one of its out-neighbors, while keeping the nonadjacency. It can be seen as a reconfiguration process where a token is placed on each vertex in the current set, and the local operation slides a token along an arc respecting its direction. Previously, such a problem was extensively studied on undirected graphs, where the edges have no directions and thus the local operation is symmetric. Directed Token Sliding is a generalization of its undirected variant since an undirected edge can be simulated by two arcs of opposite directions.

In this paper, we initiate the algorithmic study of Directed Token Sliding. We first observe that the problem is PSPACE-complete even if we forbid parallel arcs in opposite directions and that the problem on directed acyclic graphs is NP-complete and W[1]-hard parameterized by the size of the sets in consideration. We then show our main result: a linear-time algorithm for the problem on directed graphs whose underlying undirected graphs are trees, which are called polytrees. Such a result is also known for the undirected variant of the problem on trees [Demaine et al. TCS 2015], but the techniques used here are quite different because of the asymmetric nature of the directed problem. We present a characterization of yes-instances based on the existence of a certain set of directed paths, and then derive simple equivalent conditions from it by some observations, which yield an efficient algorithm. For the polytree case, we also present a quadratic-time algorithm that outputs, if the input is a yes-instance, one of the shortest reconfiguration sequences.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms

Keywords and phrases Combinatorial reconfiguration, token sliding, directed graph, independent set, graph algorithm

Digital Object Identifier 10.4230/LIPIcs.MFCS.2022.58

Related Version *Full Version*: `https://arxiv.org/abs/2203.13435`

Funding *Takehiro Ito*: JSPS KAKENHI Grant Numbers JP18H04091, JP19K11814, JP20H05793
Yuni Iwamasa: JSPS KAKENHI Grant Numbers JP20K23323, JP20H05795, JP22K17854
Yasuaki Kobayashi: JSPS KAKENHI Grant Numbers JP20H00595, JP20H05793, JP20K19742
Yu Nakahata: JSPS KAKENHI Grant Numbers JP19H01103, JP20H05794, JP22H03549, JP22K17851
Yota Otachi: JSPS KAKENHI Grant Numbers JP18H04091, JP20H05793, JP21K11752, JP22H00513
Kunihiro Wasa: JSPS KAKENHI Grant Numbers JP19K20350, JP20H05793, JP22H03549, JP22K17849

1 Introduction

In a reconfiguration problem, we are given an instance of a decision problem with two feasible solutions I^{s} and I^{t}. The task is to decide if we can transform I^{s} into I^{t} by repeatedly applying a modification rule while maintaining the feasibility of intermediate solutions. Reconfiguration problems are studied for several problems such as Independent Set [17, 28, 8, 12, 13, 18, 4, 10, 29, 3, 26], Dominating Set [34, 16, 30], Clique [27], Matching [9, 5, 23], and Graph Coloring [7, 32, 6]. (See also surveys [35, 31].)

Among the problems, Independent Set Reconfiguration is one of the most well-studied problems. There are three different modification rules studied in the literature: Token Addition and Removal (TAR) [19, 28], Token Jumping (TJ) [24, 25, 10], and Token Sliding (TS) [17, 8, 12, 13, 18, 4, 29, 3, 26]. TAR allows us to add or remove any vertex in the current independent set as long as the size of the resultant set is at least a given threshold. TJ allows us to exchange any vertex in the set with any vertex outside the set. TS is a restricted version of TJ that requires the exchanged vertices to be adjacent. We call Independent Set Reconfiguration under TS simply Token Sliding. Observe that all these rules are *symmetric*: If one can transform I^{s} into I^{t}, then one also can transform I^{t} into I^{s}. Our question is: What happens if we adopt an *asymmetric* rule?

In this paper, we study a directed variant of Token Sliding, which we call Directed Token Sliding. In this problem, the input graph is directed and we can slide tokens along the arcs only in their directions. Here, we say that a set I of vertices of a directed graph G is an independent set when it is an independent set in the underlying undirected graph of G.[1] If we allow two arcs with the opposite directions, this problem is a generalization of Token Sliding, and thus PSPACE-complete in general.

We show three results for Directed Token Sliding. First, we show that the problem is PSPACE-complete even on oriented graphs, where antiparallel arcs (two parallel arcs with opposite directions) are not allowed. Second, if we restrict input graphs to directed acyclic graphs (DAGs), we prove that the problem is NP-complete. Moreover, the problem is $W[1]$-hard parameterized by the number of tokens. As for our positive and main result, we show that the problem can be solved in linear time on polytrees, that is, digraphs whose underlying graphs are trees. For a polytree, we show that all reconfiguration sequences have the same length. We can also construct one of them in $\mathrm{O}(k|V|)$ time, where k is the size of the input independent sets and V is the set of vertices of the input graph. Note that our algorithm is optimal in the following sense: We can show that there exists an infinite family of instances on directed paths whose reconfiguration sequences have length $\Omega(k|V|)$ (which is easily obtained from lower bound examples for undirected paths given in [12]). For Token Sliding on undirected trees, Demaine et al. [12] showed a linear-time algorithm to check the reconfigurability. They also showed an $\mathrm{O}(|V|^2)$-time algorithm to construct a reconfiguration sequence of length $\mathrm{O}(|V|^2)$ for yes-instances. However, the output sequence is not guaranteed to be shortest. The best known algorithm to output a shortest reconfiguration sequence for undirected trees runs in $\mathrm{O}(|V|^5)$ time [33]. In contrast to the undirected counterpart, our algorithm to construct a (shortest) reconfiguration sequence on polytrees runs in $\mathrm{O}(k|V|)$ time, which is optimal. It should be mentioned that our quadratic-time algorithm does not imply a quadratic-time algorithm for finding a shortest reconfiguration sequence for undirected trees since we do not allow polytrees to have antiparallel arcs.

[1] One may define an independent set of a digraph in an asymmetric way: A subset I of the vertex set V of the directed graph G is an independent set if $u \in I$ implies $v \notin I$ for all $u, v \in V$ such that G has an arc (u, v). However, this definition is the same as ours. (If u and v are adjacent in the underlying undirected graph, at most one of u and v can belong to I.)

Due to the space limitation, several proofs (marked with $\star$) are omitted and can be found in the full version [21].

Related work

Token Sliding on undirected graphs was introduced by Hearn and Demaine [17]. They show that the problem is PSPACE-complete even on planar graphs with maximum degree 3. The problem is also PSPACE-complete on bipartite graphs [29] and split graphs [3]. In contrast to these hardness results, the problem is polynomial-time solvable on cographs [28], claw-free graphs [8], trees [12], cactus graphs [18], interval graphs [4], bipartite permutation graphs, and bipartite distance-hereditary graphs [13].

Interestingly, Independent Set Reconfiguration on bipartite graphs is NP-complete under TAR and TJ [29]. Such graph classes are not known for TS. In this paper, we show that Directed Token Sliding is NP-complete on directed acyclic graphs (DAGs).

Token Sliding is also studied from the viewpoint of fixed-parameter tractability (FPT). When the parameter is the number of tokens, the problem is W[1]-hard both on C_4-free graphs and bipartite graphs, while it becomes FPT on their intersection, C_4-free bipartite graphs [2]. In this paper, we show that Directed Token Sliding is W[1]-hard on DAGs.

Although Token Sliding, as far as we know, has not been studied for digraphs, there are some studies on reconfiguration considering directions of edges. An orientation of a simple undirected graph is an assignment of directions to the edges of the graph. There are several studies for reconfiguring orientations of a graph, such as strong orientations [15, 14, 20], acyclic orientations [14], nondeterministic constraint logic [17], and α-orientations [1]. Very recently, Ito et al. [22] studied reconfiguration of several subgraphs in directed graphs, such as directed trees, directed paths, and directed acyclic subgraphs. Although these reconfiguration problems are defined on directed graphs, the reconfiguration rules are symmetric, meaning that if one solution X can be obtained from another solution Y by applying some reconfiguration rule, Y can be obtained from X by the same rule as well.

2 Preliminaries

In this section, we introduce notations on graphs and define our problem. For a positive integer k, we define $[k] := \{1, \ldots, k\}$. We also define $[0] := \emptyset$.

2.1 Graph notation

Let G be a directed graph (digraph). We denote by $V(G)$ and $A(G)$ the vertex and arc sets of G, respectively. For a digraph G, its *underlying (undirected) graph*, denoted by G^{und}, is an undirected graph obtained by ignoring the directions of arcs in G. For an undirected graph G', we denote by $V(G')$ and $E(G')$ the vertex and edge sets of G', respectively. For a digraph G, a vertex set $S \subseteq V(G)$ is said to be *independent* if it is an independent set in the underlying graph G^{und}, i.e., for all two different vertices $u, v \in S$, we have $\{u, v\} \notin E(G^{\mathrm{und}})$. A digraph G is an *oriented graph* if, for all vertices $u, v \in V(G)$, G contains at most one of the possible arcs (u, v) or (v, u). If G contains no directed cycles, G is *acyclic* and such digraphs are called *directed acyclic graphs (DAGs)*. If G^{und} is a tree, G is a *polytree*. Similarly, if G^{und} is a forest, G is a *polyforest*.

For an arc $e = (u, v) \in A(G)$ of a digraph G, the vertex u is called the *tail* of e, and v called the *head* of e. The both u and v are called the *endpoints* of e. For $e \in A(G)$ and $v \in V(G)$, e is *incident to* v if v is an endpoint of e. If $(u, v) \in A(G)$, u is an *in-neighbor* *of v in G* and v is an *out-neighbor of u in G*. In addition, u is a *neighbor* of v and v is a *neighbor* of u. For a vertex v, we denote by $N^+_G(v)$ and $N^-_G(v)$ the sets of out-neighbors and in-neighbors of v in G, respectively, that is, $N^+_G(v) \coloneqq \{u \in V(G) \mid (v,u) \in A(G)\}$ and $N^-_G(v) \coloneqq \{u \in V(G) \mid (u,v) \in A(G)\}$. We denote by $N_G(v)$ the set of neighbors of v, that is, $N_G(v) \coloneqq N^+_G(v) \cup N^-_G(v)$. In addition, we define $N_G[v] \coloneqq N_G(v) \cup \{v\}$. We also define $\Gamma^+_G(v) \coloneqq \{(v,u) \mid u \in N^+_G(v)\}$, $\Gamma^-_G(v) \coloneqq \{(u,v) \mid u \in N^-_G(v)\}$, and $\Gamma_G(v) \coloneqq \Gamma^+_G(v) \cup \Gamma^-_G(v)$. If no confusion arises, we omit the subscript G from $N_G(v)$, $N^+_G(v)$, $N^-_G(v)$, $N_G[v]$, $\Gamma^+_G(v)$, $\Gamma^-_G(v)$, and $\Gamma_G(v)$. A *directed path* P is a sequence of vertices and arcs $(v_1, e_1, \dots, e_{\ell-1}, v_\ell)$ such that, all the vertices are distinct and, for every $i \in [\ell-1]$, $e_i = (v_i, v_{i+1})$ holds. We call v_1 and v_ℓ the *source* and *sink* of P, and denote by $s(P)$ and $t(P)$, respectively. This P is called a *v_1-v_ℓ (directed) path* or a *(directed) path from v_1 to v_ℓ*. The *length* of P is the number of arcs and we denote it by $|P|$, that is, $|P| = \ell - 1$. For a directed path P with $|P| \geq 2$, we define $s'(P) \coloneqq v_2$ and $t'(P) \coloneqq v_{\ell-1}$. For vertex sets X and Y with the same size k on a digraph, we refer to a set $\mathcal{P} = \{P_1, \dots, P_k\}$ of directed paths as a *directed path matching from X to Y* if $\mathcal{P}$ have distinct sources and distinct sinks, that is, $\{s(P_i) \mid i \in [k]\} = X$ and $\{t(P_i) \mid i \in [k]\} = Y$. Note that two sets X and Y may intersect in this definition.

2.2 Definition of Directed Token Sliding

Let I^{s} and I^{t} be independent sets in a digraph G with $|I^{\mathsf{s}}| = |I^{\mathsf{t}}|$. A sequence $\langle I_0, \dots, I_\ell \rangle$ of independent sets of G is a *reconfiguration sequence from I^{s} to I^{t} in G* if $I_0 = I^{\mathsf{s}}, I_\ell = I^{\mathsf{t}}$, and for all $i \in [\ell]$ we have $I_{i-1} \setminus I_i = \{u\}$ and $I_i \setminus I_{i-1} = \{v\}$ with $(u,v) \in A$. The *length* of $\langle I_0, \dots, I_\ell \rangle$ is ℓ. We say that I^{s} *is reconfigurable into I^{t} in G* and I^{t} *is reconfigurable from I^{s} in G* if there is a reconfiguration sequence from I^{s} to I^{t}.

We study the following problem:

Problem Directed Token Sliding
Instance A triple $(G, I^{\mathsf{s}}, I^{\mathsf{t}})$, where G is a digraph and $I^{\mathsf{s}}, I^{\mathsf{t}} \subseteq V(G)$ are independent sets in G with $|I^{\mathsf{s}}| = |I^{\mathsf{t}}|$.
Question Is I^{s} reconfigurable into I^{t}?

In Directed Token Sliding, the sets I^{s} and I^{t} can be seen as the initial and target positions of "tokens" placed on the vertices in the sets, and then the problem asks whether we can move the tokens from I^{s} to I^{t} by repeatedly sliding tokens along arcs keeping that the tokens are not adjacent. The difference between our problem and Token Sliding is that, in the former, the input graph is directed and we can slide tokens along the arcs only in their directions. However, since we can simulate an undirected edge with the two arcs with the opposite directions, the problem is a generalization of Token Sliding. It immediately follows from the PSPACE-completeness of Token Sliding [17] that Directed Token Sliding is PSPACE-complete in general.

3 Hardness results

In this section, we provide hardness results for Directed Token Sliding. The first hardness result is the PSPACE-completeness of Directed Token Sliding for oriented graphs. This follows from a reduction from Token Sliding on undirected graphs, which is PSPACE-complete [17].

The idea of the proof of Theorem 1 as follows. From an undirected graph G, we construct a directed graph G' by replacing each edge of G with two directed arcs in opposite directions. Then, $(G, I^{\mathsf{s}}, I^{\mathsf{t}})$ is a yes-instance of Token Sliding if and only if $(G', I^{\mathsf{s}}, I^{\mathsf{t}})$ is a yes-instance of Directed Token Sliding. To make G' being an oriented graph (that is, G' has at most one arc of (u, v) and (v, u) for $u, v \in V(G)$), we replace each vertex v of G with two copies of vertices v_1 and v_2 and each edge $\{u, v\}$ of G with a strongly connected tournament on $\{u_1, u_2, v_1, v_2\}$, instead of the above replacement. This yields an oriented graph G' and we can show that $(G, I^{\mathsf{s}}, I^{\mathsf{t}})$ is a yes-instance of Token Sliding if and only if $(G', J^{\mathsf{s}}, J^{\mathsf{t}})$ is a yes-instance of Directed Token Sliding, where $J^{\mathsf{s}} = \{v_1 \mid v \in I^{\mathsf{s}}\}$ and $J^{\mathsf{t}} = \{v_1 \mid v \in I^{\mathsf{t}}\}$.

▶ **Theorem 1** ($\star$). *Directed Token Sliding is PSPACE-complete on oriented graphs.*

The second hardness result is the NP-completeness and W[1]-hardness of Directed Token Sliding for DAGs. These results are obtained by reducing Multicolored Independent Set, which is known to be NP-complete and W[1]-hard parameterized by the solution size k [11], to Directed Token Sliding.

▶ **Theorem 2** ($\star$). *Directed Token Sliding on DAGs is NP-complete and $W[1]$-hard parameterized by the number of tokens.*

4 Linear-time algorithm for polytrees

This section is devoted to proving our main result Theorem 3 below, a linear-time algorithm for Directed Token Sliding on polytrees.

▶ **Theorem 3.** *Let $T = (V, A)$ be a polytree. Directed Token Sliding on T can be solved in $\mathrm{O}(|V|)$ time. Moreover, if the answer is affirmative, then all reconfiguration sequences have the same length, and one of them can be constructed in $\mathrm{O}(k|V|)$ time, where k is the size of the input independent set.*

To establish Theorem 3, we give a simple characterization for yes-instances of Directed Token Sliding on polytrees. This characterization is described by the concept of directed path matchings, which will be given in the next subsection, and is a vital role in proving Theorem 3. We would like to mention that our characterization is rather simple but its proof is highly non-trivial. Given this characterization, we devise a linear time algorithm for Directed Token Sliding on polytrees and a quadratic time algorithm for constructing an actual reconfiguration sequence if the answer is affirmative, which will be given in Section 4.4.

4.1 Directed path matching

Let I^{s} and I^{t} be independent sets in T with $|I^{\mathsf{s}}| = |I^{\mathsf{t}}|$ such that I^{s} is reconfigurable into I^{t}. In a reconfiguration sequence from I^{s} to I^{t}, the i-th token moves along some directed path P_i. Clearly $\mathcal{P} := \{P_1, \dots, P_k\}$ forms a directed path matching from I^{s} to I^{t}. Thus, the existence of a directed path matching is a trivial necessary condition for yes-instances. However, the converse is not true in general. Let us consider a digraph such that its underlying graph is the star with four leaves and its center has two in-neighbors and two out-neighbors. We set I^{s} (resp. I^{t}) to be the set of in-neighbors (resp. out-neighbors) of the center. Then this graph has a directed path matching from I^{s} to I^{t}, while it is a no-instance. Nevertheless, directed path matchings still give us some insight on Directed Token Sliding for polytrees, which is vital for our linear-time algorithm. To this end, in this subsection, we give a characterization of the existence of a directed path matching between given two sets of vertices in a polytree. This characterization is described in terms of an invariant associated with each arc e.

Let X and Y be (not necessarily disjoint) sets of vertices of a polytree $T = (V, A)$ with $|X| = |Y|$, and π a bijection from X to Y. Since T is a polytree, for each $x \in X$ an (undirected) x-$\pi(x)$ path P_x is uniquely determined in T^{und}. For each $e \in A$, we define $w(e; X, Y, \pi) := |\{x \in X \mid \overrightarrow{P_x} \text{ has } e\}| - |\{x \in X \mid \overrightarrow{P_x} \text{ has the reverse of } e\}|$, where $\overrightarrow{P_x}$ is a directed path obtained from P_x by orienting arcs from x to $\pi(x)$. The following lemma states that $w(e; X, Y, \pi)$ does not depend on a particular π. Let T' be the polyforest obtained from T by removing an arc e. Let C_e^+ (resp. C_e^-) denote the vertices of the (weakly) connected component in T' containing the head of e (resp. the tail of e).

▶ **Lemma 4** ($\star$). *Let X and Y be (not necessarily disjoint) sets of vertices of T with $|X| = |Y|$, and $e \in A$ an arc of T. For each bijection $\pi : X \to Y$, we have $w(e; X, Y, \pi) = |C_e^- \cap X| - |C_e^- \cap Y|$.*

Based on this fact, we define a function w on A with respect to two vertex sets X and Y:

$$w(e; X, Y) := |C_e^- \cap X| - |C_e^- \cap Y| \qquad (e \in A).$$

Then, we show the necessary and sufficient conditions mentioned above.

▶ **Lemma 5** ($\star$). *There exists a directed path matching from X to Y if and only if $w(e; X, Y) \geq 0$ for every arc $e \in A$.*

By Lemmas 4 and 5, we can observe the following corollaries.

▶ **Corollary 6.** *For a yes-instance, the number of tokens passing through an arc e is equal to $w(e; I^{\mathsf{s}}, I^{\mathsf{t}})$ in every reconfiguration sequence. In particular, all reconfiguration sequences have the same length $\sum_{e \in A} w(e; I^{\mathsf{s}}, I^{\mathsf{t}})$.*

▶ **Corollary 7.** *If there exists an arc $e \in A$ such that $w(e; I^{\mathsf{s}}, I^{\mathsf{t}}) < 0$, then the instance is not reconfigurable.*

In the following argument, we assume $w(e; I^{\mathsf{s}}, I^{\mathsf{t}}) \geq 0$ for all $e \in A$. By depth-first search on T, we can compute the function $w(e; I^{\mathsf{s}}, I^{\mathsf{t}})$ for all e in linear time.

4.2 Tokens that move at most once

In a reconfiguration sequence, there may be a token that moves at most once. In other words, a token may not move at all or may move from the initial position to one of its out-neighbors and stay there. Such a token causes an exception in our further discussion, and thus we want to remove such tokens in advance. In this subsection, we show that such tokens are determined regardless of reconfiguration sequences, and that we can remove such tokens from the input without changing the reconfigurability.

First, we consider tokens that never move. The following lemma states that such tokens are uniquely determined from the instance $(T, I^{\mathsf{s}}, I^{\mathsf{t}})$ (not depending on an actual reconfiguration sequence). From now on, we simply write $w(e)$ to denote $w(e; I^{\mathsf{s}}, I^{\mathsf{t}})$.

▶ **Lemma 8** ($\star$). *Let $(T, I^{\mathsf{s}}, I^{\mathsf{t}})$ be a yes-instance. For every reconfiguration sequence, the set of vertices containing tokens that do not move in the sequence is equal to*

$$R := \{v \in I^{\mathsf{s}} \cap I^{\mathsf{t}} \mid w(e) = 0 \textit{ for all } e \in \Gamma(v)\}.$$

A token on a vertex $v \in R$ is said to be *rigid*. By Lemma 8, all rigid tokens do not move in any reconfiguration sequence.

For some $v \in R$, suppose that there exists $u \in N(v)$ such that T has an arc $e \in \Gamma(u)$ with $w(e) > 0$. If $(T, I^{\mathsf{s}}, I^{\mathsf{t}})$ is a yes-instance, in any reconfiguration sequence, some token passes through e, implying that this token is placed on u at some point. However, since the token on v is rigid, these tokens must be adjacent. Thus, we obtain the following corollary.

▶ **Corollary 9.** *For $v \in R$, if there exists $u \in N(v)$ such that T has an arc $e \in \Gamma(u)$ with $w(e) > 0$, then the instance $(T, I^{\mathsf{s}}, I^{\mathsf{t}})$ is a no-instance.*

Next, we consider tokens that move exactly once. We can show that the set of arcs used by such tokens is uniquely determined from the instance $(T, I^{\mathsf{s}}, I^{\mathsf{t}})$ (not depending on an actual reconfiguration sequence) by a similar argument as in Lemma 8.

▶ **Lemma 10** ($\star$)**.** *Let $(T, I^{\mathsf{s}}, I^{\mathsf{t}})$ be a yes-instance. For every reconfiguration sequence, the set of arcs used by tokens that move exactly once in the sequence is equal to*

$$B \coloneqq \{e = (u, v) \in A \mid w(e) = 1 \text{ and } w(e') = 0 \text{ for every } e' \in (\Gamma(u) \cup \Gamma(v)) \setminus \{e\}\}.$$

Take any $(u, v) \in B$. By Lemma 10, for every reconfiguration sequence the token on u slides to an out-neighbor v of u and stays there, which also implies $u \in I^{\mathsf{s}}$ and $v \in I^{\mathsf{t}}$. Since the tokens must form an independent set, the other tokens can be placed on neither u nor v. Given this, we refer to an arc $e \in B$ as a *blocking arc* in $(T, I^{\mathsf{s}}, I^{\mathsf{t}})$.

We can compute the rigid tokens and blocking arcs from $(T, I^{\mathsf{s}}, I^{\mathsf{t}})$ in linear time. Let R be the set of vertices containing rigid tokens and B the set of blocking arcs. Let T' be the polyforest obtained from T by removing each arc f such that f is incident to a vertex in R, or $f \notin B$ and f is incident to an endpoint of $e \in B$. Then, every component of T' is either an isolated vertex in R, a component containing the two vertices connected by an arc in B, or a component without any rigid tokens or blocking arcs. The following lemma reduces our problem to a slightly simplified one.

▶ **Lemma 11** ($\star$)**.** *Suppose that for every $v \in R$, there is no $u \in N(v)$ such that T has an arc $e \in \Gamma(u)$ with $w(e) > 0$. Then, $(T, I^{\mathsf{s}}, I^{\mathsf{t}})$ is a yes-instance if and only if $(T', I^{\mathsf{s}}, I^{\mathsf{t}})$ is a yes-instance.*

It is easy to see that if T' has more than one connected components, then we can solve the problem independently on each connected component. Moreover, the problem is trivial on a component of size at most two. Hence, we can assume that the input polytree has no vertices on which rigid tokens are placed or blocking arcs.

4.3 Necessary and sufficient conditions for yes-instances

In this subsection, we show the necessary and sufficient conditions for yes-instances. By Lemma 11, we may assume that the instance does not contain rigid tokens and blocking arcs. For a yes-instance, pick some reconfiguration sequence and let P_i be a directed path along which the i-th token moves in the reconfiguration sequence. $\mathcal{P} = \{P_1, \ldots, P_k\}$ is obviously a directed path matching from I^{s} to I^{t}. Since the instance does not contain rigid tokens and blocking arcs, we have $|P_i| \geq 2$ for every $i \in [k]$. In addition, the successors of source vertices in $\mathcal{P}$ must be distinct. To see this, observe that if the source vertices of two paths P_i and P_j in $\mathcal{P}$ have a common successor $x \coloneqq s'(P_i) = s'(P_j)$, then the tokens on $s(P_i)$ and $s(P_j)$ are both "gazing" the vertex x, and thus we cannot slide either of the tokens on $s(P_i)$ and $s(P_j)$ at all. Therefore, $\mathcal{P}$ must satisfy that $s'(P_i) \neq s'(P_j)$ for all $i, j \in [k]$ with $i \neq j$. Symmetrically, the predecessors of sink vertices must be distinct, that is, $t'(P_i) \neq t'(P_j)$ for all $i, j \in [k]$ with $i \neq j$. The goal of this subsection is to show that these necessary conditions are also sufficient.

▶ **Lemma 12.** *Let $(T, I^{\mathsf{s}}, I^{\mathsf{t}})$ be an instance of* Directed Token Sliding *without rigid tokens and blocking arcs. Then $(T, I^{\mathsf{s}}, I^{\mathsf{t}})$ is a yes-instance if and only if there exists a set $\mathcal{P} = \{P_1, \ldots, P_k\}$ of directed paths satisfying all the following conditions:*
(P1) *$|P_i| \geq 2$ for all $i \in [k]$,*
(P2) *$\mathcal{P}$ is a directed path matching from I^{s} to I^{t},*
(P3) *$s'(P_i) \neq s'(P_j)$ for all $i, j \in [k]$ with $i \neq j$, and*
(P4) *$t'(P_i) \neq t'(P_j)$ for all $i, j \in [k]$ with $i \neq j$.*
We refer to the four conditions (P1)–(P4) as the *path-set conditions.*

We have already seen the only-if direction as above. In the following, we show the other direction. We call a vertex in a path other than the source or the sink an *internal vertex.* For vertices $u, v \in V$, we define $\mathrm{dist}(u, v)$ as the length of the unique directed u-v path in T if such a path exists. We refer to a pair of directed paths P and P' satisfying the following conditions as a *biased path pair*:

- P and P' have a common internal vertex x,
- $\mathrm{dist}(s(P), x) > \mathrm{dist}(s(P'), x)$ and $\mathrm{dist}(x, t(P)) > \mathrm{dist}(x, t(P'))$.

▶ **Lemma 13** ($\star$). *If there exists a set of paths satisfying the path-set conditions, then there also exists a set of paths that satisfies the path-set conditions and does not include any biased path pair.*

In the following, let $\mathcal{P}^* = \{P_1^*, \ldots, P_k^*\}$ be a set of paths that satisfies the path-set conditions and has no biased path pairs. We show that there exists a bijection $\pi : [k] \to [k]$ having the following property $(*)$:
$(*)$ I^{s} is reconfigurable into I^{t} in a path-by-path manner following π; that is, by first moving the $\pi(1)$-th token from $s(P^*_{\pi(1)})$ all the way to $t(P^*_{\pi(1)})$ along $P^*_{\pi(1)}$, then moving the $\pi(2)$-th token from $s(P^*_{\pi(2)})$ all the way to $t(P^*_{\pi(2)})$ along $P^*_{\pi(2)}$, and so on.
From now on, we consider how to construct π satisfying $(*)$. For a vertex v and a directed path P, we say that v *touches* P or P *touches* v if $N[v] \cap V(P) \neq \emptyset$, where $V(P)$ denotes the set of vertices in P. For $i \neq j$, let (A) and (B) be the following conditions:
(A) $s(P_i^*)$ touches P_j^*;
(B) $t(P_j^*)$ touches P_i^*.

A binary relation $\dashleftarrow$ is defined as: $i \dashleftarrow j$ if and only if at least one of (A) and (B) holds for different i and j. Let $G_{\dashleftarrow}$ be the directed graph such that the vertex set of $G_{\dashleftarrow}$ is $[k]$ and, for $i, j \in [k]$, $G_{\dashleftarrow}$ has the arc (i, j) if and only if $j \dashleftarrow i$ holds.

If $G_{\dashleftarrow}$ is a directed acyclic graph, then we can construct π satisfying $(*)$ as follows. Since $G_{\dashleftarrow}$ is a DAG, there is a vertex i such that its out-degree is 0. For any $j \neq i$, every vertex in P_i^* does not belong to $N[s(P_j^*)]$ since P_i^* does not touch $s(P_j^*)$, and every vertex in P_j^* does not belong to $N[t(P_i^*)]$ since P_j^* does not touch $t(P_i^*)$. The former implies that the token on $s(P_i^*)$ sliding to $t(P_i^*)$ along P_i^* does not make adjacent token pairs and hence forms a reconfiguration sequence from I^{s} to $I^{\mathsf{s}} \setminus \{s(P_i^*)\} \cup \{t(P_i^*)\}$. The latter implies that the graph $G_{\dashleftarrow}$ for the resulting independent set $I^{\mathsf{s}} \setminus \{s(P_i^*)\} \cup \{t(P_i^*)\}$ is obtained by deleting the vertex i, which is still a DAG. By repeating the above procedure, I^{s} is reconfigurable into I^{t} in a path-by-path manner. Thus a bijection $\pi : [k] \to [k]$ satisfying $\pi(1) < \pi(2) < \cdots < \pi(k)$ with respect to a topological order of $G_{\dashleftarrow}$ admits $(*)$, as required.

The following lemma says that $G_{\dashleftarrow}$ is actually a directed acyclic graph, which verifies the if-condition of Lemma 12.

▶ **Lemma 14.** *$G_{\dashleftarrow}$ is a directed acyclic graph.*

Proof. Suppose, to the contrary, that there is a directed cycle in $G_{\dashleftarrow}$. We can assume that $(1, 2, \dots, \ell, 1)$ is a minimal one. For convenience, the addition $+$ and subtraction $-$ are taken over modulo ℓ, i.e., $\ell + 1$ is regarded as 1 and 0 is regarded as ℓ.

Suppose moreover that P^*_i and P^*_{i+1} have a common internal vertex for all $i \in [k]$. Let x and y be the source and the sink of the maximal common subpath of P^*_i and P^*_{i+1}, respectively. Since T is a polytree, these x and y are uniquely determined. For $i \in [\ell]$, at least one of (A) and (B) holds. If (A) $s(P^*_i)$ touches P^*_{i+1}, then we have $\mathrm{dist}(s(P^*_i), x) \leq 1$. If $\mathrm{dist}(s(P^*_i), x) \geq \mathrm{dist}(s(P^*_{i+1}), x)$, either $s'(P^*_i) = s'(P^*_{i+1})$ or $s(P^*_i)$ and $s(P^*_{i+1})$ are adjacent, which do not hold as (P3) or the fact that I^{s} is an independent set. Thus, we have $\mathrm{dist}(s(P^*_i), x) < \mathrm{dist}(s(P^*_{i+1}), x)$. If (B) $t(P^*_{i+1})$ touches P^*_i, then we have $\mathrm{dist}(y, t(P^*_{i+1})) \leq 1$. Then, by a symmetric argument, we have $\mathrm{dist}(y, t(P^*_i)) > \mathrm{dist}(y, t(P^*_{i+1}))$. Let z_i be an arbitrary common internal vertex of P^*_i and P^*_{i+1}. The above argument is summarised as

$$\mathrm{dist}(s(P^*_i), z_i) < \mathrm{dist}(s(P^*_{i+1}), z_i) \text{ or } \mathrm{dist}(z_i, t(P^*_i)) > \mathrm{dist}(z_i, t(P^*_{i+1})). \tag{1}$$

Equation (1) and the non-existence of biased pairs together imply that for every i, we have

$$\mathrm{dist}(s(P^*_i), z_i) \leq \mathrm{dist}(s(P^*_{i+1}), z_i) \text{ and } \mathrm{dist}(z_i, t(P^*_i)) \geq \mathrm{dist}(z_i, t(P^*_{i+1})). \tag{2}$$

For a walk W in T^{und}, let $\overrightarrow{W}$ be an oriented walk obtained from W by orienting each edge from (arbitrary) one of the end vertices to the other. Let $\mathrm{fwd}(\overrightarrow{W})$ and $\mathrm{rev}(\overrightarrow{W})$ be the numbers of times that $\overrightarrow{W}$ passes through arcs in the forward and reverse directions in T, respectively. Here, by the fact that $\mathrm{dist}(s(P^*_i), z_i) \leq \mathrm{dist}(s(P^*_{i+1}), z_i)$ (Equation (2)), there exists an oriented walk $\overrightarrow{W}$ from $s(P^*_i)$ to $s(P^*_{i+1})$ satisfying $\mathrm{fwd}(\overrightarrow{W}) \leq \mathrm{rev}(\overrightarrow{W})$. This can be obtained by traversing T^{und} from $s(P^*_i)$ to $s(P^*_{i+1})$ via z_i (which we denote by $s(P^*_i) \to z_i \to s(P^*_{i+1})$). In particular, if $\mathrm{dist}(s(P^*_i), z_i) < \mathrm{dist}(s(P^*_{i+1}), z_i)$, then $\mathrm{fwd}(\overrightarrow{W}) < \mathrm{rev}(\overrightarrow{W})$ holds. Suppose that $\mathrm{dist}(s(P^*_1), z_1) < \mathrm{dist}(s(P^*_2), z_1)$ holds. Then, the closed oriented walk

$$\overrightarrow{W_s} := s(P^*_1) \to z_1 \to s(P^*_2) \to \cdots \to s(P^*_\ell) \to z_\ell \to s(P^*_1)$$

satisfies $\mathrm{fwd}(\overrightarrow{W_s}) < \mathrm{rev}(\overrightarrow{W_s})$. See Figure 1 for an illustration. This implies that there is an arc e in T such that e occurs in $\overrightarrow{W_s}$ at least once and the number of occurrences of e is strictly smaller than that of the reverse of e in $\overrightarrow{W_s}$. This contradicts the fact that T is a polytree. Thus suppose $\mathrm{dist}(s(P^*_1), z_1) \geq \mathrm{dist}(s(P^*_2), z_1)$ holds. Then $\mathrm{dist}(z_1, t(P^*_1)) > \mathrm{dist}(z_1, t(P^*_2))$ follows from Equation (1). By a similar argument as above, we can construct a closed oriented walk $\overrightarrow{W_t}$ from $t(P^*_1)$ to $t(P^*_1)$ satisfying $\mathrm{fwd}(\overrightarrow{W_t}) > \mathrm{rev}(\overrightarrow{W_t})$. This also contradicts the fact that T is a polytree. Thus, we derive a contradiction, assuming that $G_{\dashleftarrow}$ has a directed cycle and P^*_i and P^*_{i+1} have a common internal vertex for all $i \in [k]$.

The remaining task is to show that P^*_i and P^*_{i+1} have a common internal vertex for all i under the assumption that $G_{\dashleftarrow}$ has a directed cycle. Suppose to the contrary that P^*_i and P^*_{i+1} have no common internal vertex. We only consider the case where the condition (A) in the definition of $\dashleftarrow$ holds for $i \dashleftarrow i{+}1$, that is, P^*_{i+1} touches $s(P^*_i)$; the case for (B) is symmetric. As $|P^*_i| \geq 2$, P^*_i has an arc $(s'(P^*_i), x^*)$. See Figure 2 for an illustration. For $x \in N_T(s'(P^*_i))$, we denote by C_x the weakly connected component containing x in the polyforest obtained from T by deleting the arc $(s'(P^*_i), x)$. Then observe that $V(P^*_{i+1}) \cap C_{x^*} = \emptyset$. To see this, if $V(P^*_{i+1}) \cap C_{x^*} \neq \emptyset$, then P^*_{i+1} must have the arc $(s'(P^*_i), x^*)$ as P^*_{i+1} touches $s(P^*_i)$. This particularly implies that $s'(P^*_i)$ belongs to P^*_{i+1} and is different from $t(P^*_{i+1})$ due to the direction of arc $(s'(P^*_i), x^*)$. Since I^{s} is an independent set, we have $s'(P^*_i) \neq s(P^*_{i+1})$. Thus P^*_i and P^*_{i+1} have a common internal vertex $s'(P^*_i)$, contradicting the assumption that P^*_i and P^*_{i+1} have no common internal vertex. Hence we obtain $V(P^*_{i+1}) \cap C_{x^*} = \emptyset$.

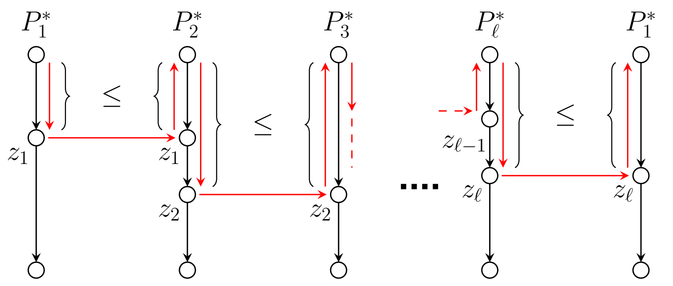

Figure 1 Closed oriented walk $s(P_1^*) \to z_1 \to s(P_2^*) \to \cdots \to s(P_\ell^*) \to z_\ell \to s(P_1^*)$.

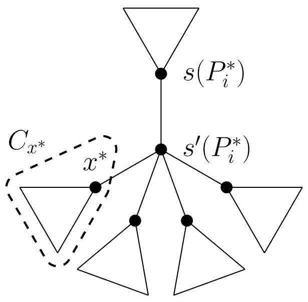

Figure 2 An illustration of a polytree considered in the proof of Lemma 14.

Suppose $i-1 = i+1$, i.e., $i+1 \dashleftarrow i$ (which implies $\ell = 2$). Then P_i^* touches $s(P_{i+1}^*)$ or P_{i+1}^* touches $t(P_i^*)$. In the former case, since $V(P_{i+1}^*) \cap C_{x^*} = \emptyset$ and I^{s} is an independent set, $s(P_{i+1}^*)$ must belong to $N_T(s'(P_i^*))$. Then we have $s'(P_{i+1}^*) = s'(P_i^*)$ by $i \dashleftarrow i+1$, which contradicts (P3). In the latter case, since $V(P_{i+1}^*) \cap C_{x^*} = \emptyset$, we have $x^* = t(P_i^*)$ and $s'(P_i^*) \in V(P_{i+1}^*)$. Moreover, neither $s'(P_i^*) = s(P_{i+1}^*)$ nor $s'(P_i^*) = t(P_{i+1}^*)$, since I^{s} and I^{t} are independent sets. Thus $s'(P_i^*)$ must be an internal vertex of P_{i+1}^*, contradicting the assumption.

Suppose $i-1 \neq i+1$ (which implies $\ell \geq 3$). Observe that $V(P_{i-1}^*) \cap C_{s(P_i^*)} = \emptyset$. To see this, suppose $V(P_{i-1}^*) \cap C_{s(P_i^*)} \neq \emptyset$. As $i-1 \dashleftarrow i$, either $s(P_{i-1}^*)$ touches P_i^* or $t(P_i^*)$ touches P_{i-1}^*. In both cases, $s(P_i^*)$ touches P_{i-1}^* and hence $i \dashleftarrow i-1$, contradicting the minimality of the cycle $(1, 2, \ldots, \ell, 1)$. Observe also that $s'(P_i^*) \notin V(P_{i-1}^*)$ as otherwise we obtain $i \dashleftarrow i-1$, which again contradicts the minimality of the cycle $(1, 2, \ldots, \ell, 1)$. Here we additionally distinguish the two cases: (i) $s'(P_i^*) \in V(P_{i+1}^*)$ and (ii) $s'(P_i^*) \notin V(P_{i+1}^*)$.

(i) $s'(P_i^*) \in V(P_{i+1}^*)$. Since P_i^* and P_{i+1}^* have no common internal vertices, we have $t(P_{i+1}^*) = s'(P_i^*)$. By the minimality of the cycle $(1, 2, \ldots, \ell, 1)$, P_{i-1}^* has none of vertices in $N_T[s'(P_i^*)]$. This implies, together with $i-1 \dashleftarrow i$ and $V(P_{i-1}^*) \cap C_{s(P_i^*)} = \emptyset$, that $V(P_{i-1}^*) \subseteq C_{x^*}$ and $x^* \notin V(P_{i-1}^*)$. By $i+1 \dashleftarrow i+2 \dashleftarrow \cdots \dashleftarrow i-1$, there is an index m with $i-1 \neq m \neq i+1$ such that $x^* \in V(P_m^*)$. This implies $m \dashleftarrow i+1$, contradicting the minimality of the cycle.

(ii) $s'(P_i^*) \notin V(P_{i+1}^*)$. In this case, P_{i+1}^* has a vertex in $N[s(P_i^*)] \setminus \{s'(P_i^*)\}$ and does not have the arc $(s(P_i^*), s'(P_i^*))$. Thus we have $V(P_{i+1}^*) \subseteq C_{s(P_i^*)}$. By $i+1 \dashleftarrow i+2 \dashleftarrow \cdots \dashleftarrow i-1$, there is an index m with $i-1 \neq m \neq i+1$ such that $s'(P_i^*) \in V(P_m^*)$. This implies $i \dashleftarrow m$, contradicting the minimality of the cycle.

This completes the proof of Lemma 14. ◀

4.4 Algorithms

In this subsection, we provide an algorithm for checking the reconfigurability in $\mathrm{O}(|V|)$ time, and that for constructing a reconfiguration sequence in $\mathrm{O}(k|V|)$ time, where k is the size of the input independent set. (if the answer is affirmative), proving Theorem 3.

For $U \subseteq V$, we define $N^+(U)$ (resp. $N^-(U)$) as $N^+(U) := \bigcup_{u \in U} N^+(u) \setminus U$ (resp. $N^-(U) := \bigcup_{u \in U} N^-(u) \setminus U$). For a mapping $f \colon I^{\mathsf{s}} \to N^+(I^{\mathsf{s}})$, let $f(I^{\mathsf{s}})$ denote the image of f, i.e., $f(I^{\mathsf{s}}) := \{f(x) \mid x \in I^{\mathsf{s}}\}$. Similarly, the image of $g \colon I^{\mathsf{t}} \to N^-(I^{\mathsf{t}})$ is denoted as $g(I^{\mathsf{t}})$.

4.4.1 Algorithm for checking the reconfigurability

Suppose that there is a set $\mathcal{P} = \{P_1, \dots, P_k\}$ of directed paths satisfying the path-set conditions. Then the set of paths obtained from $\mathcal{P}$ by exchanging each P_i with the subpath from $s'(P_i)$ to $t'(P_i)$ is a directed path matching from $\{s'(P_i) \mid i \in [k]\}$ to $\{t'(P_i) \mid i \in [k]\}$. In this case, the mappings $f\colon I^{\mathsf{s}} \to N^+(I^{\mathsf{s}})$ and $g\colon I^{\mathsf{t}} \to N^-(I^{\mathsf{t}})$ defined by $f(s(P_i)) := s'(P_i)$ and $g(t(P_i)) := t'(P_i)$, respectively, satisfy the following four conditions:

(C1) f and g are injective,
(C2) $(s, f(s)) \in A$ for all $s \in I^{\mathsf{s}}$,
(C3) $(g(t), t) \in A$ for all $t \in I^{\mathsf{t}}$, and
(C4) $w(e; f(I^{\mathsf{s}}), g(I^{\mathsf{t}})) \geq 0$ for each arc e.

In particular, (C1) follows from (P3) and (P4), and (C4) follows from Lemma 5.

Conversely, if there are mappings $f\colon I^{\mathsf{s}} \to N^+(I^{\mathsf{s}})$ and $g\colon I^{\mathsf{t}} \to N^-(I^{\mathsf{t}})$ satisfying the above four conditions (C1)–(C4), then we can construct a set of directed paths satisfying the path-set conditions (P1)–(P4) as follows. By Lemma 5 and (C4), there exists a directed path matching $\mathcal{P}' = \{P'_1, \dots, P'_k\}$ from $f(I^{\mathsf{s}})$ to $g(I^{\mathsf{t}})$. Define a set $\mathcal{P} = \{P_1, \dots, P_k\}$ of paths from $\mathcal{P}'$ by appending $f^{-1}(s(P'_i))$ and $g^{-1}(t(P'_i))$ before the source and after the sink for each P'_i, respectively. Then $|P_i| \geq 2$ for each $i \in [k]$, and $\mathcal{P}$ is a directed path matching from I^{s} and I^{t}. Moreover, since $\mathcal{P}'$ is a directed path matching from $f(I^{\mathsf{s}})$ to $g(I^{\mathsf{t}})$, we have $s'(P_i) \neq s'(P_j)$ and $t'(P_i) \neq t'(P_j)$ for $i \neq j$, implying that $\mathcal{P}$ satisfies the path-set conditions.

By the above argument, we obtain the following necessary and sufficient conditions for the reconfigurability:

▶ **Lemma 15.** *For an instance without rigid tokens and blocking arcs, there exists a set of paths satisfying the path-set conditions if and only if there exist mappings $f\colon I^{\mathsf{s}} \to N^+(I^{\mathsf{s}})$ and $g\colon I^{\mathsf{t}} \to N^-(I^{\mathsf{t}})$ satisfying* (C1)–(C4).

By Lemmas 12 and 15, for checking the reconfigurability it suffices to compute f and g satisfying (C1)–(C4) or determine that such f and g do not exist; it can be done in $\mathrm{O}(|V|)$ time in a greedy manner as follows. In the following description, we regard T as a rooted tree with an arbitrary root. We initialize the values of f and g as $f(s) := \bot$ for all $s \in I^{\mathsf{s}}$ and $g(t) := \bot$ for all $t \in I^{\mathsf{t}}$ ($\bot$ means "undefined"). To let f and g satisfy the conditions in Lemma 15, we define each value of f and g one by one. We write $I^{\mathsf{s}}_{\mathrm{undef}} \subseteq I^{\mathsf{s}}$ (resp. $I^{\mathsf{s}}_{\mathrm{def}} \subseteq I^{\mathsf{s}}$) as the set of the vertices for which the values of f have been undefined (resp. defined). We also define $I^{\mathsf{t}}_{\mathrm{undef}}$ and $I^{\mathsf{t}}_{\mathrm{def}}$ in the same way. Throughout the following process to determine f and g, we keep an invariant that the conditions (C1)–(C3) hold for $s \in I^{\mathsf{s}}_{\mathrm{def}}$ and $t \in I^{\mathsf{t}}_{\mathrm{def}}$, moreover, for each arc e, $w(e; I^{\mathsf{s}}_{\mathrm{undef}} \cup f(I^{\mathsf{s}}_{\mathrm{def}}), I^{\mathsf{t}}_{\mathrm{undef}} \cup g(I^{\mathsf{t}}_{\mathrm{def}})) \geq 0$.

In the following, we describe each step of the algorithm. Let $v^* \in I^{\mathsf{s}}_{\mathrm{undef}} \cup I^{\mathsf{t}}_{\mathrm{undef}}$ be a vertex with the largest depth among $I^{\mathsf{s}}_{\mathrm{undef}} \cup I^{\mathsf{t}}_{\mathrm{undef}}$ in the rooted tree. If $v^* \in I^{\mathsf{s}}_{\mathrm{undef}}$, we define

$$N_{\mathrm{cand}}(v^*) := \left\{ u \in N^+(v^*) \setminus f(I^{\mathsf{s}}_{\mathrm{def}}) \;\middle|\; w((v^*, u); I^{\mathsf{s}}_{\mathrm{undef}} \cup f(I^{\mathsf{s}}_{\mathrm{def}}), I^{\mathsf{t}}_{\mathrm{undef}} \cup g(I^{\mathsf{t}}_{\mathrm{def}})) \geq 1 \right\}.$$

Otherwise, i.e., $v^* \in I^{\mathsf{t}}_{\mathrm{undef}} \setminus I^{\mathsf{s}}_{\mathrm{undef}}$, define

$$N_{\mathrm{cand}}(v^*) := \left\{ u \in N^-(v^*) \setminus g(I^{\mathsf{t}}_{\mathrm{def}}) \;\middle|\; w((u, v^*); I^{\mathsf{s}}_{\mathrm{undef}} \cup f(I^{\mathsf{s}}_{\mathrm{def}}), I^{\mathsf{t}}_{\mathrm{undef}} \cup g(I^{\mathsf{t}}_{\mathrm{def}})) \geq 1 \right\}.$$

If $N_{\mathrm{cand}}(v^*)$ is empty, terminate the execution. Otherwise, choose $u^* \in N_{\mathrm{cand}}(v^*)$ with the largest depth among $N_{\mathrm{cand}}(v^*)$ and define $f(v^*) := u^*$ if $v^* \in I^{\mathsf{s}}_{\mathrm{undef}}$ and define $g(v^*) := u^*$ otherwise. When $f(v^*)$ or $g(v^*)$ is newly defined in this step, we accordingly update $I^{\mathsf{s}}_{\mathrm{def}}$, $I^{\mathsf{s}}_{\mathrm{undef}}$, $I^{\mathsf{t}}_{\mathrm{def}}$, and $I^{\mathsf{t}}_{\mathrm{undef}}$.

Note that each vertex in $I^{\mathsf{s}} \cup I^{\mathsf{t}}$ is selected as v^* at most twice. After the execution of the above algorithm, output f and g if $f(s) \neq \perp$ for all $s \in I^{\mathsf{s}}$ and $g(t) \neq \perp$ for all $t \in I^{\mathsf{t}}$; otherwise, we conclude that no such f and g exist. In each iteration, only one arc (v^*, u^*) or (u^*, v^*) decreases its w-value by 1: If $v^* \in I^{\mathsf{s}}_{\mathrm{undef}}$,

$$\begin{aligned}&w(e; I^{\mathsf{s}}_{\mathrm{undef}} \setminus \{v^*\} \cup f(I^{\mathsf{s}}_{\mathrm{def}}) \cup \{u^*\}, I^{\mathsf{t}}_{\mathrm{undef}} \cup g(I^{\mathsf{t}}_{\mathrm{def}}))\\ &= \begin{cases} w(e; I^{\mathsf{s}}_{\mathrm{undef}} \cup f(I^{\mathsf{s}}_{\mathrm{def}}), I^{\mathsf{t}}_{\mathrm{undef}} \cup g(I^{\mathsf{t}}_{\mathrm{def}})) & \text{if } e \neq (v^*, u^*)\\ w(e; I^{\mathsf{s}}_{\mathrm{undef}} \cup f(I^{\mathsf{s}}_{\mathrm{def}}), I^{\mathsf{t}}_{\mathrm{undef}} \cup g(I^{\mathsf{t}}_{\mathrm{def}})) - 1 & \text{if } e = (v^*, u^*);\end{cases}\end{aligned}$$

and otherwise $w(e; I^{\mathsf{s}}_{\mathrm{undef}} \cup f(I^{\mathsf{s}}_{\mathrm{def}}), I^{\mathsf{t}}_{\mathrm{undef}} \setminus \{v^*\} \cup g(I^{\mathsf{t}}_{\mathrm{def}}) \cup \{u^*\})$ can be computed in a symmetric fashion. Using this property, we can implement the algorithm that runs in $\mathrm{O}(|V|)$ time. More precisely, we maintain the values of w and the indicator functions of $I^{\mathsf{s}}_{\mathrm{def}}$ and $I^{\mathsf{t}}_{\mathrm{def}}$ in tables. By referring the tables, we can compute $N_{\mathrm{cand}}(v^*)$ in $\mathrm{O}(|N(v^*)|)$ time. Moreover, we can update the tables in $\mathrm{O}(1)$ time. Therefore each iteration runs in $\mathrm{O}(|N(v^*)|)$ time, and thus the algorithm runs in total in $\mathrm{O}(|V|)$ time since each edge is touched twice.

▶ **Lemma 16** ($\star$). *If there exist mappings f and g that satisfy the conditions in Lemma 15, then the algorithm described above correctly outputs one of such mappings. Otherwise the algorithm reports that no such mappings exist.*

4.4.2 Algorithm for constructing a reconfiguration sequence

By Corollary 6, if the instance is a yes-instance, all reconfiguration sequences have the same length. In this subsection, we present an algorithm to construct one of them in $\mathrm{O}(k|V|)$ time, where k is the size of the input independent set. We first assume that two mappings f and g satisfying conditions (C1)–(C4) have already been computed. If such f and g do not exist, the instance is a no-instance by Lemma 15. Here, our target is to fill in gaps between $f(I^{\mathsf{s}})$ and $g(I^{\mathsf{t}})$ avoiding biased path pairs. For preparation of further arguments, we define *weakly biased path pairs* by relaxing “common internal vertex” to “common vertex” in the definition of biased path pairs. If there is a directed path matching $\mathcal{P} = \{P_1, \ldots, P_k\}$ from $f(I^{\mathsf{s}})$ to $g(I^{\mathsf{t}})$ without any weakly biased path pairs (not necessarily satisfying the path-set conditions), we can easily construct a directed path matching $\mathcal{P}' = \{P'_1, \ldots, P'_k\}$ from I^{s} to I^{t} satisfying the path-set conditions: P'_i is obtained from P_i by appending $f^{-1}(s(P_i))$ before P_i and $g^{-1}(t(P_i))$ after P_i. Thus, in the following, it suffices to show that $\mathcal{P}$ can be constructed in $\mathrm{O}(k|V|)$ time.

Fix an arbitrary vertex $r \in V$. For each $v \in V$, we denote by $P_{r,v}$ the unique path between r and v in T^{und}. Let $\overrightarrow{P_{r,v}}$ be a directed path obtained by orienting each edge of $P_{r,v}$ from r to v. We define $d_r(v) = \mathrm{fwd}(\overrightarrow{P_{r,v}}) - \mathrm{rev}(\overrightarrow{P_{r,v}})$, where $\mathrm{fwd}(\overrightarrow{P_{r,v}})$ and $\mathrm{rev}(\overrightarrow{P_{r,v}})$ denote the numbers of arcs in T that $\overrightarrow{P_{r,v}}$ passes in the forward and reverse directions, respectively.

Using the values of $d_r(v)$ for $v \in V$, we iteratively construct a directed path matching $\mathcal{P} = \{P_1, \ldots, P_k\}$ from $f(I^{\mathsf{s}})$ to $g(I^{\mathsf{t}})$ as follows. In the i-th step of the algorithm, we construct P_i. Let $X_i \subseteq f(I^{\mathsf{s}})$ and $Y_i \subseteq g(I^{\mathsf{t}})$ be the sets of vertices that are not chosen for the sources and sinks of $P_1, \ldots, P_{i-1}$, respectively. Throughout the execution of the algorithm, we keep an invariant that for each arc e, $w(e; X_i, Y_i) \geq 0$, which means that there is a directed path matching from X_i to Y_i by Lemma 5.

Let $x \in X_i$ be a vertex with the smallest value of $d_r(x)$ among X_i. Let $Y'_i \subseteq Y_i$ be the set of vertices to which there is a directed path from x consisting of only arcs with $w(e; X_i, Y_i) > 0$. Since there exists some directed path matching from X_i to Y_i by our invariant that $w(e, X_i, Y_i) \geq 0$ for $e \in A$, the set Y'_i is not empty. Choose an arbitrary vertex $y \in Y_i'$ with the smallest value of $d_r(y)$. We set P_i to the unique directed path from x to y in T. Since we use only arcs e with $w(e; X_i, Y_i) > 0$ to construct P_i, we have $w(e; X_{i+1}, Y_{i+1}) \geq 0$, where $X_{i+1} = X_i \setminus \{x\}$ and $Y_{i+1} = Y_i \setminus \{y\}$, for all arcs and the invariant still holds. We repeat this until $X_i = \emptyset$ (and hence $Y_i = \emptyset$).

Here, we show that the obtained directed path matching $\{P_1, \ldots, P_k\}$ does not contain any weakly biased path pairs. Assume that P_i and P_j $(i < j)$ have a common vertex v. By the choice of x in the construction above, $d_r(s(P_i)) \leq d_r(s(P_j))$ holds. Thus, we have

$$\mathrm{dist}(s(P_i), v) = -d_r(s(P_i)) + d_r(v) \geq -d_r(s(P_j)) + d_r(v) = \mathrm{dist}(s(P_j), v).$$

As v is a common vertex of P_i and P_j, $t(P_j) \in Y_i'$ and thus $d_r(t(P_i)) \leq d_r(t(P_j))$ holds by the choice of y. Then, similarly we have

$$\mathrm{dist}(v, t(P_i)) = -d_r(v) + d_r(t(P_i)) \leq -d_r(v) + d_r(t(P_j)) = \mathrm{dist}(v, t(P_j)).$$

Therefore, P_i and P_j do not form a weakly biased path pair. We can compute each P_i in $\mathrm{O}(|V|)$ time, and thus the whole running time (including computing f and g) is $\mathrm{O}(k|V|)$.

References

1 Oswin Aichholzer, Jean Cardinal, Tony Huynh, Kolja Knauer, Torsten Mütze, Raphael Steiner, and Birgit Vogtenhuber. Flip distances between graph orientations. *Algorithmica*, 83:116–143, 2021. doi:10.1007/s00453-020-00751-1.

2 Valentin Bartier, Nicolas Bousquet, Clément Dallard, Kyle Lomer, and Amer E. Mouawad. On girth and the parameterized complexity of token sliding and token jumping. In *Proceedings of the 31st International Symposium on Algorithms and Computation (ISAAC 2020)*, volume 181 of *LIPIcs*, pages 44:1–44:17, 2020. doi:10.4230/LIPIcs.ISAAC.2020.44.

3 Rémy Belmonte, Eun Jung Kim, Michael Lampis, Valia Mitsou, Yota Otachi, and Florian Sikora. Token sliding on split graphs. *Theory of Computing Systems*, 65:662–686, 2021. doi:10.1007/s00224-020-09967-8.

4 Marthe Bonamy and Nicolas Bousquet. Token sliding on chordal graphs. In *Proceedings of the 43rd International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2017)*, volume 10520 of *LNCS*, pages 127–139, 2017. doi:10.1007/978-3-319-68705-6_10.

5 Marthe Bonamy, Nicolas Bousquet, Marc Heinrich, Takehiro Ito, Yusuke Kobayashi, Arnaud Mary, Moritz Mühlenthaler, and Kunihiro Wasa. The perfect matching reconfiguration problem. In *Proceedings of the 44th International Symposium on Mathematical Foundations of Computer Science (MFCS 2019)*, volume 138 of *LIPIcs*, pages 80:1–80:14, 2019. doi:10.4230/LIPIcs.MFCS.2019.80.

6 Marthe Bonamy, Marc Heinrich, Takehiro Ito, Yusuke Kobayashi, Haruka Mizuta, Moritz Mühlenthaler, Akira Suzuki, and Kunihiro Wasa. Diameter of colorings under Kempe changes. *Theoretical Computer Science*, 838:45–57, 2020. doi:10.1016/j.tcs.2020.05.033.

7 Paul S. Bonsma and Luis Cereceda. Finding paths between graph colourings: PSPACE-completeness and superpolynomial distances. *Theoretical Computer Science*, 410(50):5215–5226, 2009. doi:10.1016/j.tcs.2009.08.023.

8 Paul S. Bonsma, Marcin Kaminski, and Marcin Wrochna. Reconfiguring independent sets in claw-free graphs. In *Proceedings of the 14th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2014)*, volume 8503 of *LNCS*, pages 86–97, 2014. doi:10.1007/978-3-319-08404-6_8.

9 Nicolas Bousquet, Tatsuhiko Hatanaka, Takehiro Ito, and Moritz Mühlenthaler. Shortest reconfiguration of matchings. In *Proceedings of the 45th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2019)*, volume 11789 of *LNCS*, pages 162–174, 2019. doi:10.1007/978-3-030-30786-8_13.

10 Nicolas Bousquet, Arnaud Mary, and Aline Parreau. Token jumping in minor-closed classes. In *Proceedings of the 21st International Symposium on Fundamentals of Computation Theory (FCT 2017)*, volume 10472 of *LNCS*, pages 136–149, 2017. doi:10.1007/978-3-662-55751-8_12.

11 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Daniel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer Publishing Company, Incorporated, 1st edition, 2015.

12 Erik D. Demaine, Martin L. Demaine, Eli Fox-Epstein, Duc A. Hoang, Takehiro Ito, Hirotaka Ono, Yota Otachi, Ryuhei Uehara, and Takeshi Yamada. Linear-time algorithm for sliding tokens on trees. *Theoretical Computer Science*, 600:132–142, 2015. doi:10.1016/j.tcs.2015.07.037.

13 Eli Fox-Epstein, Duc A. Hoang, Yota Otachi, and Ryuhei Uehara. Sliding token on bipartite permutation graphs. In *Proceedings of the 26th International Symposium on Algorithms and Computation (ISAAC 2015)*, volume 9472 of *LNCS*, pages 237–247, 2015. doi:10.1007/978-3-662-48971-0_21.

14 Komei Fukuda, Alain Prodon, and Tadashi Sakuma. Notes on acyclic orientations and the shelling lemma. *Theoretical Computer Science*, 263(1-2):9–16, 2001. doi:10.1016/S0304-3975(00)00226-7.

15 Curtis Greene and Thomas Zaslavsky. On the interpretation of Whitney numbers through arrangements of hyperplanes, zonotopes, non-Radon partitions, and orientations of graphs. *Transactions of the American Mathematical Society*, 280(1):97–126, 1983.

16 Arash Haddadan, Takehiro Ito, Amer E. Mouawad, Naomi Nishimura, Hirotaka Ono, Akira Suzuki, and Youcef Tebbal. The complexity of dominating set reconfiguration. *Theoretical Computer Science*, 651:37–49, 2016. doi:10.1016/j.tcs.2016.08.016.

17 Robert A. Hearn and Erik D. Demaine. PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theoretical Computer Science*, 343(1-2):72–96, 2005. doi:10.1016/j.tcs.2005.05.008.

18 Duc A. Hoang and Ryuhei Uehara. Sliding tokens on a cactus. In *Proceedings of the 27th International Symposium on Algorithms and Computation (ISAAC 2016)*, volume 64 of *LIPIcs*, pages 37:1–37:26, 2016. doi:10.4230/LIPIcs.ISAAC.2016.37.

19 Takehiro Ito, Erik D. Demaine, Nicholas J. A. Harvey, Christos H. Papadimitriou, Martha Sideri, Ryuhei Uehara, and Yushi Uno. On the complexity of reconfiguration problems. *Theoretical Computer Science*, 412(12-14):1054–1065, 2011. doi:10.1016/j.tcs.2010.12.005.

20 Takehiro Ito, Yuni Iwamasa, Naonori Kakimura, Naoyuki Kamiyama, Yusuke Kobayashi, Shun ichi Maezawa, Yuta Nozaki, Yoshio Okamoto, and Kenta Ozeki. Monotone edge flips to an orientation of maximum edge-connectivity à la Nash-Williams. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms (SODA 2022)*, pages 1342–1355, 2022.

21 Takehiro Ito, Yuni Iwamasa, Yasuaki Kobayashi, Yu Nakahata, Yota Otachi, Masahiro Takahashi, and Kunihiro Wasa. Independent set reconfiguration on directed graphs. *CoRR*, abs/2203.13435, 2022. doi:10.48550/arXiv.2203.13435.

22 Takehiro Ito, Yuni Iwamasa, Yasuaki Kobayashi, Yu Nakahata, Yota Otachi, and Kunihiro Wasa. Reconfiguring directed trees in a digraph. In *Proceedings of the 27th International Computing and Combinatorics Conference (COCOON 2021)*, volume 13025 of *LNCS*, pages 343–354, 2021.

23 Takehiro Ito, Naonori Kakimura, Naoyuki Kamiyama, Yusuke Kobayashi, and Yoshio Okamoto. Shortest reconfiguration of perfect matchings via alternating cycles. In *Proceedings of the 27th Annual European Symposium on Algorithms (ESA 2019)*, volume 144 of *LIPIcs*, pages 61:1–61:15, 2019. doi:10.4230/LIPIcs.ESA.2019.61.

24 Takehiro Ito, Marcin Kaminski, Hirotaka Ono, Akira Suzuki, Ryuhei Uehara, and Katsuhisa Yamanaka. On the parameterized complexity for token jumping on graphs. In *Proceedings of the 11th Annual Conference on Theory and Applications of Models of Computation (TAMC 2014)*, volume 8402 of *LNCS*, pages 341–351, 2014. doi:10.1007/978-3-319-06089-7_24.

25 Takehiro Ito, Marcin Jakub Kaminski, and Hirotaka Ono. Fixed-parameter tractability of token jumping on planar graphs. In *Proceedings of the 25th International Symposium on Algorithms and Computation (ISAAC 2014)*, volume 8889 of *LNCS*, pages 208–219, 2014. doi:10.1007/978-3-319-13075-0_17.

26 Takehiro Ito, Marcin Jakub Kaminski, Hirotaka Ono, Akira Suzuki, Ryuhei Uehara, and Katsuhisa Yamanaka. Parameterized complexity of independent set reconfiguration problems. *Discrete Applied Mathematics*, 283:336–345, 2020. doi:10.1016/j.dam.2020.01.022.

27 Takehiro Ito, Hirotaka Ono, and Yota Otachi. Reconfiguration of cliques in a graph. In *Proceedings of the 12th Annual Conference on Theory and Applications of Models of Computation (TAMC 2015)*, volume 9076 of *LNCS*, pages 212–223, 2015. doi:10.1007/978-3-319-17142-5_19.

28 Marcin Kaminski, Paul Medvedev, and Martin Milanic. Complexity of independent set reconfigurability problems. *Theoretical Computer Science*, 439:9–15, 2012. doi:10.1016/j.tcs.2012.03.004.

29 Daniel Lokshtanov and Amer E. Mouawad. The complexity of independent set reconfiguration on bipartite graphs. *ACM Transactions on Algorithms*, 15(1):7:1–7:19, 2019. doi:10.1145/3280825.

30 Daniel Lokshtanov, Amer E. Mouawad, Fahad Panolan, M. S. Ramanujan, and Saket Saurabh. Reconfiguration on sparse graphs. *Journal of Computer and System Sciences*, 95:122–131, 2018. doi:10.1016/j.jcss.2018.02.004.

31 Naomi Nishimura. Introduction to reconfiguration. *Algorithms*, 11(4):52, 2018. doi:10.3390/a11040052.

32 Hiroki Osawa, Akira Suzuki, Takehiro Ito, and Xiao Zhou. Algorithms for coloring reconfiguration under recolorability constraints. In *Proceedings of the 29th International Symposium on Algorithms and Computation (ISAAC 2018)*, volume 123 of *LIPIcs*, pages 37:1–37:13, 2018. doi:10.4230/LIPIcs.ISAAC.2018.37.

33 Ken Sugimori. A polynomial-time algorithm for shortest reconfiguration of sliding tokens on a tree. Master's thesis, The University of Tokyo, 2019. (In Japanese).

34 Akira Suzuki, Amer E. Mouawad, and Naomi Nishimura. Reconfiguration of dominating sets. *Journal of Combinatorial Optimization*, 32(4):1182–1195, 2016. doi:10.1007/s10878-015-9947-x.

35 Jan van den Heuvel. The complexity of change. In *Surveys in Combinatorics 2013*, volume 409 of *London Mathematical Society Lecture Note Series*, pages 127–160. Cambridge University Press, 2013. doi:10.1017/CBO9781139506748.005.