Linear-Time Algorithm for Sliding Tokens on Trees

Erik D. Demaine[a], Martin L. Demaine[a], Eli Fox-Epstein[b], Duc A. Hoang[c], Takehiro Ito[d,e], Hirotaka Ono[f], Yota Otachi[c], Ryuhei Uehara[c], Takeshi Yamada[c]

[a] *MIT Computer Science and Artificial Intelligence Laboratory, 32 Vassar St., Cambridge, MA 02139, USA*
[b] *Department of Computer Science, Brown University, 115 Waterman Street, Providence, RI 02912-1910, USA*
[c] *School of Information Science, Japan Advanced Institute of Science and Technology, Asahidai 1-1, Nomi, Ishikawa 923-1292, Japan*
[d] *Graduate School of Information Sciences, Tohoku University, Aoba-yama 6-6-05, Sendai, 980-8579, Japan*
[e] *CREST, JST, 4-1-8 Honcho, Kawaguchi, Saitama, 332-0012, Japan*
[f] *Faculty of Economics, Kyushu University, Hakozaki 6-19-1, Higashi-ku, Fukuoka, 812-8581, Japan*

Abstract

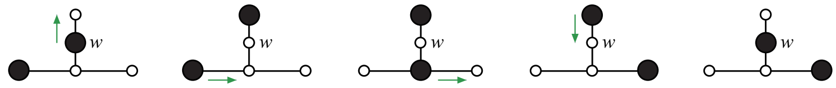
Suppose that we are given two independent sets I_b and I_r of a graph such that $|I_b| = |I_r|$, and imagine that a token is placed on each vertex in I_b. Then, the SLIDING TOKEN problem is to determine whether there exists a sequence of independent sets which transforms I_b into I_r so that each independent set in the sequence results from the previous one by sliding exactly one token along an edge in the graph. This problem is known to be PSPACE-complete even for planar graphs, and also for bounded treewidth graphs. In this paper, we thus study the problem restricted to trees, and give the following three results: (1) the decision problem is solvable in linear time; (2) for a yes-instance, we can find in quadratic time an actual sequence of independent sets between I_b and I_r whose length (i.e., the number of token-slides) is quadratic; and (3) there exists an infinite family of instances on paths for which any sequence requires quadratic length.

Keywords: combinatorial reconfiguration, graph algorithm, independent set, sliding token, tree

1. Introduction

Recently, *reconfiguration problems* have attracted the attention in the field of theoretical computer science. The problem arises when we wish to find a step-by-step transformation between two feasible solutions of a problem such that all intermediate results are also feasible and each step conforms to a fixed reconfiguration rule (i.e., an adjacency relation defined on feasible solutions of the original problem). This kind of reconfiguration problem has been studied

Email addresses: edemaine@mit.edu (Erik D. Demaine), mdemaine@mit.edu (Martin L. Demaine), ef@cs.brown.edu (Eli Fox-Epstein), hoanganhduc@jaist.ac.jp (Duc A. Hoang), takehiro@ecei.tohoku.ac.jp (Takehiro Ito), hirotaka@econ.kyushu-u.ac.jp (Hirotaka Ono), otachi@jaist.ac.jp (Yota Otachi), uehara@jaist.ac.jp (Ryuhei Uehara), tyama@jaist.ac.jp (Takeshi Yamada)



(a) $I_b = I_1$ (b) I_2 (c) I_3 (d) I_4 (e) $I_r = I_5$

Figure 1: A sequence $\langle I_1, I_2, \ldots, I_5 \rangle$ of independent sets of the same graph, where the vertices in independent sets are depicted by large black circles (tokens).

extensively for several well-known problems, including INDEPENDENT SET [2, 5, 7, 11, 12, 14, 16, 20, 22, 23, 25], SATISFIABILITY [10, 21], SET COVER, CLIQUE, MATCHING [14], VERTEX-COLORING [3, 6, 8, 25], LIST EDGE-COLORING [15, 18], LIST $L(2, 1)$-LABELING [17], SUBSET SUM [13], SHORTEST PATH [4, 19], and so on. (See also a recent survey [24].)

1.1. SLIDING TOKEN

The SLIDING TOKEN problem was introduced by Hearn and Demaine [11] as a one-player game, which can be seen as a reconfiguration problem for INDEPENDENT SET. Recall that an *independent set* of a graph G is a vertex subset of G in which no two vertices are adjacent. (Figure 1 depicts five different independent sets in the same graph.) Suppose that we are given two independent sets I_b and I_r of a graph $G = (V, E)$ such that $|I_b| = |I_r|$, and imagine that a token (coin) is placed on each vertex in I_b. Then, the SLIDING TOKEN problem is to determine whether there exists a sequence $\langle I_1, I_2, \ldots, I_\ell \rangle$ of independent sets of G such that

(a) $I_1 = I_b$, $I_\ell = I_r$, and $|I_i| = |I_b| = |I_r|$ for all i, $1 \leq i \leq \ell$; and

(b) for each i, $2 \leq i \leq \ell$, there is an edge $\{u, v\}$ in G such that $I_{i-1} \setminus I_i = \{u\}$ and $I_i \setminus I_{i-1} = \{v\}$, that is, I_i can be obtained from I_{i-1} by sliding exactly one token on a vertex $u \in I_{i-1}$ to its adjacent vertex v along $\{u, v\} \in E$.

Such a sequence is called a *reconfiguration sequence* between I_b and I_r. Figure 1 illustrates a reconfiguration sequence $\langle I_1, I_2, \ldots, I_5 \rangle$ of independent sets which transforms $I_b = I_1$ into $I_r = I_5$. Hearn and Demaine proved that SLIDING TOKEN is PSPACE-complete for planar graphs, as an example of the application of their tool, called the nondeterministic constraint logic model, which can be used to prove PSPACE-hardness of many puzzles and games [11], [12, Sec. 9.5].

1.2. Related and known results

As the (ordinary) INDEPENDENT SET problem is a key problem among thousands of NP-complete problems, SLIDING TOKEN plays an important role since several PSPACE-hardness results have been proved using reductions from it. In addition, reconfiguration problems for INDEPENDENT SET (ISRECONF, for short) have been studied under different reconfiguration rules, as follows.

- *Token Sliding* (TS rule) [6, 7, 11, 12, 20, 25]: This rule corresponds to SLIDING TOKEN, that is, we can slide a single token only along an edge of a graph.

(a) I_b (b) I_r

Figure 2: Two distinct independent sets I_b and I_r of the same star. This is a yes-instance for ISRECONF under the TJ rule, but is a no-instance for the SLIDING TOKEN problem.

- *Token Jumping* (TJ rule) [7, 16, 20, 25]: A single token can "jump" to any vertex (including a non-adjacent one) if it results in an independent set.
- *Token Addition and Removal* (TAR rule) [2, 5, 14, 20, 22, 23, 25]: We can either add or remove a single token at a time if it results in an independent set of cardinality at least a given threshold. Therefore, under the TAR rule, independent sets in the sequence do not have the same cardinality.

We note that the existence of a desired sequence depends deeply on the reconfiguration rules. (See Figure 2 for example.) However, ISRECONF is PSPACE-complete under any of the three reconfiguration rules for planar graphs [6, 11, 12], for perfect graphs [20], and for bounded bandwidth graphs [25]. The PSPACE-hardness implies that, unless NP = PSPACE, there exists an instance of SLIDING TOKEN which requires a super-polynomial number of token-slides even in a minimum-length reconfiguration sequence. In such a case, tokens should make "detours" to avoid violating independence. (For example, see the token placed on the vertex w in Figure 1(a); it is moved twice even though $w \in I_b \cap I_r$.)

We here explain only the results which are strongly related to this paper, that is, SLIDING TOKEN on trees; see the references above for the other results.

1.2.1. Results for TS rule (SLIDING TOKEN)

Kamiński et al. [20] gave a linear-time algorithm to solve SLIDING TOKEN for cographs (also known as P_4-free graphs). They also showed that, for any yes-instance on cographs, two given independent sets I_b and I_r have a reconfiguration sequence such that no token makes a detour.

Very recently, Bonsma et al. [7] proved that SLIDING TOKEN can be solved in polynomial time for claw-free graphs. Note that neither cographs nor claw-free graphs contain trees as a (proper) subclass. Thus, the complexity status for trees was open under the TS rule.

1.2.2. Results for trees

In contrast to the TS rule, it is known that ISRECONF can be solved in linear time under the TJ and TAR rules for even-hole-free graphs [20], which include trees. Indeed, the answer is always "yes" under the two rules when restricted to even-hole-free graphs (as long as two given independent sets have the same cardinality for the TJ rule.) Furthermore, tokens never make detours in even-hole-free graphs under the TJ and TAR rules.

On the other hand, under the TS rule, tokens are required to make detours even in trees. (See Figure 1.) In addition, there are no-instances for trees under the TS rule. (See Figure 2.) These make the problem much more complicated, and we think they are the main reasons why SLIDING TOKEN for trees was unsolved, even though this is certainly a natural question under the recent intensive algorithmic research on ISRECONF [2, 5, 7, 16, 20, 23].

1.3. Our contribution

In this paper, we first prove that the SLIDING TOKEN problem is solvable in $O(n)$ time for any tree T with n vertices. Therefore, we can conclude that ISRECONF for trees is in P (indeed, solvable in linear time) under any of the three reconfiguration rules.

It is remarkable that there exists an infinite family of instances on paths for which any reconfiguration sequence requires $\Omega(n^2)$ length, although we can decide if it is a yes-instance in $O(n)$ time. For example, consider a path $(v_1, v_2, \ldots, v_{8k})$ with $n = 8k$ vertices for any positive integer k, and let $I_b = \{v_1, v_3, v_5, \ldots, v_{2k-1}\}$ and $I_r = \{v_{6k+2}, v_{6k+4}, \ldots, v_{8k}\}$. In this yes-instance, any token must be slid $\Theta(n)$ times, and hence any reconfiguration sequence requires $\Theta(n^2)$ length to slide them all. As the second result of this paper, we give an $O(n^2)$-time algorithm which finds an actual reconfiguration sequence of length $O(n^2)$ between two given independent sets for a yes-instance.

Since the treewidth of any graph G can be bounded by the bandwidth of G, the result of [25] implies that SLIDING TOKEN is PSPACE-complete for bounded treewidth graphs. (See [1] for the definition of treewidth.) Thus, there exists an instance on bounded treewidth graphs which requires a super-polynomial number of token-slides even in a minimum-length reconfiguration sequence unless NP = PSPACE. Therefore, it is interesting that any yes-instance on a tree, whose treewidth is one, has an $O(n^2)$-length reconfiguration sequence even though trees require detours for transformations.

An early version of the paper has been presented in [9]. However, we note that the running time of our algorithm was improved from quadratic [9] to linear.

1.4. Technical overview

We here explain our main ideas; formal descriptions will be given later.

We say that a token on a vertex v is "rigid" under an independent set I of a tree T if it cannot be slid at all, that is, $v \in I'$ holds for *any* independent set I' of T which is reconfigurable from I. (For example, the four tokens in Figure 2 are rigid.) Our algorithm is based on the following two key points.

(1) In Lemma 1, we will give a simple but non-trivial characterization of rigid tokens, based on which we can find all rigid tokens of two given independent sets I_b and I_r in $O(n)$ time. Note that, if I_b and I_r have different placements of rigid tokens, then it is a no-instance (Observation 1).

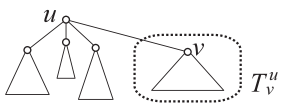


Figure 3: Subtree T_v^u in the whole tree T.

(2) Otherwise, we obtain a forest by deleting the vertices with rigid tokens together with their neighbors (Lemma 5). We will prove in Lemma 6 that the answer is "yes" as long as each tree in the forest contains the same number of tokens in I_b and I_r.

2. Preliminaries

In this section, we introduce some basic terms and notation.

2.1. Graph notation

In the SLIDING TOKEN problem, we may assume without loss of generality that graphs are simple and connected. For a graph G, we sometimes denote by $V(G)$ and $E(G)$ the vertex set and edge set of G, respectively.

In a graph G, a vertex w is said to be a *neighbor* of a vertex v if $\{v, w\} \in E(G)$. For a vertex v in G, let $N(G, v) = \{w \in V(G) \mid \{v, w\} \in E(G)\}$, and let $N[G, v] = N(G, v) \cup \{v\}$. For a subset $S \subseteq V(G)$, we simply write $N[G, S] = \bigcup_{v \in S} N[G, v]$. For a vertex v of G, we denote by $\deg_G(v)$ the degree of v in G, that is, $\deg_G(v) = |N(G, v)|$. For a subgraph G' of a graph G, we denote by $G \setminus G'$ the subgraph of G induced by the vertices in $V(G) \setminus V(G')$.

Let T be a tree. For two vertices v and w in T, the unique path between v and w is simply called the *vw-path* in T. We denote by $\mathsf{dist}(v, w)$ the number of edges in the vw-path in T. For two adjacent (and hence distinct) vertices u and v of a tree T, let T_v^u be the subtree of T obtained by regarding u as the root of T and then taking the subtree rooted at v which consists of v and all descendants of v. (See Figure 3.) It should be noted that u is not contained in the subtree T_v^u.

2.2. Definitions for SLIDING TOKEN

Let I_i and I_j be two independent sets of a graph G such that $|I_i| = |I_j|$. If there exists exactly one edge $\{u, v\}$ in G such that $I_i \setminus I_j = \{u\}$ and $I_j \setminus I_i = \{v\}$, then we say that I_j can be obtained from I_i by *sliding* the token on $u \in I_i$ to its adjacent vertex v along the edge $\{u, v\}$, and denote it by $I_i \leftrightarrow I_j$. We note that the tokens are unlabeled, while the vertices in a graph are labeled. We sometimes omit saying (the label of) the vertex on which a token is placed, and simply say "a token in an independent set I."

A *reconfiguration sequence* between two independent sets I_1 and I_ℓ of G is a sequence $\langle I_1, I_2, \ldots, I_\ell \rangle$ of independent sets of G such that $I_{i-1} \leftrightarrow I_i$ for $i = 2, 3, \ldots, \ell$. We sometimes write $I \in \mathcal{S}$ if an independent set I of G appears in the

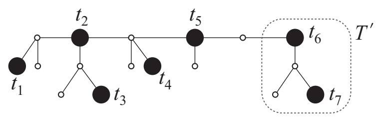


Figure 4: An independent set I of a tree T, where t_1, t_2, t_3, t_4 are (T, I)-rigid tokens and t_5, t_6, t_7 are (T, I)-movable tokens. For the subtree T', tokens t_6, t_7 are $(T', I \cap T')$-rigid.

reconfiguration sequence $\mathcal{S}$. We write $I_1 \overset{G}{\leftrightsquigarrow} I_\ell$ if there exists a reconfiguration sequence $\mathcal{S}$ between I_1 and I_ℓ such that all independent sets $I \in \mathcal{S}$ satisfy $I \subseteq V(G)$; we here define the notation emphasized with the graph G, because we will apply this notation to a subgraph of G. Note that any reconfiguration sequence is *reversible*, that is, $I_1 \overset{G}{\leftrightsquigarrow} I_\ell$ if and only if $I_\ell \overset{G}{\leftrightsquigarrow} I_1$. The *length* of a reconfiguration sequence $\mathcal{S}$ is defined as the number of independent sets contained in $\mathcal{S}$. For example, the length of the reconfiguration sequence in Figure 1 is 5.

Given two independent sets I_b and I_r of a graph G, the SLIDING TOKEN problem is to determine whether $I_b \overset{G}{\leftrightsquigarrow} I_r$ or not. We may assume without loss of generality that $|I_b| = |I_r|$; otherwise the answer is clearly “no.” Note that SLIDING TOKEN is a decision problem asking for the existence of a reconfiguration sequence between I_b and I_r, and hence it does not ask for an actual reconfiguration sequence. We always denote by I_b and I_r the *initial* and *target* independent sets of G, respectively.

3. Algorithm for Trees

In this section, we give the main result of this paper.

Theorem 1. *The* SLIDING TOKEN *problem can be solved in linear time for trees.*

As a proof of Theorem 1, we give an $O(n)$-time algorithm which solves SLIDING TOKEN for a tree with n vertices.

3.1. Rigid tokens

In this subsection, we formally define the concept of rigid tokens, and give their nice characterization.

Let T be a tree, and let I be an independent set of T. We say that a token on a vertex $v \in I$ is (T, I)-*rigid* if $v \in I'$ holds for *any* independent set I' of T such that $I \overset{T}{\leftrightsquigarrow} I'$. Conversely, if a token on a vertex $v \in I$ is not (T, I)-rigid, then it is (T, I)-*movable*; in other words, there exists an independent set I' such that $v \notin I'$ and $I \overset{T}{\leftrightsquigarrow} I'$. For example, in Figure 4, the tokens t_1, t_2, t_3, t_4 are (T, I)-rigid, while the tokens t_5, t_6, t_7 are (T, I)-movable. Note that, even though t_6 and t_7 cannot be slid to any neighbor in T under I, we can slide them after sliding t_5 downward.

Figure 5: (a) A (T, I)-rigid token on u, and (b) a (T, I)-movable token on u.

We then extend the concept of rigid/movable tokens to subgraphs of T. For any subgraph T' of T, we denote simply $I \cap T' = I \cap V(T')$. Then, a token on a vertex $v \in I \cap T'$ is $(T', I \cap T')$-*rigid* if $v \in J$ holds for *any* independent set J of T' such that $I \cap T' \overset{T'}{\leftrightsquigarrow} J$; otherwise it is $(T', I \cap T')$-*movable*. For example, in Figure 4, tokens t_6 and t_7 are $(T', I \cap T')$-rigid even though they are (T, I)-movable in the whole tree T. Note that, since the reconfiguration is restricted only to the subgraph T', we cannot use any vertex (and hence any edge) in $T \setminus T'$ during the reconfiguration. Furthermore, the vertex subset $J \cup \big(I \cap (T \setminus T')\big)$ does not necessarily form an independent set of the whole tree T.

We now give our first key lemma, which gives a characterization of rigid tokens. (See also Figure 5(a) for the claim (b) below.)

Lemma 1. *Let I be an independent set of a tree T, and let u be a vertex in I.*
(*a*) *Suppose that $|V(T)| = |\{u\}| = 1$. Then, the token on u is (T, I)-rigid.*
(*b*) *Suppose that $|V(T)| \geq 2$. Then, the token on u is (T, I)-rigid if and only if, for every neighbor $v \in N(T, u)$, there exists a vertex $w \in I \cap N(T_v^u, v)$ such that the token on w is $(T_w^v, I \cap T_w^v)$-rigid.*

Proof. Obviously, the claim (a) holds. In the following, we thus assume that $|V(T)| \geq 2$ and prove the claim (b).

We first show the if direction. Since we can slide a token only along an edge of T, if the token t on u is not (T, I)-rigid (and hence is (T, I)-movable), then it must be slid to some neighbor $v \in N(T, u)$. (See Figure 5(a).) However, by the assumption, there exists a vertex $w \in I \cap N(T_v^u, v)$ such that the token on w is $(T_w^v, I \cap T_w^v)$-rigid. We can thus conclude that t is (T, I)-rigid.

We then show the only-if direction by taking a contrapositive. Suppose that u has a neighbor $v \in N(T, u)$ such that either $I \cap N(T_v^u, v) = \emptyset$ or all tokens on $w \in I \cap N(T_v^u, v)$ are $(T_w^v, I \cap T_w^v)$-movable. (See Figure 5(b).) Then, we will prove that the token t on u is (T, I)-movable; in particular, we can slide t from u to v. Since any token t' on a vertex $w \in I \cap N(T_v^u, v)$ is $(T_w^v, I \cap T_w^v)$-movable, we can slide t' to some vertex in T_w^v via a reconfiguration sequence $\mathcal{S}_w$ in T_w^v. Recall that only the vertex v is adjacent with a vertex in T_w^v and $v \notin I$. Therefore, $\mathcal{S}_w$ can be naturally extended to a reconfiguration sequence $\mathcal{S}$ in the whole tree T such that $I' \cap \big(T \setminus T_w^v\big) = I \cap \big(T \setminus T_w^v\big)$ holds for any independent set $I' \in \mathcal{S}$ of T. Apply this process to all tokens on vertices in $I \cap N(T_v^u, v)$, and obtain an independent set I'' of T such that $I'' \cap N(T_v^u, v) = \emptyset$. Then, we can slide the token t on u to v. Thus, t is (T, I)-movable. $\square$

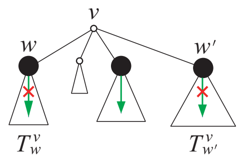


Figure 6: Illustration for Lemma 2.

The following lemma is useful for proving the correctness of our algorithm in Section 3.3.

Lemma 2. *Let I be an independent set of a tree T such that all tokens are (T, I)-movable, and let v be a vertex such that $v \notin I$. Then, there exists at most one neighbor $w \in I \cap N(T, v)$ such that the token on w is $(T_w^v, I \cap T_w^v)$-rigid.*

Proof. Suppose for a contradiction that there exist two neighbors w and w' in $I \cap N(T, v)$ such that the tokens on w and w' are $(T_w^v, I \cap T_w^v)$-rigid and $(T_{w'}^v, I \cap T_{w'}^v)$-rigid, respectively. (See Figure 6.) Since the token t on w is $(T_w^v, I \cap T_w^v)$-rigid but is (T, I)-movable, there is a reconfiguration sequence $\mathcal{S}_t$ starting from I which slides t to v. However, before sliding t to v, $\mathcal{S}_t$ must slide the token t' on w' to some vertex in $N(T_{w'}^v, w')$. This contradicts the assumption that t' is $(T_{w'}^v, I \cap T_{w'}^v)$-rigid. $\square$

3.2. Linear-time algorithm

In this subsection, we describe an algorithm to solve the sliding token problem for trees, and estimate its running time; the correctness of the algorithm will be proved in Section 3.3.

Let T be a tree with n vertices, and let I_b and I_r be two given independent sets of T. For an independent set I of T, we denote by $\mathsf{R}(I)$ the set of all vertices in I on which (T, I)-rigid tokens are placed. Then, the following algorithm determines whether $I_b \overset{T}{\leftrightsquigarrow} I_r$ or not.

Step 1. Compute $\mathsf{R}(I_b)$ and $\mathsf{R}(I_r)$. Return "no" if $\mathsf{R}(I_b) \neq \mathsf{R}(I_r)$; otherwise go to Step 2.

Step 2. Delete the vertices in $N[T, \mathsf{R}(I_b)] = N[T, \mathsf{R}(I_r)]$ from T, and obtain a forest F consisting of q trees $T_1, T_2, \ldots, T_q$. Return "yes" if $|I_b \cap T_j| = |I_r \cap T_j|$ holds for every $j \in \{1, 2, \ldots, q\}$; otherwise return "no."

We now show that our algorithm above runs in $O(n)$ time. Clearly, Step 2 can be done in $O(n)$ time, and hence we will show that Step 1 can be executed in $O(n)$ time.

We first give the following property of rigid tokens on a tree, which says that deleting movable tokens does not affect the rigidity of the other tokens.

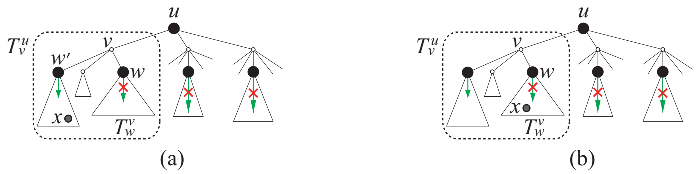


Figure 7: Illustration for Lemma 3.

Lemma 3. *Let I be an independent set of a tree T. Assume that the token on a vertex $x \in I$ is (T, I)-movable. Then, for every vertex $u \in I \setminus \{x\}$, the token on u is (T, I)-rigid if and only if it is $(T, I \setminus \{x\})$-rigid.*

Proof. The if direction is trivially true, because we cannot make a rigid token movable by adding another token. We thus show the only-if direction by contradiction.

Let $I' = I \setminus \{x\}$. Suppose that $u \in I$ is a closest vertex to x such that its token is (T, I)-rigid but (T, I')-movable. Let v be the neighbor of u such that the subtree T_v^u contains x. (See Figure 7.) Note that $x \neq v$ since $x, u \in I$ and v is a neighbor of u. Since the token t_u on u is (T, I)-rigid, by Lemma 1 the vertex $v \in N(T, u)$ has at least one neighbor $w \in I \cap N(T_v^u, v)$ such that the token t_w on w is $(T_w^v, I \cap T_w^v)$-rigid. Indeed, t_w is (T, I)-rigid, because t_u is assumed to be (T, I)-rigid. Thus, we know that $x \neq w$ since the token t_x on x is (T, I)-movable.

First, consider the case where x is contained in a subtree $T_{w'}^v$ for some neighbor w' of v other than w. (See Figure 7(a).) Then, $I' \cap T_w^v = I \cap T_w^v$. Since t_w is $(T_w^v, I \cap T_w^v)$-rigid, it is also $(T_w^v, I' \cap T_w^v)$-rigid. Therefore, by Lemma 1 the token t_u is (T, I')-rigid. This contradicts the assumption that t_u is (T, I')-movable.

We thus consider the case where $x \in V(T_w^v) \setminus \{w\}$. (See Figure 7(b).) Recall that I' is obtained by deleting only x from I. Then, since t_u is (T, I)-rigid but (T, I')-movable, there must exist a reconfiguration sequence such that the token t_u slides and its first slide is from u to v. However, before executing this token-slide, we have to slide t_w to some vertex in $N(T_w^v, w)$. Thus, t_w is $(T_w^v, I' \cap T_w^v)$-movable, and hence it is also (T, I')-movable. Since t_w is (T, I)-rigid and w is strictly closer to $x \in V(T_w^v)$ than u, this contradicts the assumption that u is a closest vertex to x such that its token is (T, I)-rigid but (T, I')-movable. $\square$

Then, the following lemma proves that Step 1 can be executed in $O(n)$ time.

Lemma 4. *For an independent set I of a tree T with n vertices, $\mathsf{R}(I)$ can be computed in $O(n)$ time.*

Proof. Lemma 3 implies that the set $\mathsf{R}(I)$ of all (T, I)-rigid tokens in I can be found by removing all (T, I)-movable tokens in I. Observe that, if I contains (T, I)-movable tokens, then at least one of them can be immediately slid to one of its neighbors. That is, there is a token on $u \in I$ which has a neighbor $w \in N(T, u)$ such that $N(T, w) \cap I = \{u\}$. Then, the following algorithm efficiently finds and removes such tokens iteratively.

Step A. Define and compute $\deg_I(w) = |N(T, w) \cap I|$ for all vertices $w \in V(T)$.

Step B. Define and compute $M = \{u \in I \mid \exists w \in N(T, u)$ such that $\deg_I(w) = 1\}$, that is, M is the set of tokens that can be immediately slid.

Step C. Repeat the following steps (i)–(iii) until $M = \emptyset$.

(i) Select an arbitrary vertex $u \in M$, and remove it from M and I.

(ii) Update $\deg_I(w) := \deg_I(w) - 1$ for each neighbor $w \in N(T, u)$.

(iii) If $\deg_I(w)$ becomes one by the update (ii) above, then add the vertex $u' \in N(T, w) \cap I$ into M.

Step D. Output I. Note that, since $M = \emptyset$, all tokens in I are now (T, I)-rigid.

Clearly, Steps A, B and D can be done in $O(n)$ time. We now show that Step C takes only $O(n)$ time. Each vertex in I can be selected at most once as u at Step C-(i). For the selected vertex u, Step C-(ii) takes $O(\deg_T(u))$ time for updating $\deg_I(w)$ of its neighbors $w \in N(T, u)$. Each vertex in $V(T) \setminus I$ can be selected at most once as w at Step C-(iii). For the selected vertex w, Step C-(iii) takes $O(\deg_T(w))$ time for finding $u' \in N(T, w) \cap I$. Therefore, Step C takes $O\Big(\sum_{v \in V(T)} \deg_T(v)\Big) = O(n)$ time in total. □

Therefore, Step 1 of our algorithm can be done in $O(n)$ time, and hence the algorithm runs in linear time in total.

3.3. Correctness of the algorithm

In this subsection, we prove that the $O(n)$-time algorithm in Section 3.2 correctly determines whether $I_b \overset{T}{\leftrightsquigarrow} I_r$ or not, for two given independent sets I_b and I_r of a tree T.

We first show the correctness of Step 1.

Observation 1. *Suppose that* $\mathsf{R}(I_b) \neq \mathsf{R}(I_r)$ *for two given independent sets* I_b *and* I_r *of a tree* T*. Then, it is a no-instance.*

Proof. By the definition of rigid tokens, $\mathsf{R}(I_b) = \mathsf{R}(I')$ holds for *any* independent set I' of T such that $I_b \overset{T}{\leftrightsquigarrow} I'$. Therefore, there is no reconfiguration sequence between I_b and I_r if $\mathsf{R}(I_r) \neq \mathsf{R}(I_b)$. □

We then show the correctness of Step 2. We first claim that deleting the vertices with rigid tokens together with their neighbors does not affect the reconfigurability.

Lemma 5. *Suppose that* $\mathsf{R}(I_b) = \mathsf{R}(I_r)$ *for two given independent sets* I_b *and* I_r *of a tree* T*, and let* F *be the forest obtained by deleting the vertices in* $N[T, \mathsf{R}(I_b)] = N[T, \mathsf{R}(I_r)]$ *from* T*. Then,* $I_b \overset{T}{\leftrightsquigarrow} I_r$ *if and only if* $I_b \cap F \overset{F}{\leftrightsquigarrow} I_r \cap F$*. Furthermore, all tokens in* $I_b \cap F$ *are* $(F, I_b \cap F)$*-movable, and all tokens in* $I_r \cap F$ *are* $(F, I_r \cap F)$*-movable.*

Proof. We first prove the if direction. Suppose that $I_b \cap F \overset{F}{\leftrightsquigarrow} I_r \cap F$, and hence there exists a reconfiguration sequence $\mathcal{S}_F$ between $I_b \cap F$ and $I_r \cap F$. Then, for each independent set $I \in \mathcal{S}_F$ of F, the vertex subset $\mathsf{R}(I_b) \cup I = \mathsf{R}(I_r) \cup I$ forms an independent set of T since F is obtained by deleting all vertices in $N[T, \mathsf{R}(I_b)] = N[T, \mathsf{R}(I_r)]$. Therefore, $\mathcal{S}_F$ can be extended to a reconfiguration sequence between I_b and I_r of T. We thus have $I_b \overset{T}{\leftrightsquigarrow} I_r$.

We then prove the only-if direction. Suppose that $I_b \overset{T}{\leftrightsquigarrow} I_r$, and hence there exists a reconfiguration sequence $\mathcal{S}_T$ between I_b and I_r. Then, for any independent set $I \in \mathcal{S}_T$, we have $I_b \overset{T}{\leftrightsquigarrow} I$ and $I \overset{T}{\leftrightsquigarrow} I_r$, and hence by the definition of rigid tokens $\mathsf{R}(I_b) = \mathsf{R}(I_r) \subseteq I$ holds. Furthermore, $I \setminus \mathsf{R}(I_b) = I \setminus \mathsf{R}(I_r)$ is a vertex subset of $V(F)$ since no token can be placed on any neighbor of $\mathsf{R}(I_b) = \mathsf{R}(I_r)$. Therefore, $I \setminus \mathsf{R}(I_b) = I \setminus \mathsf{R}(I_r)$ forms an independent set of F. For two consecutive independent sets I_{i-1} and I_i in $\mathcal{S}_T$, let $I_{i-1} \setminus I_i = \{u\}$ and $I_i \setminus I_{i-1} = \{v\}$. Since $u \notin I_i$ and $v \notin I_{i-1}$, neither u nor v are in $\mathsf{R}(I_b) = \mathsf{R}(I_r)$. Therefore, we have $u, v \in V(F)$, and hence the edge $\{u, v\}$ is in $E(F)$. Then, we can obtain a reconfiguration sequence between $I_b \cap F$ and $I_r \cap F$ by replacing all independent sets $I \in \mathcal{S}_T$ with $I \cap F$. We thus have $I_b \cap F \overset{F}{\leftrightsquigarrow} I_r \cap F$.

We finally prove that all tokens in $I_b \cap F$ are $(F, I_b \cap F)$-movable. (The proof for the tokens in $I_r \cap F$ is the same.) Notice that each token t on a vertex v in $I_b \cap F$ is (T, I_b)-movable; otherwise $t \in \mathsf{R}(I_b)$. Therefore, there exists an independent set I' of T such that $v \notin I'$ and $I_b \overset{T}{\leftrightsquigarrow} I'$. Then, $I_b \cap F \overset{F}{\leftrightsquigarrow} I' \cap F$ as we have proved above, and hence t is $(F, I_b \cap F)$-movable. □

Suppose that $\mathsf{R}(I_b) = \mathsf{R}(I_r)$ for two given independent sets I_b and I_r of a tree T. Let F be the forest consisting of q trees $T_1, T_2, \ldots, T_q$, which is obtained from T by deleting the vertices in $N[T, \mathsf{R}(I_b)] = N[T, \mathsf{R}(I_r)]$. Since we can slide a token only along an edge of F, we clearly have $I_b \cap F \overset{F}{\leftrightsquigarrow} I_r \cap F$ if and only if $I_b \cap T_j \overset{T_j}{\leftrightsquigarrow} I_r \cap T_j$ for all $j \in \{1, 2, \ldots, q\}$. Furthermore, Lemma 5 implies that, for each $j \in \{1, 2, \ldots, q\}$, all tokens in $I_b \cap T_j$ are $(T_j, I_b \cap T_j)$-movable; similarly, all tokens in $I_r \cap T_j$ are $(T_j, I_r \cap T_j)$-movable.

We now give our second key lemma, which completes the correctness proof of our algorithm.

Lemma 6. *Let* I_b *and* I_r *be two independent sets of a tree* T *such that all tokens in* I_b *and* I_r *are* (T, I_b)*-movable and* (T, I_r)*-movable, respectively. Then,* $I_b \overset{T}{\leftrightsquigarrow} I_r$ *if and only if* $|I_b| = |I_r|$*.*

The only-if direction of Lemma 6 is trivial, and hence we prove the if direction. In our proof, we do *not* reconfigure I_b into I_r directly, but reconfigure both

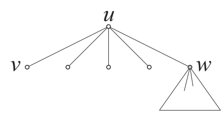


Figure 8: A degree-1 vertex v of a tree T which is safe.

I_b and I_r into some independent set I^* of T. Note that, since any reconfiguration sequence is reversible, $I_b \stackrel{T}{\leftrightsquigarrow} I^*$ and $I_r \stackrel{T}{\leftrightsquigarrow} I^*$ imply that $I_b \stackrel{T}{\leftrightsquigarrow} I_r$.

We say that a degree-1 vertex v of T is *safe* if its unique neighbor u has at most one neighbor w of degree more than one. (See Figure 8.) Note that any tree has at least one safe degree-1 vertex.

As the first step of the if direction proof, we give the following lemma.

Lemma 7. *Let I be an independent set of a tree T such that all tokens in I are (T, I)-movable, and let v be a safe degree-1 vertex of T. Then, there exists an independent set I' such that $v \in I'$ and $I \stackrel{T}{\leftrightsquigarrow} I'$.*

Proof. Suppose that $v \notin I$; otherwise the lemma clearly holds. We will show that one of the closest tokens from v can be slid to v. Let $M = \{w \in I \mid \mathsf{dist}(v, w) = \min_{x \in I} \mathsf{dist}(v, x)\}$. Let w be an arbitrary vertex in M, and let $(p_0 = v, p_1, \ldots, p_\ell = w)$ be the vw-path in T. (See Figure 9.) If $\ell = 1$ and hence $p_1 \in I$, then we can simply slide the token on p_1 to v. Thus, we may assume that $\ell \geq 2$.

We note that no token is placed on the vertices $p_0, \ldots, p_{\ell-1}$ and the neighbors of $p_0, \ldots, p_{\ell-2}$, because otherwise the token on w is not closest to v. Let $M' = M \cap N(T, p_{\ell-1})$. Since $p_{\ell-1} \notin I$, by Lemma 2 there exists at most one vertex $w' \in M'$ such that the token on w' is $(T^{p_{\ell-1}}_{w'}, I \cap T^{p_{\ell-1}}_{w'})$-rigid. We choose such a vertex w' if it exists, otherwise choose an arbitrary vertex in M' and regard it as w'.

Since all tokens on the vertices w'' in $M' \setminus \{w'\}$ are $(T^{p_{\ell-1}}_{w''}, I \cap T^{p_{\ell-1}}_{w''})$-movable, we first slide the tokens on w'' to some vertices in $T^{p_{\ell-1}}_{w''}$. Then, we can slide

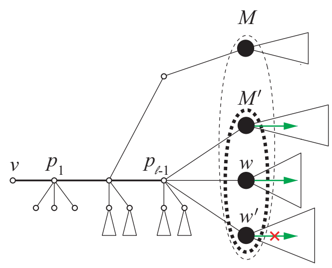
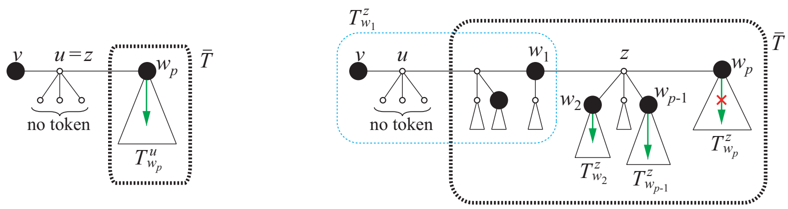


Figure 9: Illustration for Lemma 7.



(a) (b)

Figure 10: Illustration for Lemma 8.

the token on w' to v $(= p_0)$ along the path $(w', p_{\ell-1}, p_{\ell-2}, \ldots, p_0)$. In this way, we can obtain an independent set I' such that $v \in I'$ and $I \overset{T}{\leftrightsquigarrow} I'$. $\square$

We then prove that deleting a safe degree-1 vertex with a token together with its neighbor does not affect the movability of the other tokens. (See also Figure 10.)

Lemma 8. *Let v be a safe degree-1 vertex of a tree T, and let $\bar{T}$ be the subtree of T obtained by deleting v, its unique neighbor u, and the resulting isolated vertices. Let I be an independent set of T such that $v \in I$ and all tokens are (T, I)-movable. Then, all tokens in $I \setminus \{v\}$ are $(\bar{T}, I \setminus \{v\})$-movable.*

Proof. Since T^u_v consists of a single vertex v, the token on v is $(T^u_v, I \cap T^u_v)$-rigid. Therefore, no token is placed on degree-1 neighbors of u other than v (see Figure 10), because otherwise it contradicts to Lemma 2; recall that all tokens in I are assumed to be (T, I)-movable.

Let $\bar{I} = I \setminus \{v\}$. Suppose for a contradiction that there exists a token in $\bar{I}$ which is $(\bar{T}, \bar{I})$-rigid. Let $w_p \in \bar{I}$ be such a vertex closest to v, and let z be the vertex on the vw_p-path right before w_p.

Case (1): $z = u$. (See Figure 10(a).)

Recall that the token on v is (T, I)-movable, but is $(T^u_v, I \cap T^u_v)$-rigid. Therefore, by Lemma 2 the token on w_p must be $(T^u_{w_p}, I \cap T^u_{w_p})$-movable. However, this contradicts the assumption that w_p is $(\bar{T}, \bar{I})$-rigid, because $\bar{T} = T^u_{w_p}$ and $\bar{I} = I \cap T^u_{w_p}$ in this case.

Case (2): $z \neq u$. (See Figure 10(b).)

Let w_1 be the neighbor of z on the vw_p-path other than w_p; let $N(T, z) = \{w_1, w_2, \ldots, w_p\}$. We note that the subtree $T^z_{w_1}$ contains the deleted star $T \setminus \bar{T}$ centered at u.

We first note that the token t_p on w_p is $(\bar{T}^z_{w_p}, \bar{I} \cap \bar{T}^z_{w_p})$-rigid, because otherwise t_p can be slid to some vertex in $\bar{T}^z_{w_p}$ and hence it is $(\bar{T}, \bar{I})$-movable. Since $\bar{T}^z_{w_p} = T^z_{w_p}$ and $\bar{I} \cap \bar{T}^z_{w_p} = I \cap T^z_{w_p}$, the token t_p is also $(T^z_{w_p}, I \cap T^z_{w_p})$-rigid.

For each $j \in \{2, 3, \ldots, p-1\}$ with $w_j \in I$, since t_p is $(T^z_{w_p}, I \cap T^z_{w_p})$-rigid and all tokens in I are (T, I)-movable, by Lemma 2 each token t_j on w_j is

$(T^z_{w_j}, I \cap T^z_{w_j})$-movable. Then, since $T^z_{w_j} = \bar{T}^z_{w_j}$ and $I \cap T^z_{w_j} = \bar{I} \cap \bar{T}^z_{w_j}$, the token t_j is $(\bar{T}^z_{w_j}, \bar{I} \cap \bar{T}^z_{w_j})$-movable. Therefore, if $w_1 \notin \bar{I}$ or the token t_1 on w_1 is $(\bar{T}^z_{w_1}, \bar{I} \cap \bar{T}^z_{w_1})$-movable, then we can slide t_p from w_p to z after sliding each token t_j in $\bar{I} \cap \{w_1, w_2, \dots, w_{p-1}\}$ to some vertex of the subtree $\bar{T}^z_{w_j}$. This contradicts the assumption that t_p is $(\bar{T}, \bar{I})$-rigid.

Therefore, we have $w_1 \in \bar{I}$ and a token t_1 on w_1 is $(\bar{T}^z_{w_1}, \bar{I} \cap \bar{T}^z_{w_1})$-rigid. Then, t_1 is $(\bar{T}, \bar{I})$-rigid, because t_1 can be slid only to z which is adjacent with w_p having the $(\bar{T}^z_{w_p}, \bar{I} \cap \bar{T}^z_{w_p})$-rigid token t_p. Since w_1 is on the vw_p-path in T, this contradicts the assumption that t_p is the $(\bar{T}, \bar{I})$-rigid token closest to v. □

Proof of the if direction of Lemma 6

We now prove the if direction of the lemma by induction on the number of tokens $|I_b| = |I_r|$. The lemma clearly holds for any tree T if $|I_b| = |I_r| = 1$, because T has only one token and hence we can slide it along the unique path in T.

We choose an arbitrary safe degree-1 vertex v of a tree T, whose unique neighbor is u. Since all tokens in I_b are (T, I_b)-movable, by Lemma 7 we can obtain an independent set I'_b of T such that $v \in I'_b$ and $I_b \overset{T}{\leftrightsquigarrow} I'_b$. By Lemma 8 all tokens in $I'_b \setminus \{v\}$ are $(\bar{T}, I'_b \setminus \{v\})$-movable, where $\bar{T}$ is the subtree defined in Lemma 8. Similarly, we can obtain an independent set I'_r of T such that $v \in I'_r$, $I_r \overset{T}{\leftrightsquigarrow} I'_r$ and all tokens in $I'_r \setminus \{v\}$ are $(\bar{T}, I'_r \setminus \{v\})$-movable. Apply the induction hypothesis to the pair of independent sets $I'_b \setminus \{v\}$ and $I'_r \setminus \{v\}$ of $\bar{T}$. Then, we have $I'_b \setminus \{v\} \overset{\bar{T}}{\leftrightsquigarrow} I'_r \setminus \{v\}$. Recall that both $u \notin I'_b$ and $u \notin I'_r$ hold, and u is the unique neighbor of v in T. Furthermore, $u \notin V(\bar{T})$. Therefore, we can extend the reconfiguration sequence in $\bar{T}$ between $I'_b \setminus \{v\}$ and $I'_r \setminus \{v\}$ to a reconfiguration sequence in T between I'_b and I'_r. We thus have $I_b \overset{T}{\leftrightsquigarrow} I'_b \overset{T}{\leftrightsquigarrow} I'_r \overset{T}{\leftrightsquigarrow} I_r$.

This completes the proof of Lemma 6, and hence completes the proof of Theorem 1. □

3.4. Length of reconfiguration sequence

In this subsection, we show that an actual reconfiguration sequence can be found for a yes-instance on trees, by implementing our proofs in Section 3.3. Furthermore, the length of the obtained reconfiguration sequence is at most quadratic.

Theorem 2. *Let I_b and I_r be two independent sets of a tree T with n vertices. If $I_b \overset{T}{\leftrightsquigarrow} I_r$, then there exists a reconfiguration sequence of length $O(n^2)$ between I_b and I_r, and it can be output in $O(n^2)$ time.*

As we have mentioned in Introduction, recall that there exists an infinite family of instances on paths for which any reconfiguration sequence requires $\Omega(n^2)$ length, where n is the number of vertices.

We note that a reconfiguration sequence $\mathcal{S}$ can be represented by a sequence of edges on which tokens are slid. Therefore, the space for representing $\mathcal{S}$ can be bounded by a function linear in the length of $\mathcal{S}$.

By Theorem 1 we can determine whether $I_b \overset{T}{\leftrightsquigarrow} I_r$ or not in $O(n)$ time. In the following, we thus assume that $I_b \overset{T}{\leftrightsquigarrow} I_r$. Furthermore, suppose that all tokens in I_b are (T, I_b)-movable, and that all tokens in I_r are (T, I_r)-movable; otherwise we obtain the forest by deleting the vertices in $N[T, \mathsf{R}(I_b)] = N[T, \mathsf{R}(I_r)]$ from T, and find a reconfiguration sequence for each tree in the forest, according to Lemma 5.

As in the if-direction proof of Lemma 6, we choose an arbitrary safe degree-1 vertex v of T, and obtain an independent set I'_b of T such that $v \in I'_b$ and $I_b \overset{T}{\leftrightsquigarrow} I'_b$, as follows.

(a) Find a vertex $w \in I_b$ which is closest to v, and let $(v, p_1, p_2, \ldots, p_{\ell-1}, w)$ be the vw-path in T. Let $M' = I_b \cap N(T, p_{\ell-1})$. (See also Figure 9.)

(b) Choose a vertex w' such that the token on w' is $(T^{p_{\ell-1}}_{w'}, I \cap T^{p_{\ell-1}}_{w'})$-rigid if it exists, otherwise choose an arbitrary vertex in M' and regard it as w'.

(c) Slide each token on $w'' \in M' \setminus \{w'\}$ to some vertex in $T^{p_{\ell-1}}_{w''}$, and then slide the token on w' to v.

In Lemma 7 we have proved that such a reconfiguration sequence from I_b to I'_b always exists. We apply the same process to I_r for the same safe degree-1 vertex v, and obtain an independent set I'_r of T such that $I_r \overset{T}{\leftrightsquigarrow} I'_r$ and $v \in I'_b \cap I'_r$. Repeat these processes until we obtain the same independent set I^* of T such that $I_b \overset{T}{\leftrightsquigarrow} I^*$ and $I_r \overset{T}{\leftrightsquigarrow} I^*$. Note that, since any reconfiguration sequence is reversible, this means that we obtained a reconfiguration sequence between I_b and I_r.

Therefore, to prove Theorem 2, it suffices to show that the algorithm above runs in $O(n)$ time for one safe degree-1 vertex v and the reconfiguration sequence for sliding one token to v is of length $O(n)$. In particular, the following lemma completes the proof of Theorem 2.

Lemma 9. *Let I be an independent set of a tree T, and let $w \in I$. For a neighbor $z \in N(T, w)$, suppose that the token on w is $(T^z_w, I \cap T^z_w)$-movable. Then, there exists a reconfiguration sequence $\mathcal{S}_w$ of length at most $|V(T^z_w)|$ from I to an independent set I' of T such that $w \notin I'$ and $J \cap (T \setminus T^z_w) = I \cap (T \setminus T^z_w)$ for all $J \in \mathcal{S}_w$. Furthermore, $\mathcal{S}_w$ can be output in $O(|V(T^z_w)|)$ time.*

Proof. We prove the lemma by induction on the depth of T^z_w, where the depth of a tree is the longest distance from its root to a leaf. If the depth of T^z_w is zero (and hence T^z_w consists of a single vertex w), then the token on w is $(T^z_w, I \cap T^z_w)$-rigid; this contradicts the assumption. Therefore, we may assume that the depth is at least one. If the depth of T^z_w is exactly one, then T^z_w is a star centered at w, and no token is placed on any neighbor of w. Thus, we can slide the token on w by 1 $(< |V(T^z_w)|)$ token-slides. Then, the lemma holds for trees T^z_w with depth one.

Assume that the depth of T^z_w is $k \geq 2$, and that the lemma holds for trees with depth at most $k-1$. Since w is $(T^z_w, I \cap T^z_w)$-movable, by Lemma 1 there

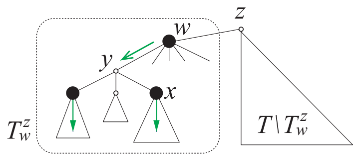


Figure 11: Illustration for Lemma 9.

is a vertex $y \in N(T_w^z, w)$ such that either $I \cap N(T_y^w, y) = \emptyset$ or all tokens on the vertices x in $I \cap N(T_y^w, y)$ are $(T_x^y, I \cap T_x^y)$-movable. (See Figure 11.) Then, we can obtain a reconfiguration sequence which (1) first slides all tokens on the vertices x in $I \cap N(T_y^w, y)$ to some vertices in T_x^y if $I \cap N(T_y^w, y) \neq \emptyset$, and (2) then slide the token on w to the vertex y. By applying the induction hypothesis to each subtree T_x^y, this reconfiguration sequence is of length at most

$$1 + \sum_{x \in I \cap N(T_y^w, y)} |V(T_x^y)| = \left|V(T_y^w)\right|,$$

and can be output in $O(\left|V(T_y^w)\right|)$ time. Note that $w \notin I'$ holds for the obtained independent set I' of T. Thus, the lemma holds for trees T_w^z with depth k. $\square$

We note that this lemma does not yield a reconfiguration sequence with the shortest length between I_b and I_r; such a reconfiguration sequence may not use any safe degree-1 vertex.

4. Concluding Remarks

In this paper, we have developed an $O(n)$-time algorithm to solve the SLIDING TOKEN problem for trees with n vertices, based on a simple but non-trivial characterization of rigid tokens. We have shown that there exists a reconfiguration sequence of length $O(n^2)$ for any yes-instance on trees, and it can be output in $O(n^2)$ time. Furthermore, there exists an infinite family of instances on paths for which any reconfiguration sequence requires $\Omega(n^2)$ length.

The complexity status of SLIDING TOKEN remains open for chordal graphs and interval graphs. Interestingly, these graphs have no-instances such that all tokens are movable. (See Figure 12 for example.)

Figure 12: No-instance for an interval graph such that all tokens are movable.

Acknowledgments

We thank anonymous referees of the preliminary version [9] and of this journal version for their helpful suggestions. This work is supported in part by NSF grant CCF-1161626 and DARPA/AFOSR grant FA9550-12-1-0423, and by MEXT/JSPS KAKENHI 24106004, 25104521, 25330003, 25730003, 26330009 and 15H00849.

References

[1] Bodlaender, H.L.: A partial k-arboretum of graphs with bounded treewidth. Theoretical Computer Science 209, pp. 1–45 (1998)

[2] Bonamy, M., Bousquet, N.: Reconfiguring independent sets in cographs. `arXiv:1406.1433` (2014)

[3] Bonamy, M., Johnson, M., Lignos, I., Patel, V., Paulusma, D.: Reconfiguration graphs for vertex colourings of chordal and chordal bipartite graphs. J. Combinatorial Optimization 27, pp. 132–143 (2014)

[4] Bonsma, P.: The complexity of rerouting shortest paths. Theoretical Computer Science 510, pp. 1–12 (2013)

[5] Bonsma, P.: Independent set reconfiguration in cographs. Proc. of WG 2014, LNCS 8747, pp. 105–116 (2014)

[6] Bonsma, P., Cereceda, L.: Finding paths between graph colourings: PSPACE-completeness and superpolynomial distances. Theoretical Computer Science 410, pp. 5215–5226 (2009)

[7] Bonsma, P., Kamiński, M., Wrochna, M.: Reconfiguring independent sets in claw-free graphs. Proc. of SWAT 2014, LNCS 8503, pp. 86–97 (2014)

[8] Cereceda, L., van den Heuvel, J., Johnson, M.: Finding paths between 3-colourings. J. Graph Theory 67, pp. 69–82 (2011)

[9] Demaine, E.D., Demaine, M.L., Fox-Epstein, E., Hoang, D.A., Ito, T., Ono, H., Otachi, Y., Uehara, R., Yamada, T.: Polynomial-time algorithm for sliding tokens on trees. Proc. of ISAAC 2014, LNCS 8889, pp. 389–400 (2014)

[10] Gopalan, P., Kolaitis, P.G., Maneva, E.N., Papadimitriou, C.H.: The connectivity of Boolean satisfiability: computational and structural dichotomies. SIAM J. Computing 38, pp. 2330–2355 (2009)

[11] Hearn, R.A., Demaine, E.D.: PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. Theoretical Computer Science 343, pp. 72–96 (2005)

[12] Hearn, R.A., Demaine, E.D.: Games, Puzzles, and Computation. A K Peters (2009)

[13] Ito, T., Demaine, E.D.: Approximability of the subset sum reconfiguration problem. J. Combinatorial Optimization 28, pp. 639–654 (2014)

[14] Ito, T., Demaine, E.D., Harvey, N.J.A., Papadimitriou, C.H., Sideri, M., Uehara, R., Uno, Y.: On the complexity of reconfiguration problems. Theoretical Computer Science 412, pp. 1054–1065 (2011)

[15] Ito, T., Kamiński, M., Demaine, E.D.: Reconfiguration of list edge-colorings in a graph. Discrete Applied Mathematics 160, pp. 2199–2207 (2012)

[16] Ito, T., Kamiński, M., Ono, H., Suzuki, A., Uehara, R., Yamanaka, K.: On the parameterized complexity for token jumping on graphs. Proc. of TAMC 2014, LNCS 8402, pp. 341–351 (2014)

[17] Ito, T., Kawamura, K., Ono, H., Zhou, X.: Reconfiguration of list $L(2,1)$-labelings in a graph. Theoretical Computer Science 544, pp. 84–97 (2014)

[18] Ito, T., Kawamura, K., Zhou, X.: An improved sufficient condition for reconfiguration of list edge-colorings in a tree. IEICE Trans. on Information and Systems E95-D, pp. 737–745 (2012)

[19] Kamiński, M., Medvedev, P., Milanič, M.: Shortest paths between shortest paths. Theoretical Computer Science 412, pp. 5205–5210 (2011)

[20] Kamiński, M., Medvedev, P., Milanič, M.: Complexity of independent set reconfigurability problems. Theoretical Computer Science 439, pp. 9–15 (2012)

[21] Makino, K., Tamaki, S., Yamamoto, M.: An exact algorithm for the Boolean connectivity problem for k-CNF. Theoretical Computer Science 412, pp. 4613–4618 (2011)

[22] Mouawad, A.E., Nishimura, N., Raman, V., Simjour, N., Suzuki, A.: On the parameterized complexity of reconfiguration problems. Proc. of IPEC 2013, LNCS 8246, pp. 281–294 (2013)

[23] Mouawad, A.E., Nishimura, N., Raman, V., Wrochna, M.: Reconfiguration over tree decompositions. Proc. of IPEC 2014, LNCS 8894, pp. 246–257 (2014)

[24] van den Heuvel, J.: The complexity of change. Surveys in Combinatorics 2013, London Mathematical Society Lecture Notes Series 409 (2013).

[25] Wrochna, M.: Reconfiguration in bounded bandwidth and treedepth. `arXiv:1405.0847` (2014)