Polynomial-Time Algorithm for Sliding Tokens on Trees

Erik D. Demaine[1], Martin L. Demaine[1], Eli Fox-Epstein[2], Duc A. Hoang[3], Takehiro Ito[4], Hirotaka Ono[5], Yota Otachi[3], Ryuhei Uehara[3], and Takeshi Yamada[3]

[1] MIT Computer Science and Artificial Intelligence Laboratory, USA.
{edemaine, mdemaine}@mit.edu
[2] Department of Computer Science, Brown University, USA.
ef@cs.brown.edu
[3] School of Information Science, JAIST, Japan.
{hoanganhduc, otachi, uehara, tyama}@jaist.ac.jp
[4] Graduate School of Information Sciences, Tohoku University, Japan.
takehiro@ecei.tohoku.ac.jp
[5] Faculty of Economics, Kyushu University, Japan.
hirotaka@econ.kyushu-u.ac.jp

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 show that the problem is solvable for trees in quadratic time. Our proof is constructive: for a yes-instance, we can find an actual sequence of independent sets between I_b and I_r whose length (i.e., the number of token-slides) is quadratic. We note that there exists an infinite family of instances on paths for which any sequence requires quadratic length.

1 Introduction

Recently, *reconfiguration problems* attract 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 abides by 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 extensively for several well-known problems, including INDEPENDENT SET [4, 6, 9–12, 15, 17, 18, 20], SATISFIABILITY [8, 16], SET COVER, CLIQUE, MATCHING [11], VERTEX-COLORING [2, 5, 20], LIST $L(2,1)$-LABELING [13], SHORTEST PATH [3, 14], and so on. (See also a recent survey [19].)

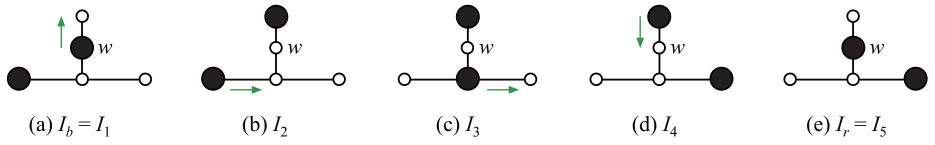

Fig. 1. A sequence $\langle I_1, I_2, \ldots, I_5\rangle$ of independent sets of the same graph, where the vertices in independent sets are depicted by large black circles (tokens).

1.1 Sliding token

The sliding token problem was introduced by Hearn and Demaine [9] 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 powerful tool, called the nondeterministic constraint logic model, which can be used to prove PSPACE-hardness of many puzzles and games [9], [10, 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 a very important role since several PSPACE-hardness results have been proved using reductions from it. Indeed, sliding token is one of the most well-studied reconfiguration problems.

In addition, reconfiguration problems for independent set (ISReconf, for short) have been studied under different reconfiguration rules, as follows.

- *Token Sliding* (TS rule) [5, 6, 9, 10, 15, 20]: This rule corresponds to the sliding token problem, that is, we can slide a single token only along an edge of a graph.
- *Token Jumping* (TJ rule) [6, 12, 15, 20]: A single token can "jump" to any vertex (including non-adjacent one) if it results in an independent set.
- *Token Addition and Removal* (TAR rule) [4, 11, 15, 17, 18, 20]: 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.

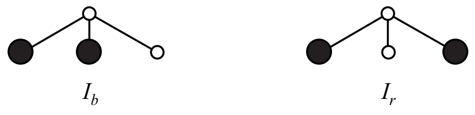

Fig. 2. A yes-instance for ISRECONF under the TJ rule, which is a no-instance for the SLIDING TOKEN problem.

We note that the existence of a desired sequence depends deeply on the reconfiguration rules. (See Fig. 2 for example.) However, ISRECONF is PSPACE-complete under any of the three reconfiguration rules for planar graphs [5, 9, 10], for perfect graphs [15], and for bounded bandwidth graphs [20]. 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 Fig. 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.

Results for TS rule (SLIDING TOKEN).

Kamiński et al. [15] 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 detour.

Very recently, Bonsma et al. [6] 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.

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 [15], 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 Fig. 1.) In addition, there are no-instances for trees under TS rule. (See Fig. 2.) These make the problem much more complicated, and we think they are the main reasons why SLIDING TOKEN for trees was open, despite the recent intensive algorithmic research on ISRECONF [4, 6, 12, 15, 18].

1.3 Our contribution

In this paper, we show that the SLIDING TOKEN problem can be solved in time $O(n^2)$ for any tree T with n vertices. Therefore, we can conclude that ISRECONF for trees is in P under any of the three reconfiguration rules.

We give a constructive proof: for a yes-instance, we can find an actual reconfiguration sequence between two given independent sets whose length is $O(n^2)$. We note that there exists an infinite family of instances on paths for which any reconfiguration sequence requires $\Omega(n^2)$ length.

We note that, since the treewidth of any graph G can be bounded by the bandwidth of G, the result of [20] 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 remarkable that any yes-instance on a tree, whose treewidth is one, has an $O(n^2)$-length reconfiguration sequence even though trees certainly require to make detours to transform.

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, in Fig. 2, every token in the two independent sets is 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 time $O(n^2)$. Note that, if I_b and I_r have different placements of rigid tokens, then it is a no-instance (Lemma 4).

(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.

Due to the page limitation, we omit some proofs from this extended abstract.

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 $N(G, v) = \{w \in V(G) \mid \{v, w\} \in E(G)\}$. 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 subgraph G' of a graph G, we denote by $G \setminus G'$ the subgraph of G induced by the vertices in $V(G) \setminus V(G')$.

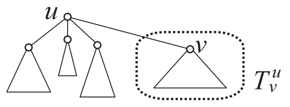

Fig. 3. Subtree T_v^u in the whole tree T.

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 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 Fig. 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 to say 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 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. 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 Fig. 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. *For a tree T with n vertices, the* SLIDING TOKEN *problem can be solved in $O(n^2)$ time.*

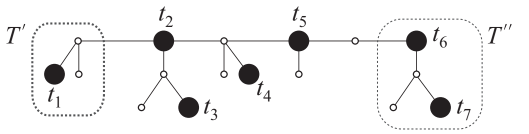

Fig. 4. An independent set I of a tree T, where t_1, t_2, t_3, t_4 are (T, I)-rigid tokens and t_5, t_6, t_7 are (T, I)-movable tokens. Token t_1 is $(T', I \cap T')$-movable for the subtree T', and tokens t_6 and t_7 are $(T'', I \cap T'')$-rigid for the subtree T''.

As a proof of Theorem 1, we give an $O(n^2)$-time algorithm which simply solves SLIDING TOKEN for a tree with n vertices. In Section 3.3 we will show that an actual reconfiguration sequence can be obtained for a yes-instance on trees, and we will estimate its length.

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 Fig. 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 I, we can slide them after sliding t_5 downward.

We then extend the concept of rigid/movable tokens to subtrees of T. For any subtree 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 the token is $(T', I \cap T')$-*movable*. For example, in Fig. 4, the token t_1 is $(T', I \cap T')$-movable even though it is (T, I)-rigid in the whole tree T, while tokens t_6 and t_7 are $(T'', I \cap T'')$-rigid even though they are (T, I)-movable in T. Note that, since independent sets are restricted only to the subtree 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 Fig. 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, a token on u is (T, I)-rigid if and only if, for all neighbors $v \in N(T, u)$, there exists a vertex $w \in I \cap N(T^u_v, v)$ such that the token on w is $(T^v_w, I \cap T^v_w)$-rigid.*

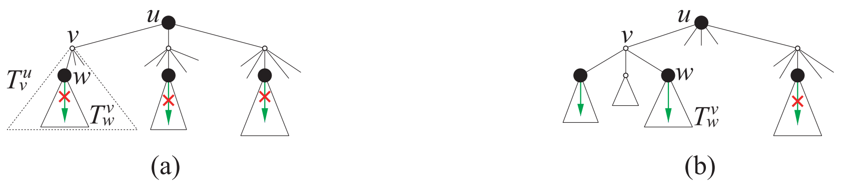

Fig. 5. (a) A (T, I)-rigid token on u, and (b) a (T, I)-movable token on u.

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-part. Suppose that, for all neighbors $v \in N(T, u)$, 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. (See Fig. 5(a).) Then, we will prove that the token t on u is (T, I)-rigid. Since we can slide a token only along an edge of T, if t is not (T, I)-rigid (and hence is (T, I)-movable), then it must be slid to some neighbor $v \in N(T, u)$. By the assumption, v is adjacent with another token t' placed on $w \in I \cap N(T_v^u, v)$, and hence we first have to slide t' to one of its neighbors other than v. However, this is impossible since the token t' on w is assumed to be $(T_w^v, I \cap T_w^v)$-rigid and hence $w \in J$ holds for any independent set J of T_w^v such that $I \cap T_w^v \overset{T_w^v}{\rightsquigarrow} J$. We can thus conclude that t is (T, I)-rigid.

We then show the only-if-part 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 Fig. 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 $I' \cap \left(T \setminus T_w^v\right) = I \cap \left(T \setminus T_w^v\right)$ 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. □

Lemma 1 implies that we can check whether one token in an independent set I of a tree T is (T, I)-rigid or not in linear time.

Lemma 2. *Given a tree T with n vertices, an independent set I of T, and a vertex $u \in I$, it can be decided in $O(n)$ time whether the token on u is (T, I)-rigid.*

The following lemma is useful for our algorithm in Section 3.2.

Lemma 3. *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 $(T_{w'}^v, I \cap$

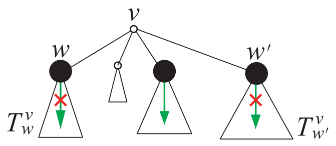

Fig. 6. Illustration for Lemma 3.

$T^v_{w'}$)-rigid, respectively. (See Fig. 6.) Since the token t on w is $(T^v_w, I \cap T^v_w)$-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 $N(T^v_{w'}, w')$. This contradicts the assumption that t' is $(T^v_{w'}, I \cap T^v_{w'})$-rigid. $\square$

3.2 Algorithm

In this subsection, we describe a quadratic-time algorithm to solve the SLIDING TOKEN problem on trees, and prove its correctness.

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.

Step 1. Compute $\mathsf{R}(I_b)$ and $\mathsf{R}(I_r)$ using Lemma 2. If $\mathsf{R}(I_b) \neq \mathsf{R}(I_r)$, then return "no"; 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 $T_1, T_2, \ldots, T_q$. If $|I_b \cap T_j| = |I_r \cap T_j|$ holds for every $j \in \{1, 2, \ldots, q\}$, then return "yes"; otherwise return "no."

By Lemma 2 we can determine whether one token in an independent set I of T is (T, I)-rigid or not in $O(n)$ time, and hence Step 1 can be done in time $O(n) \times (|I_b| + |I_r|) = O(n^2)$. Clearly, Step 2 can be done in $O(n)$ time. Therefore, our algorithm above runs in $O(n^2)$ time in total. In the remainder of this subsection, we thus prove the correctness of our algorithm.

We first show the correctness of Step 1.

Lemma 4. *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.*

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.*

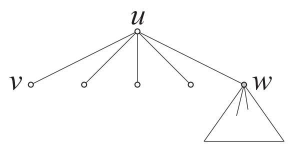

Fig. 7. A degree-1 vertex v of a tree T which is safe.

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-part is trivial, and hence we prove the if-part. In our proof, we do *not* reconfigure I_b into I_r directly, but reconfigure both I_b and I_r into some independent set I^* of T.

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 Fig. 7.) Note that any tree has at least one safe degree-1 vertex.

As the first step of the if-part 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 $I \overset{T}{\leftrightsquigarrow} I'$.*

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 $P = (p_0 = v, p_1, \ldots, p_\ell = w)$ be the vw-path in T. (See Fig. 8.) 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 3 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'.

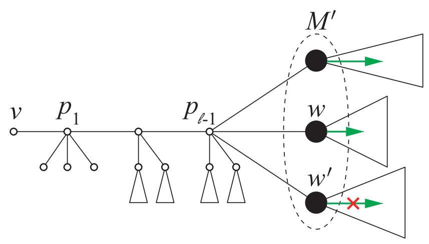

Fig. 8. Illustration for Lemma 7.

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 the token on w' to v along the path P. In this way, we can obtain an independent set I' such that $v \in I'$ and $I \overset{T}{\leftrightsquigarrow} I'$. □

We then prove that deleting a safe degree-1 vertex with a token does not affect the movability of the other tokens.

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.*

In Lemma 8, note that the token on v is $(T^u_v, I \cap T^u_v)$-rigid since T^u_v consists of a single vertex v. Therefore, no token is placed on degree-1 neighbors of u other than v, because otherwise it contradicts to Lemma 3; recall that all tokens in I are assumed to be (T, I)-movable.

Proof of the if-part of Lemma 6.

We prove the if-part of the lemma by the 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. 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 $I_b \overset{T}{\leftrightsquigarrow} I_r$.

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

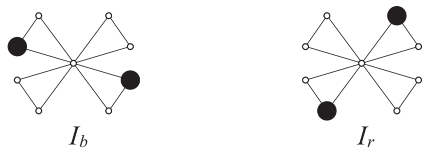

Fig. 9. No-instance for an interval graph such that all tokens are movable.

3.3 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 3.2. 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 found in $O(n^2)$ time.*

It is interesting 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. For example, consider a path $(v_1, v_2, \dots, v_{8k})$ with $n = 8k$ vertices for any positive integer k, and let $I_b = \{v_1, v_3, v_5, \dots, v_{2k-1}\}$ and $I_r = \{v_{6k+2}, v_{6k+4}, \dots, v_{8k}\}$. 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.

4 Concluding Remarks

In this paper, we have developed an $O(n^2)$-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 found in $O(n^2)$ time; while there exists an infinite family of instances on paths for which any reconfiguration sequence requires $\Omega(n^2)$ length.

Recently, we have improved the running time of our algorithm [7]: we proposed a linear-time algorithm which simply decides whether $I_b \overset{T}{\leftrightsquigarrow} I_r$ or not, for two given independent sets I_b and I_r of a tree T.

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 Fig. 9 for example.)

Acknowledgments

The authors thank anonymous referees for their helpful suggestions. This work is supported in part by NSF grant CCF-1161626 and DARPA/AFOSR grant FA9550-12-1-0423 (E.D. Demaine), and by JSPS KAKENHI 25106504 and 25330003 (T. Ito), 25104521, 26540005 and 26540005 (H. Ono) and 26330009 (R. Uehara).

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., 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)
3. Bonsma, P.: The complexity of rerouting shortest paths. Theoretical Computer Science 510, pp. 1–12 (2013)
4. Bonsma, P.: Independent set reconfiguration in cographs. To appear in WG 2014, also available at `arXiv:1402.1587` (2014)
5. Bonsma, P., Cereceda, L.: Finding paths between graph colourings: PSPACE-completeness and superpolynomial distances. Theoretical Computer Science 410, pp. 5215–5226 (2009)
6. Bonsma, P., Kamiński, M., Wrochna, M.: Reconfiguring independent sets in claw-free graphs. Proc. of SWAT 2014, LNCS 8503, pp. 86–97 (2014)
7. Demaine, E.D., Demaine, M.L., Fox-Epstein, E., Hoang, D.A., Ito, T., Ono, H., Otachi, Y., Uehara, R., Yamada, T.: Linear-time algorithm for sliding tokens on trees. `arXiv:1406.6576` (2014)
8. 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)
9. 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)
10. Hearn, R.A., Demaine, E.D.: Games, Puzzles, and Computation. A K Peters (2009)
11. 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)
12. 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)
13. 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)
14. Kamiński, M., Medvedev, P., Milanič, M.: Shortest paths between shortest paths. Theoretical Computer Science 412, pp. 5205–5210 (2011)
15. Kamiński, M., Medvedev, P., Milanič, M.: Complexity of independent set reconfigurability problems. Theoretical Computer Science 439, pp. 9–15 (2012)
16. 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)
17. 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)
18. Mouawad, A.E., Nishimura, N., Raman, V., Wrochna, M.: Reconfiguration over tree decompositions. `arXiv:1405.2447`
19. van den Heuvel, J.: The complexity of change. Surveys in Combinatorics 2013, London Mathematical Society Lecture Notes Series 409 (2013).
20. Wrochna, M.: Reconfiguration in bounded bandwidth and treedepth. `arXiv:1405.0847` (2014)