Sliding Tokens on Block Graphs

Duc A. Hoang[1], Eli Fox-Epstein[2], and Ryuhei Uehara[1]

[1] JAIST, Japan {hoanganhduc, uehara}@jaist.ac.jp
[2] Brown University, USA ef@cs.brown.edu

Abstract. Let I, J be two given independent sets of a graph G. Imagine that the vertices of an independent set are viewed as tokens (coins). A token is allowed to move (or slide) from one vertex to one of its neighbors. The Sliding Token problem asks whether there exists a sequence of independent sets of G starting from I and ending with J such that each intermediate member of the sequence is obtained from the previous one by moving a token according to the allowed rule. In this paper, we claim that this problem is solvable in polynomial time when the input graph is a block graph—a graph whose blocks are cliques. Our algorithm is developed based on the characterization of a non-trivial structure that, in certain conditions, can be used to indicate a no-instance of the problem. Without such a structure, a sequence of token slidings between any two independent sets of the same cardinality exists.

1 Introduction

Recently, motivated by the purpose of understanding the solution space of a problem, many theoretical computer scientists have focused on the study of *reconfiguration problems*. *Reconfiguration problems* are the set of problems in which we are given a collection of *feasible solutions*, together with some *reconfiguration rule(s)* that defines an *adjacency relation* on the set of feasible solutions of the original problem. The question is, using a reconfiguration rule, whether there is a step-by-step transformation which transforms one feasible solution to another, such that each intermediate result is also feasible. A simple example is the famous Rubik's cube puzzle. The reconfigurability of several well-known problems, including satisfiability, independent set, vertex-colouring, matching, clique, etc. have been studied extensively. For more information about this research area, see the survey [1].

As the independent set problem is one of the most important problems in the computational complexity theory, its reconfiguration variants have been well-studied [2,3,4]. Recall that an *independent set* of a graph is a set of pairwise non-adjacent vertices. Among these variants, the Sliding Token problem (first introduced by Hearn and Demaine [2]) is of particular interest (see [4] for the other variants). Given two independent sets I and J of a graph G, and imagine that a token is placed on each vertex in I. Then, the Sliding Token problem asks whether there exists a sequence (called a TS-*sequence*) $\mathcal{S} = \langle I_1, I_2, \ldots, I_\ell \rangle$ of independent sets of G such that

(a) $I_1 = I$, $I_\ell = J$, and $|I_i| = |I| = |J|$ for all i, $1 \leq i \leq \ell$; and
(b) for each i, $1 \leq i \leq \ell - 1$, there is an edge uv in G such that $I_i \setminus I_{i+1} = \{u\}$ and $I_{i+1} \setminus I_i = \{v\}$.

If such a sequence $\mathcal{S}$ exists, we say that $\mathcal{S}$ *reconfigures* I to J in G and write $I \overset{G}{\leftrightsquigarrow} J$. An example of a TS-sequence is given in Fig. 1. Observe that " $\overset{G}{\leftrightsquigarrow}$ " is indeed an equivalence relation. Sliding Token is PSPACE-complete even for planar graphs [2] and bounded-treewidth graphs [5]. On the positive side, polynomial-time algorithms have been designed recently for claw-free graphs [6], cographs [4], trees [7], bipartite permutation graphs [8], and cactus graphs [9].

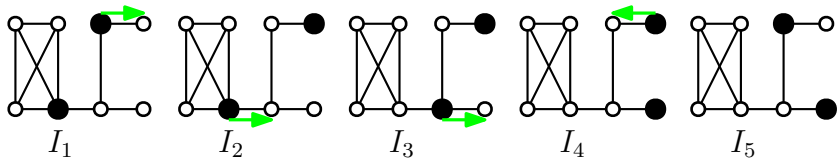


Fig. 1. Example of a TS-sequence $\langle I_1, I_2, \ldots, I_5 \rangle$ in a given graph that reconfigures I_1 to I_5. The vertices in independent sets are depicted by black circles (tokens).

A *block* of a graph G is a maximal connected subgraph with no cut vertex. A *block graph* is a graph whose blocks are cliques (for example, see the graph in Fig. 1). Note that, in order to preserve the independence property of the set of tokens, a token sometimes needs to make "detours". This restriction indeed makes Sliding Token more complicated (recall that the problem is PSPACE-complete even for bounded-treewidth graphs), even when the input graph is a tree (see [7]). As there might be exponential number of paths between any two vertices of a block graph (while in a tree, there is a unique path), for each token, we may have exponentially many choices of "routes" to slide and possibly super polynomial detours in general. Thus, in this case, the problem becomes more difficult. In this paper, we design a polynomial-time algorithm for solving the Sliding Token problem for block graphs.

Our algorithm is designed based on the following observations. Given a block graph G and an independent set I of G, one can characterize the properties of a non-trivial structure, called (G, I)-*confined clique* (Section 4). More precisely, we claim that one can find all (G, I)-confined cliques in polynomial time (Lemma 3), and, in certain conditions, we can easily derive if an instance of Sliding Token is a no-instance (Lemma 5). Without such a structure, we claim that for any pair of independent sets I, J, I is reconfigurable to J (and vice versa) if and only if they are of the same cardinality (Lemma 9).

Due to the limitation of space, some proofs are omitted.

2 Preliminaries

Graph notation

We define some notation that is commonly used in graph theory. For the notation that is not mentioned here, see [10]. Let G be a given graph, with edge set $E(G)$ and vertex set $V(G)$.
We sometimes denote by $|G|$ the size of $V(G)$. For a vertex v, we define $N_G(v) = \{w \in V(G) : vw \in E(G)\}$, $N_G[v] = N_G(v) \cup \{v\}$ and $\deg_G(v) = |N_G(v)|$. For two vertices u, v, we denote by $\mathsf{dist}_G(u, v)$ the *distance* between u and v in G. For a graph G, sometimes we write $I \cap G$ and $I - G$ to indicate the sets $I \cap V(G)$ and $I \setminus V(G)$, respectively.
For $X \subseteq V(G)$, we denote by $G[X]$ the subgraph of G *induced* by vertices of X. We write $G - X$ to indicate the graph $G[V(G) \setminus X]$. Similarly, for an induced subgraph H of G, $G - H$ indicates the graph $G[V(G) \setminus V(H)]$, and we say that the graph $G - H$ is obtained by *removing H from G*.

Notation for Sliding Token

We now define some useful notation for tackling Sliding Token. For a TS-sequence $\mathcal{S}$, we write $I \in \mathcal{S}$ if an independent set I of G appears in $\mathcal{S}$. For a vertex v, if there exists $I \in \mathcal{S}$ such that $v \in I$, then we say that $\mathcal{S}$ *involves* v. We say that $\mathcal{S} = \langle I_1, I_2, \ldots, I_\ell \rangle$ *slides* (or *moves*) the token t placed at $u \in I_1$ to $v \notin I_1$ in G if after applying the sliding steps described in $\mathcal{S}$, the token t is placed at $v \in I_\ell$. For convenience, we sometimes identify the token placed at a vertex with the vertex itself, and simply say "a token in an independent set I."
Let $W \subseteq V(G)$ and assume that $I \cap W \neq \emptyset$. We say that a token t placed at some vertex $u \in I \cap W$ is (G, I, W)-*confined* if for every J such that $I \overset{G}{\leftrightsquigarrow} J$, t is always placed at some vertex of W. In other words, t can only be slid along edges of $G[W]$. In case $W = \{u\}$, t is said to be (G, I)-*rigid*. The token t is (G, I)-*movable* if it is not (G, I)-rigid.
Let H be an induced subgraph of G. H is called (G, I)-*confined* if $I \cap H$ is a maximum independent set of H and all tokens in $I \cap H$ are $(G, I, V(H))$-confined. In particular, if H is a clique of G, we say that it is a (G, I)-*confined clique*. Note that if H is a clique then $|I \cap H| \leq 1$. We denote by $\mathsf{K}(G, I)$ the set of all (G, I)-confined cliques of G. For a vertex $v \in V(H)$, we define G_H^v to be the (connected) component of G_H containing v, where G_H is obtained from G by removing all edges of H.

3 Some useful observations

In this section, we present several useful observations. These observations will be implicitly used in many statements of this paper. The next proposition characterizes some properties of a (G, I)-confined induced subgraph.

Proposition 1 ([9, Lemma 1]). *Let I be an independent set of a graph G. Let H be an induced subgraph of G. Then the following conditions are equivalent.*

(i) H *is* (G, I)*-confined.*

(ii) *For every independent set* J *satisfying* $I \overset{G}{\rightsquigarrow} J$, $J \cap H$ *is a maximum independent set of* H.

(iii) $I \cap H$ *is a maximum independent set of* H *and for every* J *satisfying* $I \overset{G}{\rightsquigarrow} J$, *any token* t_x *placed at* $x \in J \cap H$ *is* $(G_H^x, J \cap G_H^x)$*-rigid.*

The next proposition says that when G is disconnected, one can deal with each component separately. In other words, when dealing with Sliding Token, it suffices to consider only connected graphs.

Proposition 2 ([9, Proposition 2]). *Let* I, J *be two given independent set of* G. *Assume that* $G_1, \ldots, G_k$ *are the components of* G. *Then* $I \overset{G}{\rightsquigarrow} J$ *if and only if* $I \cap G_i \overset{G_i}{\rightsquigarrow} J \cap G_i$ *for* $i = 1, 2, \ldots, k$.

In the next proposition, we claim that in certain conditions, a TS-sequence in a subgraph of G can be somehow "extended" to a sequence in G, and vice versa.

Proposition 3 ([9, Proposition 3]). *Let* u *be a vertex of a graph* G. *Let* $\mathcal{S} = \langle I_1, I_2, \ldots, I_\ell \rangle$ *be a* TS*-sequence in* G *such that for any* $I \in \mathcal{S}$, $u \in I$. *Let* $G'' = G - N_G[u]$. *Then* $I_1 \cap G' \overset{G''}{\rightsquigarrow} I_\ell \cap G'$. *Moreover, for any* TS*-sequence* $\mathcal{S}' = \langle I'_1, \ldots, I'_l \rangle$ *in* G'', $I'_1 \cup \{u\} \overset{G}{\rightsquigarrow} I'_l \cup \{u\}$.

In case G is a block graph, we also have:

Proposition 4. *Let* I *be an independent set of a block graph* G. *Let* B *be a block of* G *and suppose that* $I \cap B = \{u\}$. *Let* $\mathcal{S} = \langle I_1, I_2, \ldots, I_\ell \rangle$ *be a* TS*-sequence in* G *such that for any* $J \in \mathcal{S}$, $u \in J$. *Let* $G' = G - B$. *Then* $I_1 \cap G' \overset{G'}{\rightsquigarrow} I_\ell \cap G'$. *Moreover, for any* TS*-sequence* $\mathcal{S}' = \langle I'_1, \ldots, I'_l \rangle$ *in* G' *such that* $N_G(u) \cap I'_i = \emptyset$, *where* $i \in \{1, 2, \ldots, \ell\}$, $I'_1 \cup \{u\} \overset{G}{\rightsquigarrow} I'_l \cup \{u\}$.

Proposition 5. *Let* G *be a block graph and let* I *be an independent set of* G. *Let* $v \in V(G)$ *be such that no token in* $N_G(v) \cap I$ *is* $(G, I, N_G[v])$*-confined. Then there exists an independent set* J *of* G *such that* $I \overset{G}{\rightsquigarrow} J$ *and* $N_G[v] \cap J = \emptyset$.

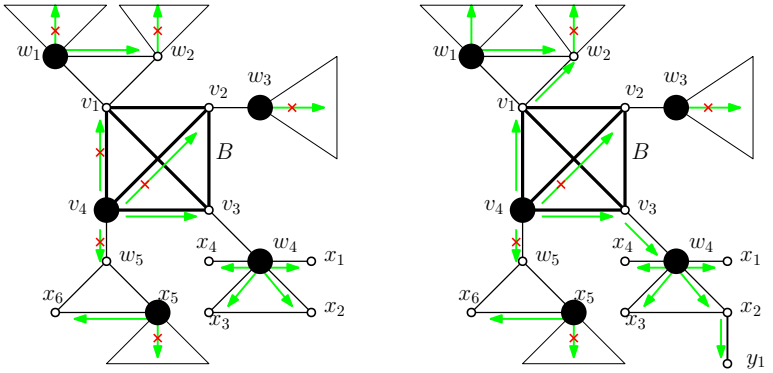
Proposition 6. *Let* I *be an independent set of a block graph* G. *Let* $w \in V(G)$. *Assume that no block of* G *containing* w *is* (G, I)*-confined. If there exists some vertex* $x \in N_G[w] \cap I$ *such that the token* t_x *placed at* x *is* $(G, I, N_G[w])$*-confined, then* x *is unique. Consequently, there must be some independent set* J *such that* $I \overset{G}{\rightsquigarrow} J$ *and* $N_G[w] \cap J = \{x\}$. *Moreover, let* H *be the graph obtained from* G *by turning* $N_G[w]$ *into a clique, called* B_w. *Then* t_x *is* $(G, J, N_G[w])$*-confined if and only if* B_w *is* (H, J)*-confined.*

4 Confined cliques in block graphs

In this section, we show that one can compute $\mathsf{K}(G, I)$ in polynomial time, where G is a block graph and I is an independent set of G. First, we prove an useful characterization of (G, I)-confined cliques.

Lemma 1. *Let I be an independent set of a block graph G. Let B be a block of G with $I \cap B \neq \emptyset$. Let $G' = G - B$. Then B is (G, I)-confined (see Fig. 2(a)) if and only if either $G = B$ or for every cut vertex $v \in V(B)$, one of the following conditions holds.*

- *(i) There exists a block $B' \neq B$ of G containing v such that $B' - v$ is $(G', I \cap G')$-confined (for example, the vertices v_1 and v_2 in Fig. 2(a)).*
- *(ii) For every block $B' \neq B$ of G containing v, $B' - v$ is not $(G', I \cap G')$-confined; and for every $w \in N_G(v) \setminus V(B)$, either*
  - *(ii-1) there exists a block B'' of G' containing w such that B'' is $(G', I \cap G')$-confined (for example, the vertex v_4 in Fig. 2(a)); or*
  - *(ii-2) every block B'' of G' containing w is not $(G', I \cap G')$-confined; and there exists $x \in N_{G'}[w] \cap I$ such that the token t_x placed at x is $(G', I \cap G', N_{G'}[w])$-confined (for example, the vertex v_3 in Fig. 2(a)).*



(a) (b)

Fig. 2. (a) B is (G, I)-confined and (b) B is not (G, I)-confined.

Next, we characterize (G, I)-rigid tokens.

Lemma 2. *Let I be an independent set of a block graph G. Let $u \in I$. The token t placed at u is (G, I)-rigid (see Fig. 3) if and only if for every $v \in N_G(u)$, there exists a vertex $w \in \big(N_G(v) \setminus \{u\}\big) \cap I$ such that one of the following conditions holds.*

- *(i) The token t_w placed at w is $(G'', I \cap G'')$-rigid, where $G'' = G - N_G[u]$ (for example, the vertex w_1 in Fig. 3(a)).*

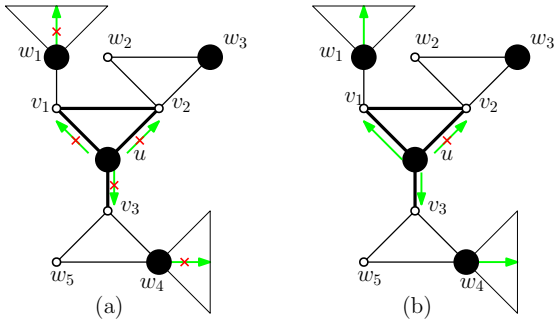


Fig. 3. (a) The token placed at u is (G, I)-rigid and (b) The token placed at u is (G, I)-movable.

(ii) The token t_w placed at w is not $(G'', I \cap G'')$-rigid; and the block B' of G containing v and w satisfies that $B' - v$ is $(G'', I \cap G'')$-confined (for example, the vertices w_3 and w_4 in Fig. 3(a)).

The next lemma says that one can compute all (G, I)-confined blocks in polynomial time, where G is a block graph and I is an independent set of G.

Lemma 3. *Let I be an independent set of a block graph G. Let $m = |E(G)|$. Let B be a block of G with $I \cap B \neq \emptyset$. Then, one can check if B is (G, I)-confined in $O(m)$ time. Consequently, one can compute $\mathsf{K}(G, I)$ in $O(m^2)$ time.*

Proof. We describe a recursive function CheckConfined(G, I, H) which returns YES if an input induced subgraph H is (G, I)-confined, where I is an independent set of G and H is either a clique or a vertex. Otherwise, it returns NO and a TS-sequence $\mathcal{S}_H$ in G which slides the token in $I \cap H$ (if exists) to a vertex in $\bigcup_{v \in V(H)} N_G(v) \setminus V(H)$. Clearly, if $I \cap H = \emptyset$ then CheckConfined(G, I, H) returns NO and there is no such $\mathcal{S}_H$ described above. Thus, we now assume that $I \cap H \neq \emptyset$. Note that since H is either a clique or a vertex, $|I \cap H| = 1$. By definition, it is clear that if $G = H$ then CheckConfined(G, I, H) returns YES. Then, we now consider the case when $G \neq H$, i.e., G contains more than one block. Let u be the unique vertex in $I \cap H$, and t_u be the token placed at u. Let $G' = G - H$ and $G'' = G - N_G[u]$. If H is a clique, we will use Lemma 1 to check if H is (G, I)-confined. On the other hand, if H contains only vertex u (i.e., $H = (\{u\}, \emptyset)$), we will use Lemma 2 to check if H is (G, I)-confined (by definition, it is equivalent to checking if t_u is (G, I)-rigid).
If H is a clique, then by Lemma 1, for every cut vertex $v \in V(H)$, we need to check if one of the conditions (i), (ii) of Lemma 1 holds. Note that since v is a cut vertex, there is at least one block $B' \neq H$ of G containing v. To check if Lemma 1(i) holds, we recursively call CheckConfined$(G', I \cap G', B' - v)$ for every block $B' \neq H$ of G containing v. If CheckConfined$(G', I \cap G', B' - v)$ returns NO for all blocks $B' \neq H$ of G containing v,

i.e. Lemma refauxlem:characterize-confined-clique(i) does not hold, we can construct a TS-sequence $\mathcal{S}_v$ in G that slides t_u to v as follows. If $u = v$ then nothing needs to be done. Thus, we assume that $u \neq v$, which then implies that $v \notin I$. In order to slide t_u to v, we need to make sure that for every block $B' \neq H$ of G containing v, if $I \cap (B' - v) \neq \emptyset$, the token in $I \cap (B' - v)$ need to be moved to a vertex not in $B' - v$ first. To do this, note that for each such B', the function CheckConfined(G', $I \cap G'$, $B' - v$) also returns a TS-sequence $\mathcal{S}_{B'-v}$ in G' that slides the token in $I \cap (B' - v)$ to a vertex in $\bigcup_{x \in V(B'-v)} N_{G'}(x) \setminus V(B' - v)$. By Proposition 4, such a sequence $\mathcal{S}_{B'-v}$ can indeed be performed in G. Hence, $\mathcal{S}_v$ can be constructed (using the results from CheckConfined(G', $I \cap G'$, $B' - v$)) by first performing all $\mathcal{S}_{B'-v}$, then performing a single step of sliding t_u to v. If Lemma 1(i) does not hold, for every $w \in N_G(v) \setminus V(H)$, we need to check if Lemma 1(ii) holds. We first need to check whether there exists a block B'' of G' containing w such that B'' is $(G', I \cap G')$-confined. This can be done by calling CheckConfined(G', $I \cap G'$, B'') for all blocks B'' of G' containing w such that $I \cap B'' \neq \emptyset$. If the result is NO for every such B'', i.e., Lemma 1(ii-1) does not hold, we still need to check if Lemma 1(ii-2) holds. To do this, we consider the following cases.

- **Case 1:** $|N_{G'}[w] \cap I| = 0$.

In this case, Lemma 1(iii) does not hold, which then implies that CheckConfined(G, I, H) returns NO. To see this, we shall construct a TS-sequence $\mathcal{S}_H$ in G that slides t_u to $w \in N_G(v) \setminus V(H)$. Indeed, $\mathcal{S}_H$ can be constructed by simply performing two steps of sliding: t_u to v, and then t_u from v to w (since $|N_{G'}[w] \cap I| = 0$).

- **Case 2:** $|N_{G'}[w] \cap I| = 1$.

Let K be the block graph obtained from G' by turning $N_{G'}[w]$ into a clique, called B_w. By Proposition 6, checking if Lemma 1(iii) holds is equivalent to checking if B_w is (K, I)-confined. In case Lemma 1(iii) holds, the construction of $\mathcal{S}_H$ can be done by first sliding the token in $N_{G'}[w] \cap I$ to some vertex not in $N_{G'}[w] \cap I$ (converting a TS-sequence in K to a TS-sequence in G' as in Proposition 6, and extending that TS-sequence to a TS-sequence in G using Proposition 4), and then use the process described in **Case 1** to slide t_u to w.

- **Case 3:** $|N_{G'}[w] \cap I| \geq 2$.

We first show how to construct an independent set J such that $I \overset{G}{\leftrightsquigarrow} J$ and $|N_{G'}[w] \cap J| \leq 1$. Note that since $|N_{G'}[w] \cap I| \geq 2$, we have $w \notin I$. The idea of this construction comes from Proposition 5 and Proposition 6. Proposition 6 indeed implies that there is at most one token t_x in $N_{G'}[w] \cap I$ that is $(G', I \cap G', N_{G'}[w])$-confined. In other words, all tokens in $N_{G'}[w] \cap I$ except t_x (if exists) can be slid to a vertex not in $N_{G'}[w]$. Now, for each block B'' of G' containing w with $I \cap B'' \neq \emptyset$, from the results of calling CheckConfined(G', $I \cap G''$, B''), we obtain a TS-sequence $\mathcal{S}_{B''}$ in G' (which can also be extended in G using Proposition 4) that moves the token in $I \cap B''$ to a vertex not in B''. Note that $\mathcal{S}_{B''}$ may or may not contain the step of sliding the token in $I \cap B''$ to w. If for some block B'' of G' containing w with $I \cap B'' \neq \emptyset$, $\mathcal{S}_{B''}$ contains such a step, then clearly it will move all other tokens in $I \cap N_{G'}[w]$ "out of" $N_{G'}[w]$ first, and then moves the token in $I \cap B''$ to w. Stop at this point,

we obtain an independent set J such that $I \overset{G}{\rightsquigarrow} J$ and $|N_{G'}[w] \cap J| = 1$. The only token in $N_{G'}[w] \cap J$ is now indeed the token placed at w. On the other hand, if for all blocks B'' of G' containing w with $I \cap B'' \neq \emptyset$, $\mathcal{S}_{B''}$ does not contain the step of sliding the token in $I \cap B''$ to w, then we simply perform all such $\mathcal{S}_{B''}$. Since G is a block graph, all such $\mathcal{S}_{B''}$ can indeed be performed independently, i.e., no sequence involves any vertex that is involved by other sequences. At the end of this process, we obtain an independent set J such that $I \overset{G}{\rightsquigarrow} J$ and $|N_{G'}[w] \cap J| = 0$. Once we have J, the checking process can indeed be done using either **Case 1** or **Case 2**. Keep in mind that the construction of J uses only the results that can be obtained from the recursive callings of the CheckConfined function.

In the above arguments, we have analyzed the cases that CheckConfined(G, I, H) returns no using Lemma 1, where H is a clique. In all other cases, CheckConfined(G, I, H) indeed returns yes (by Lemma 1).

If H contains only a single vertex u, then by Lemma 2, we need to check that for every $v \in N_G(u)$, whether there exists a vertex $w \in \big(N_G(v) \setminus \{u\}\big) \cap I$ such that one of the conditions (i), (ii) of Lemma 2 holds. Clearly, if $\big(N_G(v) \setminus \{u\}\big) \cap I = \emptyset$, one can construct a TS-sequence $\mathcal{S}_H$ that slides t_u to v by performing the single step of sliding t_u to v, and hence CheckConfined(G, I, H) returns no. Next, we consider the case when $\big(N_G(v) \setminus \{u\}\big) \cap I \neq \emptyset$. In this case, for every $w \in \big(N_G(v) \setminus \{u\}\big) \cap I$, we recursively call CheckConfined(G'', $I \cap G''$, $\{w\}$) to check if Lemma 2(i) holds. If CheckConfined(G'', $I \cap G''$, $\{w\}$) = no for all $w \in \big(N_G(v) \setminus \{u\}\big) \cap I$, we still need to check if Lemma 2(ii) holds by calling CheckConfined(G'', $I \cap G''$, $B_w - v$) for all $w \in \big(N_G(v) \setminus \{u\}\big) \cap I$, where B_w denotes the (unique) block of G containing both v, w. If CheckConfined(G'', $I \cap G''$, $B_w - v$) returns no for all $w \in \big(N_G(v) \setminus \{u\}\big) \cap I$, we can indeed return no for the function CheckConfined(G, I, H). The TS-sequence $\mathcal{S}_H$ that moves t_u to v in this case can be constructed as follows. For each $w \in \big(N_G(v) \setminus \{u\}\big) \cap I$, since CheckConfined($G''$, $I \cap G''$, $B_w - v$) returns no, there must be a TS-sequence $\mathcal{S}_{B'-v}$ in G'' (which can be extended to G using Proposition 3) that slides the token in $I \cap (B' - v)$ to a vertex in $\bigcup_{z \in V(B'-v)} N_{G'}(B' - v) \setminus V(B' - v)$. $\mathcal{S}_H$ then can be constructed by first performing all such $\mathcal{S}_{B'-v}$, and then performing a single step of sliding t_u to v. In the above arguments, we have analyzed the cases that CheckConfined(G, I, H) returns no using Lemma 2, where H is a vertex. In all other cases, CheckConfined(G, I, H) indeed returns yes (by Lemma 2).

Next, we analyze the complexity of the described algorithm. First of all, note that all the TS-sequences mentioned in the described algorithm can indeed be construction using the results from the recursive callings of the CheckConfined function. Thus, the running time of our algorithm is indeed proportional to the number of callings of the CheckConfined function. For a vertex $v \in V(G)$, let $f(v)$ be the number of calling CheckConfined *related* to v, in the sense that the function CheckConfined is either called for v or for a block containing v. Thus, the total number of callings CheckConfined is indeed bounded by

$\sum_{v \in V(G)} f(v)$. Moreover, from the described algorithm, note that $f(v)$ is at most $O(\deg_G(v))$. Hence, checking if H is (G, I)-confined takes at most $O(\sum_{v \in V(G)} \deg_G(v)) = O(m)$ time, where H is either a clique or a vertex. Consequently, since the number of blocks of G is $O(m)$, computing $\mathsf{K}(G, I)$ takes at most $O(m^2)$ time.

5 Sliding tokens on block graphs

Let G be a block graph, and let I, J be two independent sets of G. In this section, we prove the following main result of this paper.

Theorem 1. *Let (G, I, J) be an instance of the* Sliding Token *problem, where I, J are two independent sets of a block graph G. Then, one can decide if $I \overset{G}{\leftrightsquigarrow} J$ in $O(m^2)$ time, where $m = |E(G)|$.*

To prove Theorem 1, we shall describe a polynomial-time algorithm for deciding if $I \overset{G}{\leftrightsquigarrow} J$, estimate its running time, and then prove its correctness. The following algorithm checks if $I \overset{G}{\leftrightsquigarrow} J$.

- **Step 1:**
  - **Step 1-1:** If $\mathsf{K}(G, I) \neq \mathsf{K}(G, J)$, return NO.
  - **Step 1-2:** Otherwise, remove all cliques in $\mathsf{K}(G, I)$ and go to **Step 2**. Let G' be the resulting graph.
- **Step 2:** If $|I \cap F| \neq |J \cap F|$ for some component F of G', return NO. Otherwise, return YES.

We now analyze the time complexity of the algorithm. Let m, n be respectively the number of edges and vertices of G. By Lemma 3, **Step 1-1** takes at most $O(m^2)$ time. **Step 1-2** clearly takes at most $O(n)$ time. Hence, **Step 1** takes at most $O(m^2)$ time. **Step 2** takes at most $O(n)$ time. In total, the algorithm runs in $O(m^2)$ time.
The rest of this section is devoted to showing the correctness of the algorithm. First of all, the following lemma is useful.

Lemma 4. *Let I be an independent set of a block graph G. Let $w \in V(G)$. Assume that every block of G containing w is not (G, I)-confined. Then, there is at most one block B of G containing w such that $B - w$ is $(G', I \cap G')$-confined, where $G' = G - w$.*

The next lemma ensures the correctness of **Step 1-1**.

Lemma 5. *Let (G, I, J) be an instance of the* Sliding Token *problem, where I, J are two independent sets of a block graph G. Then, it is a* NO*-instance if $\mathsf{K}(G, I) \neq \mathsf{K}(G, J)$.*

In the next lemma, we claim that **Step 1-2** is correct.

Lemma 6. *Let (G, I, J) be an instance of the* Sliding Token *problem, where I, J are two independent sets of a block graph G satisfying that $\mathsf{K}(G, I) = \mathsf{K}(G, J)$. Let G' be the graph obtained from G by removing all cliques in $\mathsf{K}(G, I) = \mathsf{K}(G, J)$. Then, $I \overset{G}{\leftrightsquigarrow} J$ if and only if $I \cap G' \overset{G'}{\leftrightsquigarrow} J \cap G'$. Furthermore, $\mathsf{K}(G', I \cap G') = \mathsf{K}(G', J \cap G') = \emptyset$.*

Proof. Let $\mathcal{S} = \langle I = I_1, I_2, \ldots, I_\ell = J \rangle$ be a TS-sequence in G that reconfigures I to J. We claim that there exists a TS-sequence $\mathcal{S}'$ in G' that reconfigures $I \cap G'$ to $J \cap G'$. Note that for any independent set I of G, $I \cap G'$ forms an independent set of G'. Moreover, for $i = 1, 2, \ldots, \ell - 1$, let uv be an edge of G such that $u \in I_i \setminus I_{i+1}$ and $v \in I_{i+1} \setminus I_i$, then clearly u and v must be either both in G' or both in some block $B \in \mathsf{K}(G, I)$. Hence, the sequence $\mathcal{S}' = \langle I_1 \cap G', \ldots, I_\ell \cap G' \rangle$ reconfigures $I_1 \cap G' = I \cap G'$ to $I_\ell \cap G' = J \cap G'$.

Let $\mathcal{S}' = \langle I \cap G' = I'_1, I'_2, \ldots, I'_l = J \cap G' \rangle$ be a TS-sequence in G' that reconfigures $I \cap G'$ to $J \cap G'$. We claim that there exists a TS-sequence $\mathcal{S}$ in G that reconfigures $I = (I \cap G') \cup \bigcup_{B \in \mathsf{K}(G,I)} (I \cap B)$ to $J = (J \cap G') \cup \bigcup_{B \in \mathsf{K}(G,I)} (J \cap B)$. Note that for an independent set I' of G' and a block $B \in \mathsf{K}(G, I)$, it is not necessary that $I' \cup (I'' \cap B)$ forms an independent set of G, where I'' is an independent set of G such that $I \overset{G}{\rightsquigarrow} I''$. For a component F of G', one can construct a TS-sequence ${\mathcal{S}'}_F = \langle I'_1 \cap F, \ldots, I'_l \cap F \rangle$ in F. We now describe how to construct $\mathcal{S}$. Let $A = \bigcup_{B \in \mathsf{K}(G,I)} \bigcup_{v \in I \cap B} \big(N_G(v) \cap V(F)\big)$. For a component F of G', we consider the following cases.

- ${\mathcal{S}'}_F$ **does not involve any vertex in** A.

In this case, note that for every independent set I_F of F and a block $B \in \mathsf{K}(G, I)$, the set $I_F \cup (J \cap B)$ forms an independent set of G, where J is any independent set of G satisfying $I \overset{G}{\rightsquigarrow} J$. Thus, such a sequence ${\mathcal{S}'}_F$ above indeed can be "extended" to a TS-sequence in G.

- ${\mathcal{S}'}_F$ **involves vertices in** A.

Note that for a block $B \in \mathsf{K}(G, I)$, since G is a block graph, there is at most one vertex $v \in V(B)$ satisfying that $N_G(v) \cap V(F) \neq \emptyset$. Moreover, if there exists two vertices $u_1, u_2 \in V(F)$ such that $N_G(u_i) \cap V(B) \neq \emptyset$ $(i = 1, 2)$ then they must be adjacent to the same vertex in B. Let v be the unique vertex in $I \cap B$ and assume that $N_G(v) \cap V(F) \neq \emptyset$. Then, the token t_v placed at v must not be (G, I)-rigid. To see this, note that, if the token t placed at $u \in I$ is (G, I)-rigid, then by definition of confined cliques, any block of G containing u must be in $\mathsf{K}(G, I)$. Hence, for a block $B \in \mathsf{K}(G, I)$ and $v \in I \cap B$ with $N_G(v) \cap V(F) \neq \emptyset$, there exists a TS-sequence $\mathcal{S}'(B, v)$ in G that moves the token t_v placed at v to some other vertex in B. Since G is a block graph, if there are two of such block B, say B_1 and B_2, with $v_1 \in I \cap B_1$ and $v_2 \in I \cap B_2$, then clearly $\mathcal{S}'(B_1, v_1)$ does not involve any token which is involved by $\mathcal{S}'(B_2, v_2)$ (and vice versa).

Now, we construct a TS-sequence $\mathcal{S}$ in G that reconfigures I to J as follows. First, we perform all TS-sequence ${\mathcal{S}'}_F$ that does not involve any vertex in A. Next, for a component F with the corresponding sequence ${\mathcal{S}''}_F$ involving let $B \in \mathsf{K}(G, I)$ such that there exists a (unique) vertex $v \in I \cap B$ satisfying that $N_G(v) \cap V(F) \subseteq A$. For such component F and such block B, we first perform $\mathcal{S}'(B, v)$, then perform ${\mathcal{S}'}_F$, and then perform $\mathcal{S}'(B, v)$ in reverse order. Note that if after performing $\mathcal{S}'(B, v)$, the token t_v (originally placed at v) is placed at some vertex $w \in J$, then in the step of reversing $\mathcal{S}'(B, v)$, we do not reverse the step of sliding t_v to w. At this moment, we have reconfigured $I \cap G'$ to $J \cap G'$ in G. It remains to reconfigure $I \cap B$ to $J \cap B$ in G for each block $B \in \mathsf{K}(G, I)$,

which can be done using the observation that for any vertex $v \in J \cap B$, $N_G(v) \cap J \neq \emptyset$.
Finally, we claim that $\mathsf{K}(G', I \cap G') = \emptyset$. Similar arguments can also be applied for showing $\mathsf{K}(G', J \cap G') = \emptyset$. Assume for the contradiction that there exists some block $B' \in \mathsf{K}(G', I \cap G')$. Let v be the unique vertex in $I \cap B'$, and let B be the block of G containing B'. We consider the following cases.

- $B = B'$.

Note that since B' is a block of both G and G', it follows that B' is not (G, I)-confined. In other words, there exists a TS-sequence $\mathcal{S}$ in G that slides the token t_v placed at $v \in I \cap B'$ to some vertex not in B'. Moreover, as before, we have proved that such a TS-sequence can indeed be "restricted" to G' based on the observation that for any independent set I of G, $I \cap G'$ forms an independent set of G' and any sliding step is performed either along edges of G' or along edges of some (G, I)-confined block. Therefore, B' is not $(G', I \cap G')$-confined, a contradiction.

- $|V(B) \setminus V(B')| = 1$.

Let w be the unique vertex in $V(B) \setminus V(B')$. Note that since w is a vertex of some (G, I)-confined block $C \neq B$, the token t_v placed at v cannot be slid to w in G. Since B is not (G, I)-confined, as before, there exists a TS-sequence $\mathcal{S}$ in G that slides the token t_v placed at $v \in I \cap B'$ to some vertex not in B'. Moreover, $\mathcal{S}$ does not move t_v to w, which means that it moves t_v to some vertex of G' that is not in B'. Thus, $\mathcal{S}$ can indeed be "restricted" to G', which means that B' is indeed not $(G', I \cap G')$-confined, a contradiction.

Before proving the correctness of **Step 2**, we need some extra definitions. Let B be a block of a block graph G. A block $B' \neq B$ of G is called a *neighbor* of B if $V(B) \cap V(B') \neq \emptyset$. B is called *safe* if it has at most one cut vertex and at most one neighbor having more than one cut vertex. A vertex $v \in V(G)$ is called *safe* if it is a non-cut vertex of a safe block of G.
The next two lemmas are useful for showing the correctness of **Step 2**.

Lemma 7. *Let I be an independent set of a block graph G such that $\mathsf{K}(G, I) = \emptyset$. Let v be a safe vertex of G. Then, there exists an independent set J of G with $I \overset{G}{\rightsquigarrow} J$ and $v \in J$.*

Lemma 8. *Let I be an independent set of a block graph G such that $\mathsf{K}(G, I) = \emptyset$. Let $v \in I$ be a safe vertex of G and let B_v be the (unique) safe block of G containing v. Let G^* be the subgraph of G obtained by removing B_v. Then, $\mathsf{K}(G^*, I \cap G^*) = \emptyset$.*

The following lemma ensures the correctness of **Step 2**.

Lemma 9. *Let (G, I, J) be an instance of the* Sliding Token *problem, where I, J are two independent sets of a block graph G satisfying that $\mathsf{K}(G, I) = \mathsf{K}(G, J) = \emptyset$. Then, $I \overset{G}{\rightsquigarrow} J$ if and only if $|I| = |J|$.*

Proof. The only-if-part is trivial. We shall prove the if-part, i.e., if $|I| = |J|$ then $I \overset{G}{\leftrightsquigarrow} J$. More precisely, we claim that there exists an independent set I^* such that $I \overset{G}{\leftrightsquigarrow} I^*$ and $J \overset{G}{\leftrightsquigarrow} I^*$. Indeed, I^* can be constructed as follows. Initially, $I^* = \emptyset$.

- Pick a safe vertex v of G. (Note that the "tree-like" structure of a block graph ensures that one can always find a safe block, and hence a safe vertex.)
- Slide a token from I and a token from J to v.

Then, add v to I^*. This can be done using Lemma 7. Let I' and J' be the resulting independent sets.

- Let G' be the graph obtained by removing B_v - the (unique) block of G containing v.
- Repeat the above steps with the new triple $(G', I' \setminus \{v\}, J' \setminus \{v\})$ instead of (G, I, J). The procedure stops when there is no token to move.

The correctness of this construction is followed from Lemma 7 and Lemma 8.

Acknowledgement. The first author would like to thank Yota Otachi for his useful discussions. This work is partially supported by MEXT/JSPS Kakenhi Grant Number 26330009 and 24106004.

References

[1] van den Heuvel, J. In: The complexity of change. Cambridge University Press (2013) 127–160

[2] 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**(1) (2005) 72–96

[3] 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**(12) (2011) 1054–1065

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

[5] Mouawad, A.E., Nishimura, N., Raman, V., Wrochna, M.: Reconfiguration over tree decompositions. In Cygan, M., Heggernes, P., eds.: Parameterized and Exact Computation - IPEC 2014. Volume 8894 of LNCS., Springer (2014) 246–257

[6] Bonsma, P., Kamiński, M., Wrochna, M.: Reconfiguring independent sets in claw-free graphs. In Ravi, R., Gørtz, I., eds.: Algorithm Theory - SWAT 2014. Volume 8503 of LNCS., Springer (2014) 86–97

[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. Theoretical Computer Science **600** (2015) 132–142

[8] Fox-Epstein, E., Hoang, D.A., Otachi, Y., Uehara, R.: Sliding token on bipartite permutation graphs. In Elbassioni, K., Makino, K., eds.: Algorithms and Computation - ISAAC 2015. Volume 9472 of LNCS., Springer (2015) 237–247

[9] Hoang, D.A., Uehara, R.: Sliding tokens on a cactus. In Hong, S.H., ed.: Algorithms and Computation - ISAAC 2016. Volume 64 of LIPIcs., Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik (2016) 37:1–37:26

[10] Diestel, R.: Graph Theory. 4th edn. Volume 173 of Graduate Texts in Mathematics. Springer (2010)