The Complexity of Independent Set Reconfiguration on Bipartite Graphs

DANIEL LOKSHTANOV, University of Bergen, Norway
AMER E. MOUAWAD, American University of Beirut, Lebanon

We settle the complexity of the Independent Set Reconfiguration problem on bipartite graphs under all three commonly studied reconfiguration models. We show that under the token jumping or token addition/removal model, the problem is NP-complete. For the token sliding model, we show that the problem remains PSPACE-complete.

CCS Concepts: • Theory of computation → Graph algorithms analysis; Complexity classes; Problems, reductions and completeness;

Additional Key Words and Phrases: Independent set, vertex cover, reconfiguration, solution space, bipartite graphs

ACM Reference format:
Daniel Lokshtanov and Amer E. Mouawad. 2018. The Complexity of Independent Set Reconfiguration on Bipartite Graphs. ACM Trans. Algorithms 15, 1, Article 7 (October 2018), 19 pages.
https://doi.org/10.1145/3280825

1 INTRODUCTION

Many real-world problems present themselves in the following form: given the description of a system state and the description of a state we would "prefer" the system to be in, is it possible to transform the system from its current state into the desired one without "breaking" the system in the process? Such questions, with some generalizations and specializations, have received a substantial amount of attention under the so-called reconfiguration framework [7, 32, 35, 37]. Historically, the study of reconfiguration questions predates the field of computer science, as many classic one-player games can be formulated as reachability questions [24, 27], e.g., the 15-puzzle and Rubik’s cube. More recently, reconfiguration problems have emerged from computational problems in different areas such as graph theory [8, 20, 21], constraint satisfaction [14, 30], computational geometry [11, 26, 29], and even quantum complexity theory [13].

In this work, we focus on the reconfiguration of independent sets and vertex covers of bipartite graphs. We view an independent set as a collection of tokens placed on the vertices of a graph such that no two tokens are adjacent. This gives rise to three natural adjacency relations between independent sets (or token configurations), also called reconfiguration steps. In the token addition/removal (TAR) model, first introduced by Ito et al. [20], we are allowed to either add or
remove one token at a time as long as there are at least \( k \) (non-adjacent) tokens on the graph at all times. In the token jumping (TJ) model, introduced by Kamiński et al. [25], a single reconfiguration step consists of first removing a token on some vertex \( u \) and then immediately adding it back on any other vertex \( v \), as long as no two tokens become adjacent. The token is said to jump from vertex \( u \) to vertex \( v \). Finally, in the token sliding (TS) model, introduced by Hearn and Demaine [17], two independent sets are adjacent if one can be obtained from the other by a token jump from vertex \( u \) to vertex \( v \) with the additional requirement of \( uv \) being an edge of the graph. The token is then said to slide from vertex \( u \) to vertex \( v \) along the edge \( uv \). Note that, in both the TJ and TS models, the size of independent sets is fixed, while the TAR model only enforces a lower bound. Generally speaking, in the \( M \)-independent set reconfiguration (\( M \)-ISR) problem, where \( M \in \{ \text{TAR}, \text{TJ}, \text{TS} \} \), we are given a graph \( G \) and two independent sets \( I \) and \( J \) of \( G \). The goal is to determine whether there exists a sequence of reconfiguration steps—a reconfiguration sequence—that transforms \( I \) into \( J \) (where the reconfiguration step depends on the model). \( M \)-ISR has been extensively studied under the reconfiguration framework, albeit under different names [5, 6, 10, 12, 22, 23, 25, 28, 31]. It is known that the problem is PSPACE-complete for all three models, even on restricted graph classes such as graphs of bounded bandwidth/pathwidth [36] and planar graphs [17]. A popular open question related to \( M \)-ISR is whether the problem becomes polynomial-time solvable on bipartite graphs [12, 18, 19, 31]. A few positive results for subclasses of bipartite graphs are known. For instance, it was shown by Demaine et al. [10] that TS-ISR can be solved in polynomial time on trees. Fox-Epstein et al. [12] gave polynomial-time algorithms for solving TS-ISR on bipartite permutation and bipartite distance-hereditary graphs, and conjectured that the problem remains polynomial-time solvable on bipartite graphs. Mouawad et al. [31] studied the shortest path variant of TAR-ISR, where we seek a shortest reconfiguration sequence, and showed that it is NP-hard on bipartite graphs. They asked whether the problem is in NP and whether the problem remains hard without any length restrictions. We settle the complexity of \( M \)-ISR (and the shortest path variant) on bipartite graphs under all three models. We show that under the token jumping or token addition/removal model, the problem is NP-complete. For the token sliding model, we show that the problem remains PSPACE-complete. Our NP-completeness result comes as somewhat of a surprise, as reconfiguration problems are typically in \( P \) or PSPACE-complete [35]. To the best of our knowledge, TAR-ISR on bipartite graphs is the first “natural” NP-complete reconfiguration problem (that asks for the existence of a reconfiguration sequence of any length).

It is known [25] that the token addition/removal model generalizes the token jumping model in the following sense. There exists a sequence between two independent sets \( I \) and \( J \), with \(|I| = |J|\), under the TJ model if and only if there exists a sequence between them under the TAR model, with \( k = |I| - 1 \). Hence, we only consider the TAR model. In addition, TAR-ISR is easily seen to be equivalent to the following problem (by taking complements of sets), namely vertex cover reconfiguration (VCR). We are given an \( n \)-vertex graph \( G \), an integer \( k \), and two vertex covers of \( G \), \( S \), and \( T \), of size at most \( k \). The goal is to determine whether there exists a sequence \( \sigma = \langle Q_0, \ldots, Q_t \rangle \) satisfying the following:

\(- Q_0 = S \) and \( Q_t = T \);
\(- Q_i \) is a vertex cover of \( G \) and \( |Q_i| \leq k \), for \( 0 \leq i \leq t \);
\(- |Q_i \triangle Q_{i+1}| = 1 \), where \( 0 \leq i < t \) and \( Q_i \triangle Q_{i+1} = (Q_i \setminus Q_{i+1}) \cup (Q_{i+1} \setminus Q_i) \) denotes the symmetric difference of \( Q_i \) and \( Q_{i+1} \).

An alternative perspective on the VCR problem (and reconfiguration problems in general) is via the notion of the reconfiguration graph \( R_k(G) \). Nodes in \( R_k(G) \) represent vertex covers of \( G \) of size at most \( k \) and two nodes \( Q \) and \( Q' \) are connected by an edge whenever \( |Q \triangle Q'| = 1 \). In other
The Complexity of Independent Set Reconfiguration on Bipartite Graphs

Fig. 1. Example of a non-monotone sequence going to a local minimum ($k = 44$). Each edge $xy$ corresponds to a biclique with $x$ vertices on one side and $y$ vertices on the other. Vertices colored gray belong to the vertex cover. Note that vertices that are added or removed more than once do not belong to $S \Delta T$.

words, $Q'$ can be obtained from $Q$ by the addition or removal of a single vertex. An edge in $R_k(G)$ is sometimes referred to as a reconfiguration step, and a walk or path in this graph is a reconfiguration sequence. An equivalent formulation of VCR is then to determine whether $S$ and $T$ belong to the same connected component of $R_k(G)$.

To prove NP-hardness of VCR, we consider instances $(G, S, T, k)$ of the problem where $G$ is an $n$-vertex bipartite graph with bipartition $(L, R)$, starting set $S = L$, and target set $T = R$. Informally, we call such instances the “left-to-right instances.” We show that there is a reconfiguration sequence from $L$ to $R$ in $R_k(G)$ if and only if the treewidth of the cobipartite graph $\overline{G}$ is at most $k$, where $\overline{G}$ is obtained from $G$ by adding all edges between vertices in $L$ and adding all edges between vertices in $R$. We obtain the aforementioned equivalence by relating left-to-right instances of VCR to the cops-and-robber game played on $\overline{G}$. The seminal result of Seymour and Thomas [34] establishes the equivalence between the cops-and-robber game and computing the treewidth of the underlying graph $\overline{G}$. That is, the number of cops needed to catch a robber in $\overline{G}$ is exactly equal to the treewidth of $\overline{G}$ plus one. Computing treewidth is known to be NP-complete, even for cobipartite graphs [2, 4, 38]. This implies that VCR is NP-complete when restricted to left-to-right instances or, more generally, to instances where $S \cap T = \emptyset$. To show membership in NP for instances that are not necessarily left-to-right, we prove that the diameter of every connected component of $R_k(G)$, for $G$ bipartite, is at most $O(n^4)$. While we believe that $O(n^4)$ is an overestimation, we know that there are instances of VCR where vertices have to be “touched” (added or removed) more than once. An example of such an instance (where vertices need to be added and removed twice) is shown in Figure 1.

Finally, we consider ISR under the token sliding model. This problem is commonly known in the literature as the Token Sliding problem. We prove PSPACE-completeness of Token Sliding in bipartite graphs by a reduction from a variant of the Word Reconfiguration problem, first introduced by Wrochna [36].

2 PRELIMINARIES

We denote the set of natural numbers by $\mathbb{N}$. For $n \in \mathbb{N}$, we let $[n] = \{1, 2, \ldots, n\}$. We assume that each graph $G$ is finite, simple, and undirected. We let $V(G)$ and $E(G)$ denote the vertex set and edge set of $G$, respectively. We use $[V(G)]^{\leq k}$ to denote the set of all subsets of $V(G)$ of cardinality at most

ACM Transactions on Algorithms, Vol. 15, No. 1, Article 7. Publication date: October 2018.
k, where k is a non-zero positive integer. The open neighborhood of a vertex v is denoted by \( N_G(v) = \{ u \mid uv \in E(G) \} \) and the closed neighborhood by \( N_G[v] = N_G(v) \cup \{ v \} \). For a set of vertices \( Q \subseteq V(G) \), we define \( N_G(Q) = \{ v \mid \exists u \in Q, uv \in E(G) \} \) and \( N_G[Q] = N_G(Q) \cup Q \). The subgraph of G induced by Q is denoted by \( G[Q] \), where \( G[Q] \) has vertex set \( Q \) and edge set \( \{ uv \in E(G) \mid u, v \in Q \} \).

We let \( G - Q = G[V(G) \setminus Q] \). For a pair of vertices \( u \) and \( v \) in \( V(G) \), by \( \text{dist}_G(u, v) \), we denote the length of a shortest path between \( u \) and \( v \) in G (measured in number of edges and set to \( \infty \) if \( u \) and \( v \) belong to different connected components). Given a graph \( G \) and a set \( Q \subseteq V(G) \), by \( \text{cliquifying} \), we denote the operation that adds all missing edges between vertices in \( Q \), resulting in a new graph \( G' \). Given \( G \) and a vertex \( u \in V(G) \), by \( \text{duplicating} \) \( u \) we denote the operation that adds a new vertex \( v \) connected to all vertices in \( N_G(u) \), resulting in a new graph \( G' \). In \( G' \), \( u \) and \( v \) are twins. A graph \( G \) is bipartite if the vertex set of \( G \) can be partitioned into two disjoint sets \( L \) and \( R \), i.e., \( V(G) = L \cup R \), where \( G[L] \) and \( G[R] \) are edgeless. A graph \( G \) is cobipartite if the vertex set of \( G \) can be partitioned into two disjoint sets \( L \) and \( R \), where \( G[L] \) and \( G[R] \) are cliques.

Canonical Path and Tree Decompositions. A tree decomposition \([3, 33]\) of a graph \( G \) is a pair \( T = (T, \{B_t\}_{t \in V(T)}) \), where \( T \) is a tree whose every vertex \( t \) is assigned a vertex subset \( B_t \subseteq V(G) \), called a bag, such that the following three conditions hold.

- (P1) \( \bigcup_{t \in V(T)} B_t = V(G) \), in other words, every vertex of \( G \) is in at least one bag;
- (P2) For every \( uv \in E(G) \), there exists \( t \in V(T) \) such that bag \( B_t \) contains \( u \) and \( v \);
- (P3) For every \( u \in V(G) \), the set \( T_u = \{ t \in V(T) \mid u \in B_t \} \), i.e., the set of vertices whose corresponding bags contain \( u \), induces a connected subtree of \( T \).

The width of tree decomposition \( T = (T, \{B_t\}_{t \in V(T)}) \) equals \( \max_{t \in V(T)} |B_t| - 1 \), that is, the maximum size of any bag minus 1. The treewidth of a graph \( G \), denoted by \( \text{tw}(G) \), is the minimum possible width of a tree decomposition of \( G \). It is convenient to think of tree decompositions as rooted trees. That is, for a tree decomposition \( T \), we distinguish one vertex \( r \) of \( T \), which will be the root of \( T \). This introduces natural parent-child and ancestor-descendant relations in the tree \( T \). We will say that such a rooted tree decomposition is canonical or nice if the following conditions are satisfied. \( B_r = \emptyset \) and \( B_t = \emptyset \) for every leaf \( \ell \) of \( T \). In other words, all the leaves as well as the root contain empty bags. Furthermore, every non-leaf vertex of \( T \) is of one of the following three types.

- **Introduce vertex**: a vertex \( t \) with exactly one child \( t' \) such that \( B_t = B_{t'} \cup \{ v \} \) for some vertex \( v \notin B_{t'} \); we say that \( v \) is introduced at \( t \);
- **Forget vertex**: a vertex \( t \) with exactly one child \( t' \) such that \( B_t = B_{t'} \setminus \{ w \} \) for some vertex \( w \in B_{t'} \); we say that \( w \) is forgotten at \( t \);
- **Join vertex**: a vertex \( t \) with two children \( t_1 \) and \( t_2 \) such that \( B_t = B_{t_1} \cup B_{t_2} \).

A (nice) path decomposition of a graph \( G \) is simply a (nice) tree decomposition where \( T \) must be a path. It will be convenient to denote a (nice) path decomposition by a sequence \( P = (B_1, B_2, \ldots, B_p) \). The pathwidth of a graph \( G \), denoted by \( \text{pw}(G) \), is the minimum possible width of a path decomposition of \( G \). Note that for a nice path decomposition, we only have introduce and forget vertices.

**Proposition 2.1.** For every clique \( C \) of a graph \( G \) and any (nice) path or tree decomposition of \( G \), there exists a bag \( B \) in the decomposition such that \( V(C) \subseteq B \).

### 3 NP-COMPLETENESS UNDER THE TOKEN ADDITION/REMOVAL MODEL

We denote an instance of the **Vertex Cover Reconfiguration** problem by \((G, S, T, k)\), where \( G \) is the input graph, \( S \) and \( T \) are the source and target vertex covers, respectively, and \( k \) is the
maximum allowed capacity. The reconfiguration graph $R_k(G)$ contains a node for each vertex cover $Q$ of $G$ of size at most $k$. Two nodes $Q$ and $Q'$ are adjacent in $R_k(G)$ whenever $|Q \triangle Q'| = 1$. To avoid confusion, we refer to nodes in reconfiguration graphs, as distinguished from vertices in the input graph. By a slight abuse of notation, we use upper case letters to refer to both a node in the reconfiguration graph as well as the corresponding vertex cover. For any node $Q \in V(R_k(G))$, the quantity $k - |Q|$ corresponds to the available capacity at $Q$. Given $Q \in V(R_k(G))$, let $C$ denote the connected component of $R_k(G)$ containing $Q$. We say $Q$ is a local minimum if there exists no $Q' \in V(C)$ such that $|Q'| < |Q|$.

We use two representations of reconfiguration sequences. The first representation consists of a sequence of vertex covers, $\sigma = \langle Q_0, Q_1, \ldots, Q_{t-1}, Q_t \rangle$. Given $\sigma$, we associate it with a sequence of edit operations as follows. We assume all vertices of $G$ are labeled from 1 to $n$, i.e., $V(G) = \{v_1, v_2, \ldots, v_n\}$. We let $M^+ = \{v^+_1, \ldots, v^+_n\}$ and $M^- = \{v^-_1, \ldots, v^-_n\}$ denote the sets of addition markers and removal markers, respectively. An edit sequence $\eta = (m_1, m_2, \ldots, m_t)$ is an ordered sequence of elements obtained from the full set of markers $M^+ \cup M^-$, i.e., $m_i \in M^+ \cup M^-$. Here, $v^+_i$ stands for “add vertex $v_i$” and $v^-_i$ stands for “remove vertex $v_i$,” $1 \leq i, j \leq n$. We say a vertex $v \in V(G)$ is touched in the course of a reconfiguration sequence if it is either added (or removed ($v^- \in \eta$) at least once. We say a reconfiguration sequence is monotone if it touches no vertex more than once. Note that a monotone sequence from $S$ to $T$ can only remove vertices of $S \setminus T$ and add vertices of $T \setminus S$ (in some order). Given an edit sequence $\eta$ and a vertex cover $S$, we say $\eta$ is valid at $S$ if applying $\eta$ starting from $S$ results in a reconfiguration sequence, i.e., the sequence corresponds to a walk in the reconfiguration graph $R_k(G)$. In other words, $\eta = (m_1, m_2, \ldots, m_{t-1}, m_t)$ is valid at $S$ if and only if there exists a sequence $\sigma$ such that $Q_0 = S, Q_1, \ldots, Q_{t-1}, Q_t$ such that $Q_i$ is a vertex cover of $G$ of size at most $k$, for $0 \leq i \leq t$, and if $m_{i+1} \in \{v^+_j, v^-_j\}$, then $Q_i \Delta Q_{i+1} = \{v_j\}$, for $0 \leq i < t$. We let $\eta(S)$ denote the vertex cover obtained after applying $\eta$ starting at $S$. Given $\eta$ and $\eta'$, we use $\eta \cdot \eta'$ to denote the concatenation of both sequences.

**NP-hardness**

To prove NP-hardness we will show an equivalence between VCR and the cops-and-robber game. Let us start by formally describing the game. The game is played on a finite, undirected, and connected graph $G$. Throughout the game, a robber is standing on some vertex in $V(G)$. The robber can, at any time, run at “infinite” speed to any other vertex along a path of the graph. However, running through a cop is not permitted. There are $k$ cops; each of them is either standing on a vertex or in a helicopter (temporarily removed from the graph). The cops can see the robber at all times. The objective of the cops is to land a helicopter on the vertex occupied by the robber, and the robber’s objective is to elude capture. Note that, since they are equipped with helicopters, cops are not constrained to moving along paths of the graph. The robber can see the helicopter approaching the landing spot and can run to a new vertex before the helicopter actually lands (when possible). Therefore, the only way for the cops to capture the robber is by having a cop land on the vertex $v$ occupied by the robber while all neighbors of $v$ are also occupied by cops.

A state in the game is a pair $(X, F)$, where $X \in [V(G)]^{\leq k}$ and $F$ is an $X$-flap, i.e., the vertex set of a component of $G - X$. $X$ is the set currently occupied by cops and $F$ tells us where the robber is; since he can run arbitrarily fast, the only information we need is which component of $G - X$ contains him. The initial state is $(X_0, F_0)$, where $X_0 = \emptyset$ and $F_0$ is the flap (or the vertex set of the connected component) chosen by the robber. At round $i \geq 1$ of the game, we have $(X_{i-1}, F_{i-1})$ and the cops pick a new set $X_i \subseteq [V(G)]^{\leq k}$ such that $|X_i \Delta X_{i-1}| = 1$; either a helicopter lands on a vertex or a cop leaves the graph on a helicopter. Then, the robber chooses (if possible) an $X_i$-flap $F_i$ such that $F_i \subseteq F_{i-1}$ or $F_{i-1} \subseteq F_i$. If $F_{i-1} \subseteq X_i$, then the cops win. Otherwise, the game continues
with round $i + 1$. If there is a winning strategy for the cops, we say that $\leq k$ cops can search the graph. If, in addition, the cops can always win in such a way that the sequence $(X_0, X_1, \ldots)$ satisfies $X_i \cap X_{i'} \subseteq X_i$, for $i \leq i' \leq i''$, then we say that $\leq k$ cops can monotonely search the graph. In other words, a monotone strategy implies that cops never return to a vertex that has been previously vacated. The following theorem is due to Seymour and Thomas [34].

**Theorem 3.1.** Let $G$ be a graph and $k$ be a non-zero positive integer. Then, the following are equivalent:

- $\leq k$ cops can search $G$;
- $\leq k$ cops can monotonely search $G$;
- $G$ has treewidth at most $k - 1$.

It is well-known that computing the pathwidth or the treewidth of a graph is an NP-hard problem [2]. In what follows, we let $(G, S = L, T = R, k)$ be an instance of VCR, where $G$ is bipartite, $V(G) = L \cup R$, $S = L$, and $T = R$ (a left-to-right instance). We let $\overline{G}$ be the cobipartite graph obtained from $G$ by first cliquifying $L$ and then cliquifying $R$. We let $V(\overline{G}) = \overline{L} \cup \overline{R}$. The treewidth of a cobipartite graph is known to be equal to its pathwidth [15]. Moreover, it is NP-hard to determine, given a cobipartite graph $\overline{G}$ and an integer $k$, whether $\overline{G}$ has treewidth at most $k$ [2, 4, 38]. Lemmas 3.2 and 3.3 below, combined with Theorem 3.1 and the NP-hardness of computing treewidth of cobipartite graphs, imply the NP-hardness of VCR.

**Lemma 3.2.** If $L$ and $R$ belong to the same connected component of $R_k(G)$, then $k$ cops can search the cobipartite graph $\overline{G}$.

**Proof.** Let $(Q_0 = L, Q_1, \ldots, Q_{t-1}, Q_t = R)$ denote a reconfiguration sequence from $L$ to $R$. Let $L = \overline{L} = \{u_1, \ldots, u_t\}$ and $R = \overline{R} = \{v_1, \ldots, v_t\}$. We claim that $(X_0 = \{u_1\}, X_1 = X_0 \cup \{u_2\}, \ldots, X_t = X_{t-1} \cup \{u_t\} = \overline{L}, X_{t+1} = Q_1, \ldots, X_{t+t} = Q_t = \overline{R})$ is a winning strategy for the cops (on $\overline{G}$). Since all vertex covers (of $G$) in the reconfiguration sequence have size at most $k$, it follows that $|X_i| \leq k$, $0 \leq i \leq t$. Note that at round $i = 1$, the robber must move to some vertex in $\overline{R}$ (as all vertices in $\overline{L}$ are occupied by cops). Moreover, at any round $i > 1$, the robber cannot move to a vertex in $\overline{L}$. Assume otherwise. Then, there exists some path starting at the robber’s vertex $v^* \in \overline{R}$ and ending at some vertex $u^* \in \overline{L}$. Let $uv$ be the first edge on this path that takes the robber from $\overline{R}$ to $\overline{L}$. Both $u$ and $v$ are not occupied by cops, contradicting the fact that $Q_i$ is vertex cover of $G$. Hence, the robber is captured at round $t + 1$, i.e., when all of $\overline{R}$ is occupied by cops.

**Lemma 3.3.** If $k$ cops can search the cobipartite graph $\overline{G}$, then $L$ and $R$ belong to the same connected component of $R_k(G)$. Moreover, there exists a monotone reconfiguration sequence from $L$ to $R$ in $R_k(G)$.

**Proof.** As there exists a winning strategy for the cops, by Theorem 3.1, we can assume that $\overline{G}$ has treewidth at most $k - 1$. Since the treewidth of a cobipartite graph is equal to its pathwidth [15], let $P = \langle B_1, B_2, \ldots, B_p \rangle$ be a nice path decomposition of $\overline{G}$ of width at most $k - 1$. Combining Proposition 2.1 with the fact that $\overline{G}[\overline{L}]$ and $\overline{G}[\overline{R}]$ are cliques, we know that there exists $q$ and $q'$ such that $1 \leq q < q' \leq p$ and either $\overline{L} \subseteq B_q$ and $\overline{R} \subseteq B_{q'}$ or $\overline{R} \subseteq B_q$ and $\overline{L} \subseteq B_{q'}$ (if $q = q'$, then $k \geq |\overline{L}| + |\overline{R}| = |V(\overline{G})| = |V(G)|$ and the lemma trivially holds). Assume, without loss of generality, that $\overline{L} \subseteq B_q$ and $\overline{R} \subseteq B_{q'}$ (if the assumption does not hold, then we consider the reverse path decomposition). Let $q^* = q' - q$, $Z = B_q \setminus \overline{L} = \{z_1, \ldots, z_{|Z|}\}$, and $Y = B_{q'} \setminus \overline{R} = \{y_1, \ldots, y_{|Y|}\}$. We define a monotone reconfiguration sequence $\sigma$ from $L$ to $R$ in $R_k(G)$. Informally, $\sigma$ starts from $L$, adds vertices until reaching $B_q$, uses the bags between $B_q$ and $B_{q'}$ as reconfiguration steps, and finally removes vertices to reach $R$. Formally, consider the following three sequences:
\[ \sigma_1 = (Q_0, Q_1, \ldots, Q_{|Z|-1}, Q_1), \text{ where } Q_0 = \overline{L} \text{ and } Q_i = Q_{i-1} \cup \{z_i\}, 1 \leq i \leq |Z| \] (note that \( Q_{|Z|} = B_q \));
\[ \sigma_2 = (Q_{|Z|+1}, Q_{|Z|+2}, \ldots, Q_{|Z|+q^*-1}, Q_{|Z|+q^*}), \text{ where } Q_{|Z|+i} = B_{q+i}, 1 \leq i \leq q^* \] (note that \( Q_{|Z|+q^*} = B_{q+q^*} = B_q \));
\[ \sigma_3 = (Q_{|Z|+q^*+1}, Q_{|Z|+q^*+2}, \ldots, Q_{|Z|+q^*+|Y|-1}, Q_{|Z|+q^*+|Y|}), \text{ where } Q_{|Z|+q^*+1} = Q_{|Z|+q^*} \setminus \{y_1\} \text{ and } Q_{|Z|+q^*+i} = Q_{|Z|+q^*+i-1} \setminus \{y_1\}, 1 < i \leq |Y| \] (note that \( S_{|Z|+q^*+|Y|} = S_{|Z|+q^*} \setminus Y = B_{q'} \setminus Y = \overline{R} \)).

We claim that \( \sigma = \sigma_1 \cdot \sigma_2 \cdot \sigma_3 \) is a monotone reconfiguration sequence from \( L \) to \( R \) in \( R_k(G) \). First, note that, by our construction of \( \sigma_1 \) and \( \sigma_2 \) and the fact that \( P \) is a nice path decomposition, we have \( |Q_0 \Delta Q'| = 1 \) for any two consecutive sets in \( \sigma \). Second, since the size of the bags is at most \( k \), all the sets in \( \sigma \) have size at most \( k \); all sets appearing before \( B_q = Q_{|Z|} \) and after \( B_{q'} = Q_{|Z|+q^*} \) have size strictly less than \( |B_q| \) and \( |B_{q'}| \), respectively. Moreover, as both \( L \) and \( R \) are vertex covers of \( G \), it follows that \( Q_i \) is also a vertex cover of \( G \) for \( i \leq |Z| \) (\( L \) is a subset of every such \( Q_i \)) and \( j \geq |Z| + q^* \) (\( R \) is a subset of every such \( Q_j \)). Hence, it remains to show that every set \( Q_i \) in \( \sigma_2 \) is a vertex cover of \( G \). Assume that there exists a set \( Q_i \) in \( \sigma_2 \) that is not a vertex cover of \( G \). Then, there exists an uncovered edge \( uv \) with \( u \in L \) and \( v \in R \). Since \( L \subseteq Q_{|Z|} \) and \( R \subseteq Q_{|Z|+q^*} \), it must be the case that \( u \notin Q_i (u \text{ was forgotten}) \) and \( v \notin Q_i (v \text{ was not yet introduced}) \). This implies that there exists some other bag \( B_j, j \neq i \), covering the edge \( uv \) (by property (P2) of tree decompositions). But in both of the cases \( j < i \) and \( j > i \), we contradict property (P3) of tree decompositions, as the bags containing \( u \) (and \( v \)) must appear consecutively in \( P \). Finally, the monotonicity of \( \sigma \) follows from our construction (for \( \sigma_1 \) and \( \sigma_3 \)) and property (P3) of tree decompositions (for \( \sigma_2 \)).

**Theorem 3.4.** Let \((G, S, T, k)\) be an instance of VCR where \( S \cap T = \emptyset \). Let \( \overline{G} \) be the graph obtained from \( G \) by first cliquifying \( S \) and then cliquifying \( T \). Then, the following are equivalent:

- There exists a reconfiguration sequence from \( S \) to \( T \) in \( R_k(G) \);
- There exists a monotone reconfiguration sequence from \( S \) to \( T \) in \( R_k(G) \);
- \( \overline{G} \) has treewidth/pathwidth at most \( k - 1 \).

**Proof.** Note that any graph \( G \) having two disjoint vertex covers \( S \) and \( T \) must be bipartite; there can be no edges with both endpoints in \( S \), no edges with both endpoints in \( T \), and any vertex in \( V(G) \setminus (S \cup T) \) must be isolated. We can safely delete all isolated vertices in \( V(G) \setminus (S \cup T) \) to obtain a graph \( G' \). In other words, \( R_k(G') \) is an induced subgraph of \( R_k(G) \). Hence, combining Lemmas 3.2 and 3.3 with Theorem 3.1, we get the claimed equivalences.

Theorem 3.4 implies that VCR on bipartite graphs is NP-complete if we restrict ourselves to instances where \( S \cap T = \emptyset \). Hence, both TAR-ISR and TJ-ISR are NP-hard on bipartite graphs.

**Membership in NP**

To prove membership in NP (without any restrictions on \( S \cap T \)), we prove the following.

**Theorem 3.5.** Let \( G \) be an \( n \)-vertex bipartite graph and let \( k \) be a non-zero positive integer. Let \( S \) and \( T \) be two vertex covers of \( G \) of size at most \( k \). If \( \text{dist}_{R_k(G)}(S, T) \) is finite, then \( \text{dist}_{R_k(G)}(S, T) = O(n^k) \).

We proceed in two stages. In the first stage, we prove that finding a reconfiguration sequence from a vertex cover \( S \) to any vertex cover of size at least one less can be done in a monotone way. Formally, we show the following.

**Lemma 3.6.** Given an \( n \)-vertex bipartite graph \( G \), an integer \( k \), and a vertex cover \( S \) of \( G \) of size at most \( k \), then either \( S \) is a local minimum in \( R_k(G) \) or there exists a vertex cover \( S' \) such that \( |S'| < |S| \) and \( \text{dist}_{R_k(G)}(S, S') \leq n \).

ACM Transactions on Algorithms, Vol. 15, No. 1, Article 7. Publication date: October 2018.
Assuming Lemma 3.6, we know that given $S$ and $T$, we can, in $O(n^2)$ steps, reconfigure $S$ and $T$ to local minima $S'$ and $T'$, respectively (by repeatedly applying Lemma 3.6 as long as possible). If $S$ and $T$ live in a connected component of $R_k(G)$ with only one local minimum, then we are done. Unfortunately, this is not true in general. We remedy the situation by “perturbing” the vertex covers of $G$ to guarantee uniqueness in the following sense. Let $S^*$ be a local minimum in some connected component $C$ of $R_k(G)$. We let $H$ be the graph obtained from $G$ by duplicating each vertex $u \in S^*$ a total of $2n - 1$ times (resulting in $2n$ copies of $u$) and duplicating each vertex $v \notin S^*$ a total of $2n$ times (resulting in $2n + 1$ copies of $v$). We set $\mu = 2nk + 2n - 1$. We call $H$ the perturbation of $G$ with respect to $S^*$. For a vertex $v \in V(G)$, we define $f(v)$ to be the set of twins in $V(H)$ corresponding to $v$. By a slight abuse of notation, we generalize the function $f$ to sets as follows. For $X \subseteq V(G)$, we let $f(X) = \bigcup_{x \in X} f(x)$. Finally, for $\hat{X} \subseteq V(H)$, we define the inverse of $f$ as $f^{-1}(\hat{X}) = \{v \in V(G) \mid f(v) \subseteq \hat{X}\}$. Intuitively, we will show that $R_\mu(H)$ “almost” preserves the distances from $R_k(G)$ and that the component of interest in $R_\mu(H)$ has a unique local minimum, i.e., the component containing $f(S^*)$ will have $f(S^*)$ as its unique local minimum. In more technical terms, one can think of $R_k(G)$ as having a low distortion embedding into $R_\mu(H)$. The properties we require from the graph $H$ are captured in the following lemmas. Note that $|V(H)| \leq (2n + 1)n = O(n^2)$ and $\mu = 2nk + 2n - 1 = O(n^2)$.

**Lemma 3.7.** If $Q$ is a vertex cover of $G$, then $f(Q)$ is a vertex cover of $H$ and $2n|Q| \leq |f(Q)| \leq (2n + 1)|Q|$. Moreover, if $\hat{Q}$ is a vertex cover of $H$, then $f^{-1}(\hat{Q})$ is a vertex cover of $G$ and $|f^{-1}(\hat{Q})| \leq \lceil \frac{|\hat{Q}|}{2n} \rceil$.

**Proof.** Every edge $uv \in E(G)$ corresponds to a biclique in $H$, i.e., $H[f(u) \cup f(v)]$ is a biclique. By our construction of $H$, there is a surjective mapping from edges in $H$ to edges in $G$. That is, we can map every edge $yz \in E(H)$ to the unique edge $uv \in E(G)$ such that $yz \in E(H[f(u) \cup f(v)])$. Moreover, every edge of $G$ is mapped to (in the sense described above) by at least one edge of $H$. In other words, for all $uv \in E(G)$, there exists at least one edge $yz \in E(H)$ such that $yz \in E(H[f(u) \cup f(v)])$, and for all other edges $u'v' \in E(G)$, we have $yz \notin E(H[f(u') \cup f(v')])$.

If $Q$ is a vertex cover of $G$, then, for every edge $uv \in E(G)$, either $u \in Q$ or $v \in Q$. This implies that either $f(u) \subseteq f(Q)$ or $f(v) \subseteq f(Q)$. Therefore, given the surjective mapping from $E(H)$ to $E(G)$, $f(Q)$ must be a vertex cover of $H$. Since $|f(v)| = 2n$ if $v \in S^*$ and $|f(v)| = 2n + 1$ otherwise, we have $2n|Q| \leq |f(Q)| \leq (2n + 1)|Q|$. Similarly, if $\hat{Q}$ is a vertex cover of $H$, and since $H[f(u) \cup f(v)]$ is a biclique, then either $f(u) \subseteq \hat{Q}$ or $f(v) \subseteq \hat{Q}$ (for all $uv \in E(G)$). Therefore, either $u \in f^{-1}(\hat{Q})$ or $v \in f^{-1}(\hat{Q})$, as needed. To see why $|f^{-1}(\hat{Q})| \leq \lceil \frac{|\hat{Q}|}{2n} \rceil$, note that for every vertex cover $\hat{Q}$ of $H$, we have $|\hat{Q}| = (2n + 1)|X| + 2n|Y| + |Z|$, where $X = \{v \in V(G) \setminus S^* \mid f(v) \subseteq \hat{Q}\}$, $Y = \{v \in S^* \mid f(v) \subseteq \hat{Q}\}$, and $Z = \hat{Q} \setminus \bigcup_{v \in X \cup Y} f(v)$. Hence, $|\hat{Q}| = 2n(|X| + |Y|) + |X| + |Z|$ and $|f^{-1}(\hat{Q})| = |X| + |Y| \leq \lceil \frac{|\hat{Q}|}{2n} \rceil$. □

**Corollary 3.8.** Let $Q$ be a vertex cover of $G$ and let $\hat{Q}$ be a vertex cover of $H$.

- If $|Q| \leq k$, then $|f(Q)| \leq (2n + 1)k = 2nk + k < \mu$;
- If $|Q| \geq k + 1$, then $|f(Q)| \geq (k + 1)2n = 2nk + 2n > \mu$;
- If $|\hat{Q}| \leq k$, then $|f^{-1}(\hat{Q})| \leq \lceil \frac{2nk + 2n}{2n} \rceil \leq k$;

**Lemma 3.9.** For all $S, T \in V(R_k(G))$, if $\text{dist}_{R_k(G)}(S, T)$ is finite, then $\text{dist}_{R_\mu(H)}(f(S), f(T))$ is finite and $\text{dist}_{R_k(G)}(S, T) \leq \text{dist}_{R_\mu(H)}(f(S), f(T)) \leq (2n + 1) \cdot \text{dist}_{R_k(G)}(S, T)$.

---

1Interestingly, the “bipartiteness” of the graph $G$ is only required in the first stage of our proof (Lemma 3.6). The perturbation stage holds for any graph class that is closed under the operation of introducing (false) twins.
The Complexity of Independent Set Reconfiguration on Bipartite Graphs 7:9

Proof. Let \( \sigma = (Q_0 = S, Q_1, \ldots, Q_{t-1}, Q_t = T) \) denote a shortest reconfiguration sequence from \( S \) to \( T \) in \( R_k(G) \). By Corollary 3.8, we know that \( |f(Q_i)| \leq \mu \), and, therefore, \( f(Q_i) \in V(\mu(H)) \) for all \( i \). Hence, to obtain a corresponding reconfiguration sequence \( \hat{\sigma} \) in \( R_\mu(H) \) (from \( f(S) \) to \( f(T) \)), it is enough to replace the addition or removal of any vertex \( v \) by the addition or removal of \( f(v) \) (one vertex at a time). It is not hard to see that every set in \( \hat{\sigma} \) is a vertex cover of \( \hat{G} \) of size at most \((2n+1)k < \mu \) (and the size of the symmetric difference of consecutive sets is exactly one).

Assume that there exists a reconfiguration sequence \( \hat{\sigma} = (\hat{Q}_0 = f(S), \hat{Q}_1, \ldots, \hat{Q}_{t-1}, \hat{Q}_t = f(T)) \) from \( f(S) \) to \( f(T) \) in \( R_\mu(H) \) such that \( |\hat{\sigma}| < |\sigma| \). Consider the sequence \( \sigma' \) obtained from \( \hat{\sigma} \) by replacing every set \( \hat{Q} \) by \( f^{-1}(\hat{Q}) \) and then deleting—consecutive—duplicate sets. We claim that \( \sigma' \) is a reconfiguration sequence from \( S \) to \( T \) in \( R_k(G) \). However, \( |\sigma'| \leq \hat{\sigma} < |\sigma| \), contradicting our assumption that \( \sigma \) is a shortest reconfiguration sequence from \( S \) to \( T \) in \( R_k(G) \). Therefore, \( dist_{R_k(G)}(S, T) \leq dist_{R_\mu(H)}(f(S), f(T)) \leq (2n+1) \cdot dist_{\mu(H)}(S, T) \) (where the second inequality follows from our construction of \( \hat{\sigma} \) and the fact that \( f(v) \leq 2n+1 \), for all \( v \in V(G) \)). To show that \( \sigma' \) is a reconfiguration sequence from \( S \) to \( T \), we invoke Corollary 3.8, i.e., for every \( \hat{Q} \in \hat{\sigma} \), we have \( |f^{-1}(\hat{Q})| \leq k \) and, therefore, \( f^{-1}(\hat{Q}) \in V(R_k(G)) \). Note that, after deleting consecutive duplicate sets to obtain \( \sigma' \), the size of the symmetric difference of consecutive sets is again exactly one.

Lemma 3.10. For all \( \hat{S}, \hat{T} \in V(R_\mu(H)) \), if \( dist_{R_\mu(H)}(\hat{S}, \hat{T}) \) is finite, then \( dist_{R_k(G)}(f^{-1}(\hat{S}), f^{-1}(\hat{T})) \) is finite and \( dist_{R_k(G)}(f^{-1}(\hat{S}), f^{-1}(\hat{T})) \leq (2n+1) \cdot dist_{R_\mu(H)}(f^{-1}(\hat{S}), f^{-1}(\hat{T})) + O(n^2) \).

Proof. Let \( \hat{\sigma} = (\hat{Q}_0 = \hat{S}, \hat{Q}_1, \ldots, \hat{Q}_{t-1}, \hat{Q}_t = \hat{T}) \) denote a shortest reconfiguration sequence from \( \hat{S} \) to \( \hat{T} \) in \( R_\mu(H) \). By Corollary 3.8, we know that \( |f^{-1}(\hat{Q}_i)| \leq k \) and, therefore, \( f^{-1}(\hat{Q}_i) \in V(R_k(G)) \) for all \( i \). Consider the sequence \( \sigma \) obtained from \( \hat{\sigma} \) by replacing every set \( \hat{Q} \in \hat{\sigma} \) by \( f^{-1}(\hat{Q}) \) and then deleting—consecutive—duplicate sets. Using the same arguments as in the proof of Lemma 3.9, we know that \( \sigma \) is a reconfiguration sequence from \( f^{-1}(\hat{S}) \) to \( f^{-1}(\hat{T}) \) in \( R_k(G) \). Moreover, from Lemma 3.9, we know that

\[
\text{dist}_{R_k(G)}(f^{-1}(\hat{S}), f^{-1}(\hat{T})) \leq \text{dist}_{R_\mu(H)}(f(f^{-1}(\hat{S})), f(f^{-1}(\hat{T})))
\]

\[
\leq (2n+1) \cdot \text{dist}_{R_\mu(H)}(f^{-1}(\hat{S}), f^{-1}(\hat{T})).
\]

In addition, we know that

\[
\text{dist}_{R_\mu(H)}(\hat{S}, \hat{T}) \leq \text{dist}_{R_\mu(H)}(\hat{S}, f(f^{-1}(\hat{S}))) + \text{dist}_{R_\mu(H)}(f(f^{-1}(\hat{S})), f(f^{-1}(\hat{T}))) + \text{dist}_{R_\mu(H)}(f(f^{-1}(\hat{T})), \hat{T}).
\]

Combining inequalities (1) and (2) with the fact that \( \text{dist}_{R_\mu(H)}(\hat{Q}, f(f^{-1}(\hat{Q}))) \) is at most \( |V(H)| \) (for any \( \hat{Q} \in R_\mu(H) \)), we get the claimed bound on \( \text{dist}_{R_k(G)}(\hat{S}, \hat{T}) \).

The next corollary follows from Lemma 3.9 and 3.10 above.

Corollary 3.11. For all \( S, T \in V(R_k(G)) \), \( S \) and \( T \) belong to the same connected component of \( R_k(G) \) if and only if \( f(S) \) and \( f(T) \) belong to the same connected component of \( R_\mu(H) \). For all \( \hat{S}, \hat{T} \in V(R_\mu(H)) \), \( \hat{S} \) and \( \hat{T} \) belong to the same connected component of \( R_\mu(H) \) if and only if \( f^{-1}(\hat{S}) \) and \( f^{-1}(\hat{T}) \) belong to the same connected component of \( R_k(G) \).

Lemma 3.12. Let \( C \) denote the connected component of \( R_k(G) \) containing \( S^* \) and let \( \hat{C} \) denote the connected component of \( R_\mu(H) \) containing \( f(S^*) \). Then, \( f(S^*) \) is the unique local minimum of \( \hat{C} \). Moreover, for all \( T \in V(C) \), \( \text{dist}_{R_k(G)}(S^*, T) \leq \text{dist}_{R_\mu(H)}(f(S^*), f(T)) \leq O(n^2) \).

Proof. First, recall that, for \( X \subseteq V(G) \), we have \( |f(X)| = (2n+1)|X| - |S^* \cap X| \). Moreover, for \( \hat{X} \subseteq V(H) \), we have \( |\hat{X}| \geq |f(f^{-1}(\hat{X}))| = (2n+1)|f^{-1}(\hat{X})| - |S^* \cap f^{-1}(\hat{X})| \). Consider \( Q \in V(C) \) and \( \hat{Q} \in \hat{C} \). From Corollary 3.11, we know that
\( f(Q) \in \widehat{C} \) and \( f^{-1} (\widehat{Q}) \in V(C) \). Assume that there exists \( Q^* \in V(\widehat{C}) \) such that \( |Q^*| \leq |f(S^*)| = (2n + 1)|S^*| - |S^*| \). Then,

\[
(2n + 1)|f^{-1}(Q^*)| - |S^* \cap f^{-1}(Q^*)| \leq |Q^*| \leq (2n + 1)|S^*| - |S^*|.
\]

Since both \( S^* \) and \( f^{-1}(Q^*) \) belong to \( C \), and \( S^* \) is a local minimum in \( C \), it must be the case that \( |f^{-1}(Q^*)| \geq |S^*| \). But then

\[
(2n + 1)|S^*| - |S^* \cap f^{-1}(Q^*)| \leq (2n + 1)|f^{-1}(Q^*)| - |S^* \cap f^{-1}(Q^*)| \\
\leq (2n + 1)|S^*| - |S^*|.
\]

Consequently, we have \( |S^* \cap f^{-1}(Q^*)| \geq |S^*| \), which is only possible if \( f^{-1}(Q^*) = S^* \). It follows that \( f(S^*) \) is the unique local minimum of \( \widehat{C} \).

Applying Lemma 3.9, we get \( dist_{R_k(G)}(S^*, T) \leq dist_{R_k(H)}(f(S^*), f(T)) \). Finally, applying Lemma 3.6 at most \( \mu \) times starting from \( f(T) \), we get \( dist_{R_k(H)}(f(S^*), f(T)) \leq \mu \cdot |V(H)| = O(n^4) \), as needed.

Theorem 3.5 follows from combining Lemmas 3.6 and 3.12. To see why, let \( C \) be a connected component of \( R_k(G) \) and let \( S \) and \( T \) be two vertices in \( V(C) \). By repeated applications of Lemma 3.6, we know that—assuming \( S \) is not a local minimum—there exists \( S^* \in V(C) \) such that \( S^* \) is a local minimum and \( dist_{R_k(G)}(S, S^*) \leq nk = O(n^2) \). Then, applying Lemma 3.12, we know that \( dist_{R_k(G)}(S^*, T) = O(n^4) \), implying a reconfiguration sequence from \( S \) to \( T \) of length at most \( O(n^4) \).

**Proof of Lemma 3.6.** We start with a few additional definitions. Given a reconfiguration sequence \( \sigma \) and its corresponding edit sequence \( \eta \), we use \( touch(\eta) \) to denote the set of vertices touched by \( \eta \). Similarly, we use \( add(\eta) \) and \( rem(\eta) \) to denote the sets of added and removed vertices, respectively. We decompose \( \eta \) into two types of (maximal) blocks. That is, we let \( \eta = \eta_1 \cdot \eta_2 \cdot \ldots \cdot \eta_\ell \). When \( add(\eta_i) \subseteq R \) and \( rem(\eta_i) \subseteq L \), we say \( \eta_i \) is a right block (vertices are added from the right side of \( G \) and deleted from the left side). When \( add(\eta_i) \subseteq L \) and \( rem(\eta_i) \subseteq R \), we say \( \eta_i \) is a left block. Note that blocks are monotone, i.e., every vertex is touched at most once in each block and \( \eta \) alternates between blocks of different types. A sequence (or block) is winning whenever \( add(\eta) < rem(\eta) \), it is losing when \( add(\eta) > rem(\eta) \), and it is neutral otherwise. Given two vertex cover \( S \) and \( T \) belonging to the same connected component of \( R_k(G) \), we say a shortest sequence \( \eta \) between them is tight, if among all shortest sequences between \( S \) and \( T \), \( \eta \) minimizes the sum of the sizes of vertex covers in \( \sigma \). Formally, we define a potential function, \( pot(\sigma) = \sum_{Q \in \sigma} |Q| \). We say \( \sigma \) (or \( \eta \)) is tight if \( \sigma \) is shortest and \( pot(\sigma) \) is minimized.

**Lemma 3.13.** Let \( \eta = \eta_1 \cdot \eta_2 \cdot \ldots \cdot \eta_\ell \) be a valid edit sequence starting at \( S \), where each \( \eta_i \) is a (maximal) block. If there exists \( i \) such that \( \eta_i \) is not winning, \( \eta_{i+1} \) is winning, and \( touch(\eta_i) \cap touch(\eta_{i+1}) = \emptyset \), then \( \eta = \eta_1 \cdot \ldots \cdot \eta_i \cdot \eta_{i+1} \cdot \eta_{i+2} \cdot \ldots \cdot \eta_\ell \) is also valid starting at \( S \).

**Proof.** Assume, w.l.o.g., that \( \eta_i \) is a right block and \( \eta_{i+1} \) is a left block. Let \( A_1 = add(\eta_i), D_1 = rem(\eta_i), A_2 = add(\eta_{i+1}), \) and \( D_2 = rem(\eta_{i+1}) \). Since \( touch(\eta_i) \cap touch(\eta_{i+1}) = \emptyset \), we know that \( D_1 \cap A_2 = D_2 \cap A_1 = \emptyset \). Let \( Q' = \eta_1 \cdot \eta_2 \cdot \ldots \cdot \eta_{i-1}(S) \) and \( Q'' = \eta_1 \cdot \eta_2 \cdot \ldots \cdot \eta_{i+1}(S) \). Note that before the start of \( \eta_i \), all vertices in \( D_1 \cup D_2 \) are in the vertex cover \( Q' \), i.e., \( D_1 \cup D_2 \subseteq Q' \) and all vertices in \( A_1 \cup A_2 \) are not, i.e., \( (A_1 \cup A_2) \cap Q' = \emptyset \). Hence, there are no edges between \( A_1 \) and \( A_2 \). Similarly, after \( \eta_{i+1} \), all vertices in \( A_1 \cup A_2 \) are in \( Q'' \) and all vertices in \( D_1 \cup D_2 \) are not, and therefore, there are no edges between \( D_1 \) and \( D_2 \).

Since \( \eta \) is valid and \( \eta_{i+1} \) is a winning block, we know that every set in \( \sigma' \) is of size at most \( k \). If there exists a set \( Q \in \sigma' \) that is not a vertex cover, then some edge \( uv \) is left uncovered. However,
must have one endpoint in $A_1$ and the other in $A_2$ or one endpoint in $D_1$ and the other in $D_2$. But we have just shown that such edges cannot exist.

A crown is an ordered pair $(C, H)$ of subsets of vertices from $G$ that satisfies the following criteria: (1) $C \neq \emptyset$ is an independent set of $G$, (2) $H = N(C)$, and (3) there exists a matching $M$ on the edges connecting $C$ and $H$ such that all elements of $H$ are matched ($M$ saturates $H$). $H$ is called the head of the crown. Note that, by definition, $|C| \geq |H|$. Crowns have proved to be very useful for the design of fixed-parameter tractable and kernelization algorithms for the Vertex Cover problem and many others [1, 9]. Given $G$, two vertex covers $S$ and $T$ of $G$ of size at most $k$, and an edit sequence $\eta$ transforming $S$ to $T$, we say $(C, H)$ is an $\eta$-local crown if $C \cup H \subseteq \text{touch}(\eta)$ ($(C, H)$ is a crown in $G[\text{touch}(\eta)]$). In other words, a local crown ignores all vertices that are never touched by $\eta$. The usefulness of local crowns is captured by the next lemma.

**Lemma 3.14.** Let $S$ and $T$ be two vertex covers of $G$ of size at most $k$ and let $\eta$ be an edit sequence transforming $S$ to $T$. Moreover, let $(C, H)$ be an $\eta$-local crown, i.e., $C \cup H \subseteq \text{touch}(\eta)$. If $H \subseteq S, C \cap S = \emptyset, H \subseteq T$, and $C \cap T = \emptyset$, then $\eta$ is not a shortest edit sequence from $S$ to $T$.

**Proof.** Note that, as $M$ saturates $H$, at least $|H|$ vertices are needed to cover edges in $G[C \cup H]$. In other words, for any vertex cover $Q$ of $G$, at least $|H|$ vertices of $Q$ are in $C \cup H$. Moreover, $(Q \setminus (C \cup H)) \cup H$ is a vertex cover of $G$ of size at most $|Q|$. Now consider the edit sequence $\eta'$ obtained from $\eta$ by skipping/deleting all additions and removals of vertices in $C \cup H$. Since $|C| \geq |H|$, $H = N(C)$, and vertices in $H$ are never touched, it follows that $\eta'$ remains valid.

We are now ready to prove our main lemma.

**Lemma 3.15.** Let $G$ be a bipartite graph and $S$ be a (containment-wise) minimal vertex cover of $G$ of size at most $k$. Let $\eta$ be a valid tight winning sequence starting from $S$. Then, $\eta$ consists of one (either left or right) block, i.e., $\eta$ is monotone.

**Proof.** Assume otherwise. Let $\eta = \eta_1 \cdots \eta_{\ell}$ and let $\sigma = \sigma_1 \cdots \sigma_{\ell}$ be the corresponding sequence of vertex covers. Then, there exists at least one non-winning block followed by a winning block, say $\eta_1$ and $\eta_{i+1}$. If $\text{touch}(\eta_1) \cap \text{touch}(\eta_{i+1}) = \emptyset$, then, by Lemma 3.13, we can swap $\eta_1$ and $\eta_{i+1}$ to obtain sequences $\eta' = \eta_1 \cdots \eta_{i+1} \cdot \eta_i \cdots \eta_{\ell}$ and $\sigma' = \sigma_1 \cdots \sigma_{i+1} \cdots \sigma_i \cdots \sigma_{\ell}$, which are also valid. However, as $\eta_{i+1}$ is winning and $\eta_1$ is not, $\eta_1 \cdots \eta_{i-1} \cdot \eta_{i+1} \cdot \eta_i \cdots \eta_{\ell}$ and, therefore, $\sum_{Q' \subseteq Q} |Q'| < \sum_{Q' \subseteq Q} |Q|$ (the size of all other vertex cover does not increase). Hence, $\text{pot}(\sigma') = \sum_{Q' \subseteq Q} |Q'| < \text{pot}(\sigma) = \sum_{Q' \subseteq Q} |Q|$, contradicting the fact that $\eta$ is tight.

From now on, we assume, w.l.o.g., that $\text{touch}(\eta_1) \cap \text{touch}(\eta_{i+1}) \neq \emptyset$, $\eta_1$ is a right block, and $\eta_{i+1}$ is a left block. Let $A_1 = \text{add}(\eta_1), D_1 = \text{rem}(\eta_1), A_2 = \text{add}(\eta_{i+1}),$ and $D_2 = \text{rem}(\eta_{i+1})$. Let $Q^\alpha = \eta_1 \cdot \eta_2 \cdots \cdot \eta_{i-1}(S), Q^\beta = \eta_1 \cdot \eta_2 \cdots \cdot \eta_{i}(S),$ and $Q^\gamma = \eta_1 \cdot \eta_2 \cdots \cdot \eta_{i+1}(S)$ (see Figure 2).

**Claim 1.** There are no edges between $A_1 \cap D_2$ and $D_1 \setminus A_2$, or $A_1 \cap D_2$ and $A_2 \setminus D_1$. Similarly, there are no edges between $D_2 \setminus A_1$ and $D_2 \setminus A_2$ and $A_2 \setminus D_1$ and $A_1 \setminus D_2$.

**Proof.** The fact that $((A_1 \setminus D_2) \cup (A_2 \setminus D_1)) \cap Q^\alpha = \emptyset$ and $((D_1 \setminus A_2) \cup (D_2 \setminus A_1)) \cap Q^\gamma = \emptyset$ implies that there are no edges between $A_1 \setminus D_2$ and $A_2 \setminus D_1$ or between $D_1 \setminus A_2$ and $D_2 \setminus A_1$. As $(A_1 \cap D_2) \cap Q^\alpha = \emptyset$ and $(A_2 \cap D_1) \cap Q^\alpha = \emptyset$, there can be no edges between $A_1 \cap D_2$ and $A_2 \setminus D_1$. Similarly, as $(A_2 \cap D_2) \cap Q^\beta = \emptyset$ and $(D_1 \cap A_2) \cap Q^\beta = \emptyset$, there can be no edges between $A_2 \cap D_2$ and $D_1 \setminus A_2$ (Figure 2).

Now consider the sequence $\eta' = \eta_1 \cdots \cdot \eta_{i+1} \cdot \eta_i \cdots \cdot \eta_{\ell}$ obtained from $\eta$ by swapping $\eta_1$ and $\eta_{i+1}$ and deleting all additions and removals of vertices in $(A_1 \cap D_2) \cup (A_2 \cap D_1)$. Let $\sigma' = \sigma_1 \cdots \cdot \sigma_{i+1} \cdot \sigma_i \cdots \cdot \sigma_{\ell}$ be the corresponding sequence of vertex covers.
Fig. 2. (Parts of) vertex covers $Q^x$ (left), $Q^y$ (middle), and $Q^y$ (right) shown in black (forbidden edges drawn as dotted lines).

Fig. 3. $|Q'| = |Q| + |D_1| - |A_1| + |R_{out}| - |L_{in}|$ (left, forbidden edges shown as dotted lines). $|Q'| = |Q| + |A_2| - |D_2| + |R'_{out}| - |L'_{in}|$ (right).

**Claim 2.** Every set $Q'$ in $\sigma_{i+1} \cdot \sigma_i'$ is a vertex cover of $G$.

**Proof.** By Claim 1, we know that $N(A_1 \cap D_2) \subseteq A_2 \cap D_1$. Since vertices of $A_2 \cap D_1 \subseteq Q^x$ are never removed, all edges with one endpoint in $A_2 \cap D_1$ are covered. Again, by Claim 1, for each remaining edge $uv$, we either have $u \in A_1 \setminus D_2$ and $v \in D_1 \setminus A_2$ or $u \in A_2 \setminus D_1$ and $v \in D_2 \setminus A_1$. As the original sequence is valid, we know that such edges are covered. \hfill $\square$

**Claim 3.** Every set $Q'$ in $\sigma_{i+1}'$ can be associated with a set $Q$ in $\sigma_{i+1}$ such that $|Q'| = |Q| + |D_1| - |A_1| + |R_{out}| - |L_{in}|$, where $R_{out} \subseteq A_1 \cap D_2$ and $L_{in} \subseteq A_2 \cap D_1$. Similarly, every set $Q'$ in $\sigma_i'$ can be associated with a set $Q$ in $\sigma_i$ such that $|Q'| = |Q| + |A_2| - |D_2| + |R'_{out}| - |L'_{in}|$, where $R'_{out} \subseteq A_1 \cap D_2$ and $L'_{in} \subseteq A_2 \cap D_1$.

**Proof.** Let $Q'$ be a set in $\sigma_{i+1}'$. We associate $Q'$ with $Q$ in $\sigma_{i+1}$ such that $Q \cap ((A_2 \setminus D_1) \cup (D_2 \setminus A_1)) = Q' \cap ((A_2 \setminus D_1) \cup (D_2 \setminus A_1))$. Since we only delete additions and removals of vertices in $\sigma_{i+1}'$, such a set $Q$ must exist. Given that $\eta_i$ and $\eta_{i+1}$ are swapped in $\eta'$, it follows that $|Q'| = |Q| + |D_1| - |A_1| + |R_{out}| - |L_{in}|$; vertices in $D_1$ are not removed, vertices in $A_1$ are not yet added, some set of vertices $R_{out} \subseteq A_1 \cap D_2$ is possibly added, and some set of vertices $L_{in} \subseteq A_2 \cap D_1$ is possibly removed (Figure 3).

Let $Q'$ be a set in $\sigma_i'$. We associate $Q'$ with $Q$ in $\sigma_i$ such that $Q \cap ((A_1 \setminus D_2) \cup (D_1 \setminus A_2)) = Q' \cap ((A_1 \setminus D_2) \cup (D_1 \setminus A_2))$. It follows that $|Q'| = |Q| + |A_2| - |D_2| + |R'_{out}| - |L'_{in}|$; vertices in $A_2$ are already added, vertices in $D_2$ are removed, some set of vertices $R'_{out} \subseteq A_1 \cap D_2$ is possibly added, and some set of vertices $L'_{in} \subseteq A_2 \cap D_1$ is possibly removed (Figure 3). \hfill $\square$
Recall that $\eta_1$ is a non-winning block and $\eta_{i+1}$ is a winning block. Therefore, $|A_1| - |D_1| \geq 0$ (or $|D_1| - |A_1| \leq 0$) and $|A_2| - |D_2| \leq 0$. Consequently, if $|R_{out}| - |L_{in}| \leq 0$ and $|R'_{out}| - |L'_{in}| \leq 0$ for all sets in $\sigma_{i+1}^{t} \cdot \sigma_{i}^{t}$, then $\eta'$ is a shorter sequence from $S$ to $T$, again contradicting our choice of $\eta$. So there exists at least one set such that either $|R_{out}| - |L_{in}| > 0$ (or $|R'_{out}| - |L'_{in}| > 0$). But then there exists a pair $(C = R_{out}, H = L_{in})$ such that $|C| > |H|$, $N(C) \subseteq H$ (by Claim 1), $H \subseteq A_2 \cap D_1 \subseteq \sigma^{t} \cap \sigma^{t} = \emptyset$, $H \subseteq Q^{t}$, and $C \cap Q^{t} = \emptyset$. We claim that there exists $(C \subseteq C', H' \subseteq H)$ such that $(C', H')$ is an $(\eta_1 \cdot \eta_{i+1})$-local crown. Applying Lemma 3.14 to the local crown completes the proof, as it implies that $\eta_1 \cdot \eta_{i+1}$ is not the shortest possible sequence from $Q^{t}$ to $Q^{t}$.

We conclude the proof by constructing $(C', H')$. First, we set $C' = C$ and $H' = H$. Next, we delete all vertices in $H'$ with no neighbors in $C'$ so that $N(C') = H'$ (note that $|C'| > |H'|$ still holds). The remaining condition for $(C', H')$ to satisfy is for $G[C' \cup H']$ to have a matching, which saturates $H'$. Hall’s Marriage Theorem [16] states that such a saturating matching exists if and only if for every subset $W$ of $H'$, $|W| \leq |N_{G[C' \cup H']}(W)|$. By a simple application of Hall’s theorem, if no saturating matching exists, then there exists a subgraph $Z$ of $G[C' \cup H']$ such that $|V(Z) \cap C'| < |V(Z) \cap H'|$. By repeatedly deleting such subgraphs, we eventually reach a pair $(C', H')$ that satisfies all the required properties. Since $|C| > |H|$, such a pair is guaranteed to exist as otherwise every subset $W$ of $H$ would satisfy $|W| > |N_{G[C \cup H]}(W)|$ and, hence, $|H| > |C|$, a contradiction.

## 4 PSPACE-COMPLETENESS UNDER THE TOKEN SLIDING MODEL

We now switch from the “vertex cover view” to the “independent set view” of the problem. Given a graph $G$ and an integer $k$, the reconfiguration graph $S_k(G)$ is the graph whose nodes are independent sets of $G$ of size exactly $k$. Two independent sets $I$ and $J$ are adjacent in $S_k(G)$ if we can transform one into the other by sliding a token along an edge. More formally, $I$ and $J$ are adjacent in $S_k(G)$ if $J \setminus I = \{u\}$, $I \setminus J = \{v\}$, and $uv$ is an edge of $G$. The Token Sliding problems ask, given a graph $G$ and two independent sets $I$ and $J$ of size $k$, whether $dist_{S_k(G)}(I, J)$ is finite.

We show that Token Sliding is PSPACE-complete by a reduction from a variant of the Word Reconfiguration problem. Given a pair $W = (\Sigma, A)$, where $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_{|\Sigma|}\}$ is a set of symbols and $A = \{(\sigma_x, \sigma_x'), \ldots, (\sigma_y, \sigma_y')\} \subseteq \Sigma \times \Sigma$ is a binary relation between symbols, a string over $\Sigma$ is a word if every two consecutive symbols are in the relation $A$. If one looks at $W$ as a directed graph (possibly with loops), a string is a word if and only if it is a walk in $W$. The PSPACE-complete Word Reconfiguration problem [36] asks whether two given words, $w_u$ and $w_t$, of equal length $n$ can be transformed into one another by changing one symbol at a time so that all intermediary strings are also words. A crucial observation about the Word Reconfiguration problem is that one can actually treat even/odd positions in words independently. Formally, we introduce the Even/Odd Word Reconfiguration problem, where instead of changing symbols one at a time, we allow bulk changes as long as they occur at only even or odd positions. The next proposition follows from the observation above and the PSPACE-completeness of the Word Reconfiguration problem. In other words, one can easily reduce the Word Reconfiguration problem to the Even/Odd Word Reconfiguration problem since any bulk changes of symbols at either even or odd positions can be replaced by a sequence of single symbol changes (and any valid solution to the former problem is a valid solution to the latter).

**Proposition 4.1.** Even/Odd Word Reconfiguration is PSPACE-complete.

Let $\{f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8\}$ and $\{h_1, h_2\}$ be two sets of vertices. We let $H$ be the graph depicted in Figure 4. Formally, we have $V(H) = \{f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8\}$ and $E(H)$ consists of $h_1 f_1, h_1 f_2, h_1 f_3, h_1 f_4, h_2 f_5, h_2 f_6, h_2 f_7$, and three internally vertex-disjoint paths between $f_1$ and $f_4$, namely $P_1 = \{f_1, f_2, f_3, f_4\}$, $P_2 = \{f_1, f_5, f_6, f_7\}$, and $P_3 = \{f_1, f_8, f_9, f_10\}$. We call $F = H[V(H) \setminus \{h_1, h_2\}]$ a cage (graph) and we call $H$ an extended cage (graph). In the remainder of this section,
when we say “add a cage between two vertices $u$ and $v$,” we mean creating a new extended cage and identifying $u$ with $h_1$ and $v$ with $h_2$.

**Proposition 4.2.** Let $B$ be a bipartite graph with bipartition $(L, R)$. Let $B'$ be the graph obtained from $B$ by adding a cage between $u$ and $v$, for all $u \in L$ and $v \in R$. Then, $B'$ is bipartite.

**Proof.** Consider the following bipartition of $V(B')$. Let $L' = L \cup \{f_2, f_4, f_5, f_7\}$ and let $R' = R \cup \{f_1, f_3, f_6, f_8\}$. Since there are no edges between vertices in $L'$ and no edges between vertices in $R'$, the proposition follows. \( \square \)

Given an instance $(W = (\Sigma, A), w_s, w_t)$ of the Word Reconfiguration problem (where $|w_s| = |w_t| = n$), we assume, without loss of generality, that $n$ is even. We construct a graph $G$ as follows. Let $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_{|\Sigma|}\}$. We first create $n$ sets of vertices, $X_1, X_2, \ldots, X_n$, each of size $|\Sigma|$. We set $X_i = \{x_{i1}, x_{i2}, \ldots, x_{i|\Sigma|}\}$. For each $X_i, i \in [n]$ and $i$ even, we create a vertex $e_i$ and add all edges between $e_i$ and vertices in $X_i$. For each $X_i, i \in [n]$ and $i$ odd, we create a vertex $e_i'$ and add all edges between $e_i'$ and vertices in $X_i$. We let $U_{\text{odd}} = \{o_i \mid i \text{ is odd}\}$ and $U_{\text{even}} = \{e_i \mid i \text{ is even}\}$. For all $1 \leq i < n$, $1 \leq j, j' \leq |\Sigma|$, $x_{ij} \in X_i$, and $x_{jj'+1} \in X_{j+1}$, we add a cage between $x_{ij}$ and $x_{jj'+1}$ in $G$ if and only if $(\sigma_j, \sigma_{j'}) \notin A$. That is, we add a cage between $x_{ij}$ and $x_{jj'+1}$ whenever $\sigma_j$ and $\sigma_{j'}$ cannot appear consecutively in a word. Finally, we add two vertices $z_1$ and $z_2$ connected by an edge and we add a cage between $z_1$ and each vertex in $U_{\text{odd}}$ and a cage between $z_2$ and each vertex in $U_{\text{even}}$. We call the (extended) cage between vertices $u$ and $v$ a $(u, v)$-(extended) cage. This concludes the construction of the graph $G$ (see Figure 5). Note that $|V(G)| = (|\Sigma| + 1)n + 2 + 8[m(n-1) + n]$, where $m = |\Sigma^2 \setminus A|$. \( \square \)

**Lemma 4.3.** The graph $G$ is bipartite.

**Proof.** Let $L = X_1 \cup X_3 \cup \ldots \cup X_{n-1} \cup U_{\text{even}} \cup \{z_1\}$ and $R = X_2 \cup X_4 \cup \ldots \cup X_n \cup U_{\text{odd}} \cup \{z_2\}$. First, note that both graphs $G[L]$ and $G[R]$ are edgeless, and, therefore, $G[L \cup R]$ is bipartite. Moreover, for any two vertices $u \in L$ and $v \in R$, either $uv \notin E(G)$, $uv \notin E(G)$, or $u$ and $b$ are connected by a distinct cage. Applying Proposition 4.2, we know that $G$ must also be bipartite. \( \square \)

We now describe independent sets of $G$ and distinguish between different types of tokens to help simplify the presentation. We will always have independent sets of size exactly $k = n + 1 + 3[m(n-1) + n]$. We say an independent set $I$ of $G$ is well-formed if it has size $k$ and we can label its tokens as follows. One token, labeled the switch token, is either on vertex $z_1$ or vertex $z_2$. For each $i \in [n]$, there is a token, labeled the $\Sigma_i$-token, on some vertex in $X_i \cup \{e_i\}$ if $i$ is even and on some vertex in $X_i \cup \{e_i'\}$ if $i$ is odd. We call the collection of all such tokens the $\Sigma$-tokens (there are exactly 0.

---

ACM Transactions on Algorithms, Vol. 15, No. 1, Article 7. Publication date: October 2018.
n Σ-tokens). For each (u, v)-extended cage H, there are three tokens, labeled (u, v)-caged tokens, on vertices in V(H) (which includes u and v). The total number of caged tokens is 3[m(n − 1) + n].

We say I is strictly well-formed if it is well-formed and each Σi-token is on some vertex in Xi (not on vertex e1 or o1).

Lemma 4.4. If I is a well-formed independent set, then there exists no (u, v)-cage in G such that \{u, v\} ⊂ I.

Proof. Assume otherwise and let H be such an extended cage (including u and v). Since I is well-formed, we know that |I ∩ V(H)| ≥ 3. However, all vertices of H are at distance at most one from either u or v. Hence, if |I ∩ V(H)| ≥ 3, it cannot be the case that \{u, v\} ⊂ I.

Lemma 4.5. Let I be a strictly well-formed independent set and let x1, x2, ..., and xn be the set of Σ-tokens. Then, wI = σ1σ2...σn is a word.

Proof. It suffices to show that for any two consecutive symbols σj and σj′ in wI, we have (σj, σj′) ∈ A. Assume otherwise. Then, by our construction of G, there exists an (xj, xj′)-cage and \{xj, xj′+1\} ⊂ I. Since I is well-formed, we can apply Lemma 4.4 and get a contradiction.

Given a word w and the graph G, we associate w with a strictly well-formed independent set Iw as follows. We add the switch token on vertex z1 (or z2). For each i ∈ [n], we add a Σi-token on vertex xi whenever w[i] = σi, where 1 ≤ j ≤ |Σ| and w[i] denotes the ith symbol of w. For each (u, v)-cage, u, v ∈ V(G), we add three tokens on vertices f2, f5, and f7 if there is no token on v and we add three tokens on f3, f6, and f8, otherwise.

Lemma 4.6. If (W = (Σ, A), w1, w2) is a yes-instance, then (G, Iw1, Iw2, k) is a yes-instance.
Proof. Let \( \langle w_1 = w_3, w_2, \ldots, w_{\ell-1}, w_{\ell} = w_1 \rangle \) denote the corresponding sequence of words. Let \( I_w \) and \( I_{w'} \) be (arbitrarily chosen) strictly well-formed independent sets corresponding to two consecutive words in this sequence. It is enough to show that \( \text{dist}_{S_k(G)}(I_w, I_{w'}) \) is finite. Recall that every two consecutive words differ in either odd or even positions. Therefore, to get from \( I_w \) to \( I_{w'} \), the following sequence of token slides will do. Assume that \( w \) and \( w' \) differ in odd positions.

First, for each \((e, z_2)\)-cage, we slide the token on \( f_3 \) to \( f_2 \), the token on \( f_5 \) to \( f_3 \), and the token on \( f_8 \) to \( f_7 \) (if needed). Now, we can slide the switch token from \( z_1 \) to \( z_2 \). Then, for each \((o, z_1)\)-cage, we slide the token on \( f_3 \) to \( f_2 \), the token on \( f_4 \) to \( f_3 \), and the token on \( f_8 \) to \( f_7 \) (if needed). Then, we slide each \( \Sigma_i \)-token, for \( i \) odd, to vertex \( o^i \). For each \((x_j^i, x_j^{i+1})\)-cage, where \( i \) is odd and \( w'[i] = \sigma_j \) (and, hence, there is no token on \( x_j^{i+1} \)), we slide the token on \( f_3 \) to \( f_2 \), the token on \( f_6 \) to \( f_5 \), and the token on \( f_8 \) to \( f_7 \) (if needed). For each \((x_j^i, x_j^{i+1})\)-cage, where \( i \) is odd, \( w'[i] \neq \sigma_j \), and there is no token on \( x_j^{i+1} \), we slide the token on \( f_2 \) to \( f_3 \), the token on \( f_5 \) to \( f_6 \), and the token on \( f_7 \) to \( f_8 \) (if needed). Next, for each odd \( i \in [n] \) and assuming \( w'[i] = \sigma_j \), we slide the token on \( o^i \) to vertex \( x_j^i \). The even case is handled similarly.

Lemma 4.7. Let \( I \) be an independent set of \( G \) and let \( J \) be an independent set obtained from \( I \) by sliding a single token. If \( I \) is well-formed, then so is \( J \).

Proof. We divide the proof into several cases. First, assume that the switch token slides into vertex \( f_4 \) of an \((o, z_1)\)-cage \( H \) (or symmetrically into vertex \( f_1 \) of an \((e, z_2)\)-cage). Before the slide, the token must be on vertex \( z_1 \). If \( I \) is well-formed, then \( |I \cap V(H)| \geq 3 \) and none of those three tokens can be on \( f_2, f_5, \) or \( f_7 \). Hence, at least one token must be on either \( f_3, f_6, \) or \( f_8 \), implying that the switch token cannot slide from \( z_1 \) to \( f_4 \).

Assume that a \( \Sigma_i \)-token, \( i \) even (or, symmetrically, \( i \) odd), slides into an \((e, z_2)\)-cage \((o, z_1)\)-cage \( H \). When \( I \) is well-formed, we know that the switch token is either on \( z_1 \) or \( z_2 \). Before the slide, the \( \Sigma_i \)-token must be on vertex \( e \). If the switch token is on \( z_2 \), then this is not possible (Lemma 4.4). Hence, the switch token must be on \( z_1 \) and the \( \Sigma_i \)-token must slide from vertex \( e \) to vertex \( f_3 \). But, using the same argument as in the previous case, we know that at least one token must be on either \( f_3, f_6, \) or \( f_8 \), implying that the \( \Sigma_i \)-token cannot slide from \( e \) to \( f_4 \). The same arguments also hold for the case where a \( \Sigma_i \)-token slides into an \((x_j^i, x_j^{i+1})\)-cage.

Assume that a \((x_j^i, x_j^{i+1})\)-caged token, \( i \) odd, slides out of its extended cage, i.e., either slides to vertex \( o^i \in U_{\text{odd}} \) or vertex \( e^{i+1} \in U_{\text{even}} \). Since \( I \) is well-formed, we know that, for each odd \( i, X_i \cup \{o^i\} \) contains one \( \Sigma_i \)-token. In addition, for each even \( i, X_i \cup \{e^i\} \) contains one \( \Sigma_i \)-token. Since \( o^i \) is connected to all vertices in \( X_i \), and \( e^{i+1} \) is connected to all vertices in \( X_{i+1} \), no \((x_j^i, x_j^{i+1})\)-caged token can slide to \( o^i \) nor \( e^{i+1} \).

Finally, assume that an \((e, z_2)\)-caged token or an \((o, z_1)\)-caged token slides out of its extended cage. Since the switch token must be on \( z_1 \) or \( z_2 \), it must be the case that the caged token slides from some vertex \( o^i \in U_{\text{odd}} \) or some vertex \( e^i \in U_{\text{even}} \) to its neighbor in \( X_i \). But this contradicts the fact that the \( \Sigma_i \)-token must be on some vertices in \( X_i \).

Lemma 4.8. Let \( \langle I_1, I_2, \ldots, I_{\ell-1}, I_\ell \rangle \) denote a reconfiguration sequence between two strictly well-formed independent sets in \( S_k(G) \). Let \( I_p \) and \( I_q \) be two sets in this sequence such that the following holds:

\[-1 \leq p < q \leq \ell;\]
\[-I_p \text{ is obtained from } I_{p-1} \text{ by sliding the switch token from } z_1 \text{ to } z_2 \] (or vice-versa);
\[-I_q \text{ is obtained from } I_{q-1} \text{ by sliding the switch token from } z_1 \text{ to } z_2 \] (or vice-versa);
\[-There exists no set } I_r \text{ such that } p \leq r \leq q \text{ and } I_r \text{ is obtained from } I_{r-1} \text{ by sliding the switch token}.\]
Then, $I_p$ and $I_q$ are strictly well-formed. Moreover, either $w_{I_p}[i] = w_{I_q}[i]$, for all even $i$, or $w_{I_p}[i] = w_{I_q}[i]$, for all odd $i$.

**Proof.** Since $I_1$ is well-formed, we can assume, by Lemma 4.7, that all sets in the sequence are well-formed. To show that $I_p$ and $I_q$ are strictly well-formed, we will, in fact, show that any independent set $I$ (in the sequence) obtained from its predecessor $I'$ by sliding the switch token must be strictly well-formed. Since $I$ is well-formed, it is enough to show that there are no $\Sigma$-tokens from $I$ on $U_{\text{odd}} \cup U_{\text{even}}$. Assume otherwise, and consider the case where the switch token slides from $z_1$ to $z_2$ (the other case is symmetric). Applying Lemma 4.4, we know that in $I$ there can be no tokens on vertices in $U_{\text{odd}}$. Similarly, and since the switch token is the only token to slide, we know that in $I'$ there can be no tokens on vertices in $U_{\text{even}}$. Therefore, we can conclude that $I$, $I_p$, and $I_q$ are strictly well-formed. Moreover, by Lemma 4.5, we know that $w_1$, $w_{I_p}$, and $w_{I_q}$ are words.

Now assume that $I_p$ is obtained from $I_{p-1}$ by sliding the switch token from $z_1$ to $z_2$ and $I_q$ is obtained from $I_{q-1}$ by sliding the switch token from $z_2$ to $z_1$. Then, for all $I_{p'}$, $p < p' < q$, $I_{p'} \cap U_{\text{even}} = \emptyset$. In other words, all $\Sigma_i$-tokens, $i$ even, cannot slide to vertex $e^i$ (Lemma 4.4). Hence, $w_{I_p}[i] = w_{I_q}[i]$, for all even $i$. Considering the symmetric case where the switch token starts by sliding from $z_2$ to $z_1$, we get $w_{I_p}[i] = w_{I_q}[i]$, for all odd $i$. 

**Lemma 4.9.** If $(G, I_{w_s}, I_{w_t}, k)$ is a yes-instance, then $(W = (\Sigma, A), w_s, w_t)$ is a yes-instance.

**Proof.** Let $\eta = \langle I_1 = I_{w_s}, I_2, \ldots, I_{\ell-1}, I_\ell = I_{w_t} \rangle$ denote a reconfiguration sequence from $I_{w_s}$ to $I_{w_t}$ in $S_k(G)$. Since $I_{w_s}$ and $I_{w_t}$ are (strictly) well-formed (by construction), we can, from Lemma 4.7, assume that each set in the sequence is well-formed. Note that, because of Lemma 4.7, we need not consider sets that are not well-formed since $\eta$ starts and ends with two well-formed sets.

Consider the subsequence $\eta' = \langle I_{p_1}, \ldots, I_{p_\ell} \rangle$, which is obtained by restricting $\eta$ to only those sets that are obtained from their predecessor by sliding the switch token. In other words, $I_{p_j}$ is obtained from $I_{p_{j-1}}$ by sliding the switch token. We claim that $\langle w_s, w_{I_{p_1}}, \ldots, w_{I_{p_\ell}}, w_t \rangle$ is the required solution to the instance $(W = (\Sigma, A), w_s, w_t)$. Applying Lemma 4.8, we know that each independent set in $\eta'$ is strictly well-formed and, therefore, corresponds to a word (Lemma 4.5). Moreover, from Lemma 4.8, we know that each two consecutive words differ in either odd or even positions only, as needed. Note that $\langle w_s, w_{I_{p_1}}, \ldots, w_{I_{p_\ell}}, w_t \rangle$ might contain duplicate words that can safely be deleted. Moreover, if $\eta'$ is empty, then $w_s$ and $w_t$ differ in only odd or even positions. 

**Theorem 4.10.** Token Sliding is PSPACE-complete on bipartite graphs.

**REFERENCES**

[1] Faisal N. Abu-Khzam, Michael R. Fellows, Michael A. Langston, and W. Henry Suters. 2007. Crown structures for vertex cover kernelization. *Theory Comput. Syst.* 41, 3 (2007), 411–430. DOI: https://doi.org/10.1007/s00224-007-1328-0

[2] Stefan Arnborg, Derek G. Corneil, and Andrzej Proskurowski. 1987. Complexity of finding embeddings in a $K$-tree. *SIAM J. Algebraic Discrete Methods* 8, 2 (April 1987), 277–284. DOI: https://doi.org/10.1137/0608024

[3] Hans L. Bodlaender. 1993. A tourist guide through treewidth. *Acta Cybern.* 11, 1–2 (1993), 1–21.

[4] Hans L. Bodlaender and Dimitrios M. Thilikos. 1997. Treewidth for graphs with small chordality. *Discrete Appl. Math.* 79, 1–3 (1997), 45–61. DOI: https://doi.org/10.1016/S0166-218X(97)00031-0

[5] Marthe Bonamy and Nicolas Bousquet. 2016. Token sliding on chordal graphs. *CoRR* abs/1605.00442 (2016). DOI: http://arxiv.org/abs/1605.00442

[6] Paul S. Bonsma, Marcin Kaminski, and Marcin Wrochna. 2014. Reconfiguring independent sets in claw-free graphs. In *Proceedings of the 14th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT’14)*, 86–97.

[7] Richard C. Brewster, Sean McGuinness, Benjamin Moore, and Jonathan A. Noel. 2016. A dichotomy theorem for circular colouring reconfiguration. *Theor. Comput. Sci.* 639 (2016), 1–13.
8. Luis Cerdeca, Jan van den Heuvel, and Matthew Johnson. 2008. Connectedness of the graph of vertex-colourings. *Discrete Mathematics* 308, 56 (2008), 913–919.

9. Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Damiel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. 2015. *Parameterized Algorithms*. Springer. DOI: https://doi.org/10.1007/978-3-319-21275-3

10. Erik D. Demaine, Martin L. Demaine, Eli Fox- Epstein, Duc A. Hoang, Takehiro Ito, Hirokata Ono, Yota Otachi, Ryuhei Uehara, and Takeshi Yamada. 2014. Polynomial-time algorithm for sliding tokens on trees. In *Proceedings of the 25th International Symposium on Algorithms and Computation (ISAAC’14)*. Lecture Notes in Computer Science, Hee-Kap Ahn and Chan-Su Shin (Eds.), Vol. 8889. Springer, 389–400. DOI: https://doi.org/10.1007/978-3-319-13075-0_31

11. Erik D. Demaine and Joseph O’Rourke. 2007. *Geometric Folding Algorithms—Linkages, Origami, Polyhedra*. Cambridge University Press.

12. Eli Fox-Epstein, Duc A. Hoang, Yota Otachi, and Ryuhei Uehara. 2015. Sliding token on cactus. In *Proceedings of the 26th International Symposium on Algorithms and Computation (ISAAC’15)*, 237–247.

13. Sevag Gharibian and Jamie Sikora. 2015. Ground state connectivity of local Hamiltonians. In *Proceedings, Part I, of the 42nd International Colloquium on Automata, Languages, and Programming (ICALP’15)*, 617–628. DOI: https://doi.org/10.1007/978-3-662-47672-7_50

14. Parikshit Gopalan, Phokion G. Kolaitis, Elitza N. Maneva, and Christos H. Papadimitriou. 2009. The connectivity of Boolean satisfiability: Computational and structural dichotomies. *SIAM J. Comput.* 38, 6 (2009), 2330–2355.

15. Michel Habib and Rolf H. Möhring. 1994. Treewidth of cocomparability graphs and a new order-theoretic parameter. *Order* 11, 1 (1994), 47–60. DOI: https://doi.org/10.1007/BF01462229

16. Philip Hall. 1987. On representatives of subsets. In *Classic Papers in Combinatorics*. Birkhäuser Boston, 58–62. DOI: https://doi.org/10.1007/978-0-8176-4842-8_4

17. Robert A. Hearn and Erik D. Demaine. 2005. PSPACE-Completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theor. Comput. Sci.* 343, 1–2 (2005), 72–96. DOI: https://doi.org/10.1016/j.tcs.2005.05.008

18. Duc A. Hoang and Ryuhei Uehara. 2016. Sliding tokens on a cactus. Retrieved from https://hoanganhduc.github.io/events/ISAAC2016/slide.pdf. [ISAAC Presentation slides].

19. Duc A. Hoang and Ryuhei Uehara. 2017. Polynomial-time algorithms for sliding tokens on cactus graphs and block graphs. *CoRR* abs/1705.00429 (2017). DOI: http://arxiv.org/abs/1705.00429

20. Takehiro Ito, Erik D. Demaine, Nicholas J. A. Harvey, Christos H. Papadimitriou, Martha Sideri, Ryuhei Uehara, and Yushi Uno. 2011. On the complexity of reconfiguration problems. *Theor. Comput. Sci.* 412, 12–14 (2011), 1054–1065. DOI: https://doi.org/10.1016/j.tcs.2010.12.005

21. Takehiro Ito, Marcin Kamiński, and Erik D. Demaine. 2012. Reconfiguration of list edge-colorings in a graph. *Discrete Appl. Math.* 160, 15 (2012), 2199–2207.

22. Takehiro Ito, Marcin Kamiński, Hirokata Ono, Akira Suzuki, Ryuhei Uehara, and Katsuhisa Yamanaka. 2014. On the parameterized complexity for token jumping on graphs. In *Proceedings of the 11th Annual Conference on Theory and Applications of Models of Computation, (TAMC’14)*, 341–351.

23. Takehiro Ito, Hiroyuki Nooka, and Xiao Zhou. 2016. Reconfiguration of vertex covers in a graph. *IEICE Transactions 99-D, 3* (2016), 598–606.

24. Wm. Woolsey Johnson and William E. Story. 1879. Notes on the “15” puzzle. *Am. J. Math.* 2, 4 (1879), 397–404.

25. Marcin Kamiński, Paul Medvedev, and Martin Milanič. 2012. Complexity of independent set reconfiguration problems. *Theor. Comput. Sci.* 439 (2012), 9–15.

26. Iyad A. Kanj and Ge Xia. 2015. Flip distance is in FPT time. In *Proceedings of the 32nd International Symposium on Theoretical Aspects of Computer Science (STACS’15)*, 500–512.

27. Graham Kendall, Andrew J. Parkes, and Kristian Spoerer. 2008. A survey of NP-complete puzzles. *ICGA J.* 31, 1 (2008), 13–34. https://dlbpl.org/record/bib/journals/icga/KendallIPS08.

28. Daniel Lokshtanov, Amer E. Mouawad, Fahad Panolan, M. S. Ramanujan, and Saket Saurabh. 2015. Reconfiguration on sparse graphs. In *Proceedings of the 14th International Symposium on Algorithms and Data Structures (WADS’15)*, 506–517.

29. Anna Lubiw and Vinayak Pathak. 2015. Flip distance between two triangulations of a point set is NP-complete. *Comput. Geom.* 49 (2015), 17–23. DOI: https://doi.org/10.1016/j.comgeo.2014.11.001

30. Amer E. Mouawad, Naomi Nishimura, Vinayak Pathak, and Venkatesh Raman. 2015. Shortest reconfiguration paths in the solution space of Boolean formulas. In *Proceedings, Part I, of the 42nd International Colloquium on Automata, Languages, and Programming (ICALP’15)*, 985–996.

31. Amer E. Mouawad, Naomi Nishimura, and Venkatesh Raman. 2014. Vertex cover reconfiguration and beyond. In *Proceedings of 25th International Symposium on Algorithms and Computation (ISAAC’14)*, 452–463.

32. Amer E. Mouawad, Naomi Nishimura, Venkatesh Raman, Narges Simjou, and Akira Suzuki. 2017. On the parameterized complexity of reconfiguration problems. *Algorithmica* 78, 1 (2017), 274–297.

ACM Transactions on Algorithms, Vol. 15, No. 1, Article 7. Publication date: October 2018.
[33] Neil Robertson and Paul D. Seymour. 1984. Graph minors. III. planar tree-width. J. Comb. Theory, Ser. B 36, 1 (1984), 49–64. DOI: https://doi.org/10.1016/0095-8956(84)90013-3

[34] Paul D. Seymour and Robin Thomas. 1993. Graph searching and a min-max theorem for tree-width. J. Comb. Theory, Ser. B 58, 1 (1993), 22–33. DOI: https://doi.org/10.1006/jctb.1993.1027

[35] Jan van den Heuvel. 2013. The complexity of change. Surveys in Combinatorics 2013, 409 (2013), 127–160.

[36] Marcin Wrochna. 2014. Reconfiguration in bounded bandwidth and treedepth. CoRR abs/1405.0847 (2014). DOI: http://arxiv.org/abs/1405.0847

[37] Marcin Wrochna. 2015. Homomorphism reconfiguration via homotopy. In Proceedings of the 32nd International Symposium on Theoretical Aspects of Computer Science (STACS’15). 730–742.

[38] M. Yannakakis. 1981. Computing the minimum fill-in is NP-complete. SIAM Journal on Algebraic Discrete Methods 2, 1 (1981), 77–79. DOI: https://doi.org/10.1137/0602010

Received October 2017; accepted September 2018