On finding short reconfiguration sequences between independent sets

Akanksha Agrawal
Indian Institute of Technology Madras, Chennai, India

Soumita Hait
Indian Institute of Technology, Kharagpur, India

Amer E. Mouawad
American University of Beirut, Lebanon
University of Bremen, Germany

Abstract

Assume we are given a graph G, two independent sets S and T in G of size $k \geq 1$, and a positive integer $\ell \geq 1$. The goal is to decide whether there exists a sequence $\langle I_0, I_1, ..., I_\ell \rangle$ of independent sets such that for all $j \in \{0, \ldots, \ell - 1\}$ the set I_j is an independent set of size k, $I_0 = S$, $I_\ell = T$, and I_{j+1} is obtained from I_j by a predetermined reconfiguration rule. We consider two reconfiguration rules, namely token sliding and token jumping. Intuitively, we view each independent set as a collection of tokens placed on the vertices of the graph. Then, the Token Sliding Optimization (TSO) problem asks whether there exists a sequence of at most ℓ steps that transforms S into T, where at each step we are allowed to slide one token from a vertex to an unoccupied neighboring vertex (while maintaining independence). In the Token Jumping Optimization (TJO) problem, at each step, we are allowed to jump one token from a vertex to any other unoccupied vertex of the graph (as long as we maintain independence). Both TSO and TJO are known to be fixed-parameter tractable when parameterized by ℓ on nowhere dense classes of graphs. In this work, we investigate the boundary of tractability for sparse classes of graphs. We show that both problems are fixed-parameter tractable for parameter $k + \ell + d$ on d-degenerate graphs as well as for parameter $|M| + \ell + \Delta$ on graphs having a modulator M whose deletion leaves a graph of maximum degree Δ. We complement these result by showing that for parameter ℓ alone both problems become W[1]-hard already on 2-degenerate graphs. Our positive result makes use of the notion of independence covering families introduced by Lokshtanov et al. [24]. Finally, we show as a side result that using such families we can obtain a simpler and unified algorithm for the standard Token Jumping Reachability problem (a.k.a. Token Jumping) parameterized by k on both degenerate and nowhere dense classes of graphs.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases Token sliding, token jumping, fixed-parameter tractability, combinatorial reconfiguration, shortest reconfiguration sequence

Funding *Amer E. Mouawad*: Research supported by the Alexander von Humboldt Foundation, by PHC Cedre project 2022 "PLR", and partially supported by URB project "A theory of change through the lens of reconfiguration".

1 Introduction

Given a simple undirected graph G, a set of vertices $I \subseteq V(G)$ is an *independent set* if the vertices of I are pairwise non-adjacent. Finding an independent set of size k, i.e., the Independent Set (IS) problem, is known to be NP-complete [20] and W[1]-complete parameterized by solution size k [11]. We view an independent set as a collection of k tokens placed on the vertices of a graph such that no two tokens are placed on adjacent vertices. This gives rise to two natural adjacency relations between independent sets (or token configurations), also called *reconfiguration steps*. These reconfiguration steps, in turn, give rise to several *combinatorial reconfiguration* problems [31, 28, 7].

In the Token Sliding Reachability (TSR) problem, introduced by Hearn and Demaine [15], two independent sets are adjacent if one can be obtained from the other by removing a token from a vertex u and immediately placing it on another unoccupied vertex v with the requirement that $\{u, v\}$ must be an edge of the graph. The token is said to *slide* from vertex u to vertex v along the edge $\{u, v\}$. Generally speaking, in the Token Sliding Reachability problem, we are given a graph G and two independent sets S and T of size k in G. The goal is to decide whether there exists a sequence of slides (a *reconfiguration sequence*) that transforms S to T. The TSR problem has been extensively studied [4, 5, 10, 13, 17, 19, 23]. It is known that the problem is PSPACE-complete, even on restricted graph classes such as planar graphs of bounded bandwidth (and hence pathwidth) [15, 33, 32], split graphs [3], and bipartite graphs [22]. However, Token Sliding Reachability can be decided in polynomial time on trees [10], interval graphs [4], bipartite permutation and bipartite distance-hereditary graphs [13], and line graphs [16]. In the Token Sliding Optimization (TSO) problem, we are additionally given a parameter ℓ and the goal is to decide if S can be transformed to T in at most ℓ token slides. Very little is known about the optimization variant of the problem other than the hardness results that follow immediately from the reachability variant. In fact, to the best of our knowledge, the only known polynomial-time solvable instances of TSO are those restricted to interval graphs [34, 18] and cographs [19].

In the Token Jumping Reachability (TJR) problem, introduced by Kamiński et al. [19], we drop the restriction that the token should move along an edge of G and instead we allow it to move to any unoccupied vertex of G provided it does not break the independence of the set of tokens. That is, a single reconfiguration step consists of first removing a token on some vertex u and then immediately adding it back on any other unoccupied vertex v, as long as no two tokens become adjacent. The token is said to *jump* from vertex u to vertex v. Token Jumping Reachability is also PSPACE-complete on planar graphs of bounded bandwidth [15, 33, 32]. Lokshtanov and Mouawad [22] showed that, unlike Token Sliding Reachability, which is PSPACE-complete on bipartite graphs, the Token Jumping Reachability problem becomes NP-complete on bipartite graphs. On the positive side, it is "easy" to show that Token Jumping Reachability can be decided in polynomial-time on trees (and even on split/chordal graphs) since we can simply jump tokens to leaves (resp. vertices that only appear in the bag of a leaf in the clique tree) to transform one independent set into another. In the Token Jumping Optimization (TJO) problem, we are additionally given a parameter ℓ and the goal is to decide if S can be transformed to T in at most ℓ token jumps. To the best of our knowledge, the only known polynomial-time solvable instances of TJO are those restricted to chordal graphs and even-hole-free graphs [19, 25].

In this paper we focus on the parameterized complexity of the aforementioned problems with respect to parameters k and ℓ and when restricted to sparse classes of graphs. Given an NP-hard or PSPACE-hard problem, *parameterized complexity* [12] allows us to refine the notion of hardness; does the hardness come from the whole instance or from a small parameter? A problem Π is *FPT (fixed-parameter tractable) parameterized by k* if one can solve it in time $f(k) \cdot poly(n)$, for some computable function f (sometimes called FPT-time). In other words, the combinatorial explosion can be restricted to the parameter k. In the rest of the paper, we mainly consider parameters k (the number of tokens) and ℓ (the number of reconfiguration steps). TSO and TJO are known to be W[1]-hard parameterized by $k + \ell$ on general graphs [7]. TSR and TJR are known to be W[1]-hard parameterized by k on general graphs [23]. When we restrict our attention to sparse classes of graphs, TSO and TJO are known to be fixed-parameter tractable when parameterized by ℓ on nowhere dense classes of graphs [25]. For TJR, the problem becomes fixed-parameter tractable parameterized by k on biclique-free classes of graphs [6]. Finally, for TSR, the problem becomes fixed-parameter tractable parameterized by k on planar graphs, chordal graphs of bounded clique number, and graphs of bounded degree [2]. We refer the reader to the recent survey by Bousquet et al. [7] for more background on the parameterized complexity of these problems.

Given that TSO and TJO are fixed-parameter tractable when parameterized by ℓ on nowhere dense classes of graphs, it is natural to ask whether this result can be extended beyond nowhere dense graphs to biclique-free graphs. Even simpler, can we show that TSO and TJO remain fixed-parameter tractable when parameterized by ℓ on graph of bounded degeneracy? Recall that any degenerate or nowhere dense class of graphs is a biclique-free class, but not vice versa. Motivated by these questions, we show the following:

- Both problems are fixed-parameter tractable for parameter $k + \ell + d$ on d-degenerate graphs;
- Both problems are fixed-parameter tractable for parameter $|N| + k + \ell + d$ on graphs having a modulator N whose deletion leaves a d-degenerate graph (assuming N is given as part of the input); and
- Both problems are fixed-parameter tractable for parameter $|M| + \ell + \Delta$ on graphs having a modulator M whose deletion leaves a graph of maximum degree Δ.
- We complement these result by showing that for parameter ℓ alone both problems become W[1]-hard already on 2-degenerate graphs (recall that both problems are polynomial-time solvable on 1-degenerate graphs, i.e., forests, which completes the picture based on the degeneracy of the graph).

In fact, our hardness reductions construct 2-degenerate graphs which can be partitioned into two sets V_1 and V_2, where V_1 is an independent set and every vertex in V_2 has constant degree in the graph. Hence, our positive result for parameter $|M|+\ell+\Delta$ shows that when $|M|$ is part of our parameter we can drop k and still obtain fixed-parameter tractable algorithms; and when $|M|$ (and k) is not part of the parameter the problem is W[1]-hard.

Most of our positive results make use of the notion of independence covering families introduced by Lokshtanov et al. [24], which we believe could be of independent interest for the reconfiguration of independent sets. Let us start by formally defining such families and the various algorithms for extracting them on different graph classes.

▶ **Definition 1.1** ([24]). *For a graph G and $k \geq 1$, a family of independent sets of G is called an* independence covering family for (G, k), *denoted by $\mathcal{F}(G, k)$, if for any independent set I in G of size at most k, there exists $J \in \mathcal{F}(G, k)$ such that $I \subseteq J$.*

▶ **Theorem 1.2** ([24]). *There is a deterministic algorithm that given a d-degenerate graph G and $k \geq 1$, runs in time $O((kd)^{O(k)} \cdot (n + m) \log n)$, and outputs an independence covering family for (G, k) of size at most $O((kd)^{O(k)} \cdot \log n)$.*

▶ **Theorem 1.3** ([24]). *Let $k, d \in \mathbb{N}$ and G be a graph. Let $S \subseteq V(G)$ such that $G - S$ is d-degenerate. There is a deterministic algorithm that given a G, S, and $k, d \in \mathbb{N}$, runs in time $O(2^{|S|} \cdot (kd)^{O(k)} \cdot 2^{O(kd)} \cdot (n + m) \log n)$, and outputs an independence covering family for (G, k) of size at most $O(2^{|S|} \cdot (kd)^{O(k)} \cdot 2^{O(kd)} \cdot \log n)$.*

▶ **Theorem 1.4** ([24]). *Let G be a graph such that $G \in \mathcal{G}$, where $\mathcal{G}$ is a class of nowhere dense graphs. There is a deterministic algorithm that given $k \geq 1$, runs in time $O(f_{\mathcal{G}}(k) \cdot$*

$(n+m)\log n$), and outputs an independence covering family for (G,k) of size at most $O(g_{\mathcal{G}}(k) \cdot n \log n)$, where $f_{\mathcal{G}}(k)$ and $g_{\mathcal{G}}(k)$ depend on k and the class $\mathcal{G}$ but are independent of the size of the graph.

We use Theorems 1.2 and 1.3 to design fixed-parameter tractable algorithms for parameters $k+\ell+d$ and $|N|+k+\ell+d$, respectively. Our algorithm for parameter $|M|+\ell+\Delta$ is based on the random separation technique [8]. Finally, we show that using independence covering families we can obtain a simpler and unified algorithm for the standard Token Jumping Reachability problem (a.k.a. Token Jumping) parameterized by k on both degenerate and nowhere dense classes of graphs; this is in contrast to the algorithms presented in [23]. To do so, we make use of Theorems 1.2 and 1.4. Note that the major difference between Theorems 1.2 and 1.4 is that in the former we are guaranteed a family of size at most $O((kd)^{O(k)} \cdot \log n)$ while in the latter the family is of size at least $O(g_{\mathcal{G}}(k) \cdot n \log n)$, i.e., we have an extra linear dependence on n. This difference is the reason why our algorithm for parameter $k+\ell+d$ cannot be adapted to work for nowhere dense graphs. The current complexity status of all problems considered in this work is summarized in Table 1.

Table 1 Parameterized complexity status of the reachability and optimization variants of Token Sliding and Token Jumping. Results proved in this paper are shown in bold.

	Degenerate	Nowhere dense	Biclique free
TSR parameterized by k	Open	Open	Open
TSO parameterized by k	Open	Open	Open
TSO parameterized by ℓ	**W[1]-hard**	FPT	**W[1]-hard**
TSO parameterized by $k+\ell$	**FPT**	FPT	Open
TJR parameterized by k	FPT	FPT	FPT
TJO parameterized by k	Open	Open	Open
TJO parameterized by ℓ	**W[1]-hard**	FPT	**W[1]-hard**
TJO parameterized by $k+\ell$	**FPT**	FPT	Open

The rest of the paper is organized as follows. In Section 2 we introduce required background and terminology. In Section 3 we present our main positive results which are the fixed-parameter tractable algorithms for TSO and TJO parameterized by $k+\ell+d$ on d-degenerate graphs and parameterized by $|N|+k+\ell+d$ on graphs having a modulator N whose deletion leaves a d-degenerate graph (assuming N is given as part of the input). Section 4 is devoted to the fixed-parameter tractable algorithm for parameter $|M|+\ell+\Delta$ on graphs having a modulator M whose deletion leaves a graph of maximum degree Δ. We show hardness on 2-degenerate graphs in Section 5 for TSO and in Section 6 for TJO. We conlude in Section 7 where we present a unified algoritm for Token Jumping Reachability on graphs admitting efficiently computable independence covering families of the right size.

2 Preliminaries

Sets and functions. We denote the set of natural numbers (including 0) by $\mathbb{N}$. For $n \in \mathbb{N}$, we use $[n]$ and $[n]_0$ to denote the sets $\{1, 2, \cdots, n\}$ and $\{0, 1, 2, \cdots, n\}$, respectively. For a set X, we denote its power set by $2^X = \{X' \mid X' \subseteq X\}$. For a function $f : X \to Y$ and an element $y \in Y$, $f^{-1}(y)$ denotes the set $\{x \in X \mid f(x) = y\}$. For a non-empty set X, a family $\mathcal{F} \subseteq 2^X$ is a *partition* of X, if i) for each $Y \in \mathcal{F}$, $Y \neq \emptyset$, ii) for distinct $Y, Z \in \mathcal{F}$, we have $Y \cap Z = \emptyset$, and iii) $\cup_{Y \in \mathcal{F}} Y = X$. An observation that we will make use of is the following:

▶ **Proposition 2.1.** *For $0 < k \in \mathbb{N}$ and $0 < c \leq n \in \mathbb{N}$, for some constant c, we have $(\log n)^k \leq n + k^{2k}$.*

Proof. To see why the proposition holds, we distinguish between two cases:

- If $k \leq \log n / \log\log n$ we have $k \log\log n \leq \log n$. Raising both sides to the power of 2, we obtain $(\log n)^k \leq n$.
- If $k > \log n / \log\log n$ then we claim that $\log\log n < k$. Suppose not. Then we must have, for all n, $\log n < k \log\log n \leq (\log\log n)^2$ which is false. Hence, we have $\log n \leq k^2$ and we get $(\log n)^k \leq k^{2k}$.

Combining the two inequalities we get $(\log n)^k \leq n + k^{2k}$. ◀

Graphs and graph classes. Unless otherwise stated, we assume that each graph G is a simple, undirected graph with vertex set $V(G)$ and edge set $E(G)$, where $|V(G)| = n$ and $|E(G)| = m$. The *open neighborhood*, or simply *neighborhood*, of a vertex v is denoted by $N_G(v) = \{u \mid \{u, v\} \in E(G)\}$, the *closed neighborhood* by $N_G[v] = N_G(v) \cup \{v\}$. Similarly, for a set of vertices $S \subseteq V(G)$, we define $N_G(S) = \{v \mid \{u, v\} \in E(G), u \in S, v \notin S\}$ and $N_G[S] = N_G(S) \cup S$. The *degree* of a vertex is $|N_G(v)|$. We drop the subscript G when clear from context. A *subgraph* of G is a graph G' such that $V(G') \subseteq V(G)$ and $E(G') \subseteq E(G)$. The *induced subgraph* of G with respect to $S \subseteq V(G)$ is denoted by $G[S]$; $G[S]$ has vertex set S and edge set $E(G[S]) = \{\{u, v\} \in E(G) \mid u, v \in S\}$.

Contracting an edge $\{u, v\}$ of G results in a new graph H in which the vertices u and v are deleted and replaced by a new vertex w that is adjacent to $(N_G(u) \cup N_G(v)) \setminus \{u, v\}$. If a graph H can be obtained from G by repeatedly contracting edges, H is said to be a *contraction* of G. If H is a subgraph of a contraction of G, then H is said to be a *minor* of G, denoted by $H \preceq_m G$. The class of nowhere dense graphs [26, 27] is a common generalization of proper minor closed classes, classes of graphs with bounded degree, graph classes locally excluding a fixed graph H as a minor and classes of bounded expansion. In order to formally define the class of nowhere dense graphs, we need a few additional definitions.

▶ **Definition 2.2.** *A graph H is an r-*shallow minor *of G, where r is an integer, if there exists a set of disjoint subsets $V_1, \ldots, V_{|H|}$ of $V(G)$ such that*

1. *each graph $G[V_i]$ is connected and has radius at most r, and*
2. *there is a bijection $\psi : V(H) \to \{V_1, \ldots, V_{|H|}\}$ such that for every edge $\{u, v\} \in E(H)$ there is an edge in G with one endpoint in $\psi(u)$ and the second in $\psi(v)$.*

The set of all r-shallow minors of a graph G is denoted by $G\nabla r$. Similarly, the set of all r-shallow minors of all the members of a graph class $\mathcal{C}$ is denoted by $\mathcal{C}\nabla r = \bigcup_{G\in\mathcal{C}}(G\nabla r)$.

let $\omega(G)$ denotes the size of the largest clique in G and $\omega(\mathcal{C}) = sup_{G\in\mathcal{C}}(\omega(G))$.

▶ **Definition 2.3.** *A class of graphs $\mathcal{C}$ is said to be* nowhere dense *if there exists a function $f_\omega : \mathbb{N} \to \mathbb{N}$ such that for all r we have that $\omega(\mathcal{C}\nabla r) \leq f_\omega(r)$.*

Nowhere density turns out to be a very robust concept with several natural characterizations and applications (see, e.g., [21]).

▶ **Definition 2.4.** *A class of graphs $\mathcal{C}$ is said to be d-*degenerate *if every induced subgraph of any graph $G \in \mathcal{C}$ has a vertex of degree at most d. $\mathcal{C}$ is said to be* degenerate *if it is d-degenerate for some d.*

Graphs of bounded degeneracy and nowhere dense graphs are incomparable [14]. In other words, graphs of bounded degeneracy are somewhere dense. Degeneracy is a hereditary property, hence an induced subgraph of a d-degenerate graph is also d-degenerate. It is well-known that graphs of treewidth at most d are also d-degenerate. Moreover a d-degenerate graph cannot contain $K_{d+1,d+1}$ as a subgraph, which brings us to the class of biclique-free graphs. The relationship between bounded degeneracy, nowhere dense, and $K_{d,d}$-free graphs was shown by Philip et al. and Telle and Villanger [29, 30].

▶ **Definition 2.5.** *A class of graphs $\mathcal{C}$ is said to be d-*biclique-free*, for some $d > 0$, if $K_{d,d}$ is not a subgraph of any $G \in \mathcal{C}$. $\mathcal{C}$ is said to be* biclique-free *if it is d-biclique-free for some d.*

▶ **Proposition 2.6.** *Any degenerate or nowhere dense class of graphs is biclique-free, but not vice-versa.*

3 FPT algorithm for parameter $k + \ell + d$

In this section we start by designing a fixed-parameter tractable algorithm for the TSO problem parameterized by $k+\ell+d$ on d-degenerate graphs. We then show how the algorithm can be adapted for TJO as well as for parameter $|N|+k+\ell+d$ on graphs having a modulator N whose deletion leaves a d-degenerate graph (assuming N is given as part of the input).

We let (G, S, T, k, ℓ) denote an instance of TSO, where G is d-degenerate. Moreover, we assume that we have computed in time $O((kd)^{O(k)} \cdot (n + m) \log n)$ an independence covering family $\mathcal{F}(G, k)$ for (G, k) of size at most $O((kd)^{O(k)} \cdot \log n)$ (Theorem 1.2). Without loss of generality, we assume that both S and T belong to $\mathcal{F}(G, k)$; as otherwise we can simply add them. Note that if (G, S, T, k, ℓ) is a yes-instance then there exists a sequence $\langle I_0, I_1, ..., I_\ell\rangle$ of independent sets such that for all $j \in \{0, \ldots, \ell-1\}$ the set I_j is an independent set of size k in G, $I_0 = S$, $I_\ell = T$, and I_{j+1} is obtained from I_j by a token slide. This implies that there exists a sequence $\langle J_0, J_1, ..., J_\ell\rangle$ of elements of $\mathcal{F}(G, k)$ such that $J_0 = S$, $J_\ell = T$, and for $j \in \{1, \ldots, \ell-1\}$ we have $I_j \subseteq J_j$. In what follows, we assume that we guessed a sequence $\langle J_0, J_1, ..., J_\ell\rangle$ of elements of $\mathcal{F}(G, k)$ such that $J_0 = S$ and $J_\ell = T$. Our goal now is to design an algorithm that either finds a reconfiguration sequence $\langle I_0 = S, I_1 \subseteq J_1, ..., I_{\ell-1} \subseteq J_{\ell-1}, I_\ell = T\rangle$ or determine that no such sequence exists.

We define a constraint as a pair (X, b) where $X \subseteq V(G)$ and b is a positive integer, called the budget of X. We denote a set of constraints by $C = \{(X, b), \ldots\}$. We say that the constraint (X, b) is satisfied (by Z) if $|Z \cap X| = b$, where $Z \subseteq V(G)$. We say that the set of constraints C is satisfied if all pairs $(X, b) \in C$ are satisfied. We denote a set of sets of constraints by $\mathcal{C}$. We now proceed by building sets of sets of constraints $\mathcal{C}_0$, $\mathcal{C}_1$, $\ldots$, $\mathcal{C}_\ell$ and show that for each $i \in [\ell]_0$, the following invariants are satisfied:

- **Correctness Invariant I:** If a k-sized set $Z \subseteq J_i$ satisfies at least one set of constraints in $\mathcal{C}_i$, then Z is reachable from $S = J_0$.
- **Correctness Invariant II:** For any k-sized set $Z \subseteq J_i$, if there is a reconfiguration sequence $S = I_0, I_1, I_2, \ldots, I_i = Z$, where for each $p \in [i]_0$, $I_p \subseteq J_p$, then Z satisfies at least one set of constraints in $\mathcal{C}_i$.
- **Size Invariant:** The total number of constraints at the i^{th} step is $\sum_{C \in \mathcal{C}_i} |C| \leq (i + 1)!$.

At the base case, we let $\mathcal{C}_0 = \{\{(S, k)\}\}$. The correctness of the base case immediately follows from its construction. We now proceed recursively as follows. Consider $i \in [\ell]$. We assume that for each $p \in [i-1]$, we have computed $\mathcal{C}_p$ that satisfy the correctness and size invariants. Initialize $\mathcal{C}_i = \emptyset$.

For each $C \in \mathcal{C}_{i-1}$

For each constraint $(X, b) \in C$

1. Initialize a constraint set $C' = \emptyset$;

2. If $b = 1$

a. Add $(N(X) \cap J_i, 1)$ to C';

b. Add $(X' \cap J_i, b')$ for all other constraints $(X', b') \in C$ to C';

3. Else

a. Add $(X \cap J_i, b - 1)$ to C';

b. Add $(N(X) \cap J_i, 1)$ to C';

c. Add $(X' \cap J_i, b')$ for all other constraints $(X', b') \in C$ to C';

4. Add C' to $\mathcal{C}_i$;

▶ **Lemma 3.1.** *For every $C \in \mathcal{C}_i$, $\bigcup_{(X,b)\in C} X \subseteq J_i$.*

Proof. We use induction to prove the lemma. For the base case, $i = 0$, we have $\mathcal{C}_0 = \{\{(S, k)\}\}$. For $C = \{(S, k)\}$, we can see that the lemma holds. For the inductive step, we assume that the lemma holds true for $i - 1$ and prove that it still holds for i. So, for all $C \in \mathcal{C}_{i-1}$, $\cup_{(X,b)\in C} X \subseteq J_{i-1}$. In the i^{th} step of the algorithm, we add new sets of constraints C' such that all the constraints $(Y, \beta) \in C'$ have $Y \subseteq J_i$. Hence, their union must be a subset of J_i. This completes the proof of the lemma. ◀

▶ **Lemma 3.2.** *For every $C \in \mathcal{C}_i$, $\sum_{(X,b)\in C} b = k$.*

Proof. We use induction to prove the lemma. For the base case, $i = 0$, we have $\mathcal{C}_0 = \{\{(S, k)\}\}$. For $C = \{(S, k)\}$, we can see that the lemma holds. For the inductive step, we assume that the lemma holds true for $i - 1$, and prove it for i. So, for all $C \in \mathcal{C}_{i-1}$, $\sum_{(X,b)\in C} b = k$. In the i^{th} recursive step of the algorithm, we add a new set of constraints C' corresponding to each constraint (X, b) contained in some member of $\mathcal{C}_{i-1}$. If b is 1, we add another constraint with budget 1. Otherwise, we split the budget in the previous budget, i.e. b, into two parts $b - 1$ and 1. The total budget still remains the same as the $i - 1^{th}$ step, i.e., k. This completes the proof of the lemma. ◀

▶ **Lemma 3.3.** *For every $C \in \mathcal{C}_i$, all the vertex subsets which are part of the constraints in C are pairwise disjoint.*

Proof. We use induction to prove the lemma. For the base case, $i = 0$, we have $\mathcal{C}_0 = \{\{(S, k)\}\}$. For $C = \{(S, k)\}$, we can see that the lemma holds trivially. For $i = 1$, we have $\mathcal{C}_1 = \{\{(S \cap J_1, k - 1), (N(S) \cap J_1, 1)\}\}$. For $C = \{(S \cap J_1, k - 1), (N(S) \cap J_1, 1)\}$, we can see that $(S \cap J_1) \cap (N(S) \cap J_1) = \emptyset$ and the lemma holds. For the inductive step, we assume that the lemma holds true for $i - 1$, and prove it for i. So, for all $C \in \mathcal{C}_{i-1}$, all the vertex subsets which are part of the constraints in C are pairwise disjoint. In the i^{th} recursive step of the algorithm, we add a new set of constraints C' corresponding to each constraint (X, b) contained in some member of $\mathcal{C}_{i-1}$, say C. The sets $X' \cap J_i$ added corresponding to all constraints $(X', b') \in C$ such that $(X', b') \neq (X, b)$ are pairwise disjoint by the induction hypothesis. If $b > 1$, the set $X \cap J_i$ added is disjoint with all the sets $X' \cap J_i$ such that $(X', b') \in C$ and $(X', b') \neq (X, b)$ by the induction hypothesis. The set $N(X) \cap J_i$ added is disjoint with $X' \cap J_i$ for all $(X', b') \in C$ because all X' are part of an independent set J_i by Lemma 3.1, and none of them can have their neighbourhoods intersecting with the other sets. This completes the proof of the lemma. ◀

▶ **Lemma 3.4** (Size Invariant)**.** *The total number of constraints at the i^{th} step is $c_i = \sum_{C\in\mathcal{C}_i} |C| \leq (i + 1)!$. Therefore, $c_\ell \leq (\ell + 1)!$.*

Proof. Let $c_i = \sum_{C \in \mathcal{C}_i} |C|$. We have $c_0 = 1$ from the base case of the algorithm. At each step the number of constraints in a set of constraints added increases by at most 1. For $i = 0$, we have only one constraint in $\{(S, k)\} \in \mathcal{C}_0$. Therefore, at the i^{th} step, the maximum number of constraints in any set contained in $\mathcal{C}_i$ is at most $i + 1$. In the i^{th} recursive step of the algorithm, we add a new set of constraints C' corresponding to each constraint (X, b) contained in some member of $\mathcal{C}_{i-1}$. So, we get the following recursive relation: $|\mathcal{C}_i| = c_{i-1}$. Using the fact that all members of $\mathcal{C}_i$ contain at most $i+1$ constraints, we get that $c_i = \sum_{C \in \mathcal{C}_i} |C| \leq (i + 1)|\mathcal{C}_i| = (i + 1)c_{i-1}$. Solving the recurrence, we get $c_i \leq (i + 1)!$. Therefore, $c_\ell \leq (\ell + 1)!$. ◀

▶ **Lemma 3.5** (Correctness Invariant I). *If a k-sized independent set $Z \subseteq J_i$ satisfies at least one set of constraints in $\mathcal{C}_i$, then Z is reachable from S.*

Proof. We use induction to prove the lemma. For $i = 0$, we have $\mathcal{C}_0 = \{\{(S, k)\}\}$. For $C = \{(S, k)\}$, we can see that the lemma holds trivially. For the inductive step, we assume that the lemma holds true for $i - 1$, and prove it for i. Let $Z \subseteq J_i$ and $|Z| = k$ such that it satisfies some set of constraints $C \in \mathcal{C}_i$. Let (X, b) be the constraint in $C' \in \mathcal{C}_{i-1}$ which produces this set of constraints C in the i^{th} recursive step of the algorithm.

Let v^* be the vertex in $Z \cap (N(X) \cap J_i)$. Let u^* be a vertex in X sharing an edge with v^*. Take $Z' = (Z \cup \{u^*\}) \setminus \{v^*\}$. It can be seen that $|Z'| = k$ and Z can be obtained from Z' by sliding one token. Since Z satisfies the set of constraints C, we have:

1. $|Z \cap (N(X) \cap J_i)| = 1$
2. $|Z \cap (X \cap J_i)| = |Z \cap X| = b - 1$ (0 if $b = 1$)
3. $|Z \cap (X' \cap J_i)| = |Z \cap X'| = b'$ for all other constraints $(X', b') \in C'$

The way we construct Z', it must satisfy the following conditions:

1. $|Z' \cap X| = b \geq 1$ (since u^* is included in Z'); and
2. $|Z' \cap X'| = b'$ for all other constraints $(X', b') \in C'$.

It can be clearly seen that Z' satisfies the set of constraints $C' \in \mathcal{C}_{i-1}$. So, $|Z' \cap (\cup_{(X',b') \in C'} X')| = \sum_{(X',b') \in C'} |Z' \cap X'| = \sum_{(X',b') \in C'} b' = k$, where the first equality follows from the fact that all X' such that $(X', b') \in C'$ are pairwise disjoint by Lemma 3.3 and the last equality follows from Lemma 3.2. Therefore, $Z' \subseteq \cup_{(X',b') \in C'} X' \subseteq J_{i-1}$ by Lemma 3.1.

Thus, Z' is a k-sized subset of J_{i-1} and satisfies at least one set of constraints in $\mathcal{C}_{i-1}$. By the induction hypothesis, Z' is reachable from S. Now, since Z is reachable from Z', Z is also reachable from S. ◀

▶ **Lemma 3.6** (Correctness Invariant II). *For any k-sized independent set $Z \subseteq J_i$, if there is a reconfiguration sequence $S = I'_0, I'_1, I'_2, \ldots, I'_i = Z$, where for each $p \in [i]_0$, $I'_p \subseteq J_p$, then Z satisfies at least one set of constraints in $\mathcal{C}_i$.*

Proof. We use induction to prove the lemma. For $i = 0$, we have $\mathcal{C}_0 = \{\{(S, k)\}\}$. The set S satisfies the set of constraints $C = \{(S, k)\}$ and the lemma holds.

We now assume that the lemma holds true for $i - 1$, and prove it for i. Let $C \in \mathcal{C}_{i-1}$ be the set of constraints that I'_{i-1} satisfies. So, $|I'_{i-1} \cap (\cup_{(X,b) \in C} X)| = \sum_{(X,b) \in C} |I'_{i-1} \cap X| = \sum_{(X,b) \in C} b = k$, where the first equality follows from the fact that all X such that $(X, b) \in C$ are pairwise disjoint by Lemma 3.3 and the last equality follows from Lemma 3.2. Since $|I'_{i-1}| = k$, we have $I'_{i-1} \subseteq \cup_{(X,b) \in C} X$. In the i^{th} step of the reconfiguration sequence, we slide a token from I'_{i-1} to I'_i, i.e. from some set X such that $(X, b) \in C$ to its open neighbourhood. Consider the set of constraints $C' \in \mathcal{C}_i$ obtained by splitting the constraint

(X, b) in the i^{th} recursive step of the algorithm. We will show that I'_i satisfies C'. Since I'_{i-1} satisfies the set of constraints C, we have $|I'_{i-1} \cap X| = b$ for all other constraints $(X, b) \in C$. So, in the i^{th} step we have $|I'_i \cap (N(X) \cap J_i)| = |I'_i \cap N(X)| = 1$, where the first equality is because $I'_i \subseteq J_i$ and the second equality is because I'_i is obtained from I'_{i-1} by sliding a token from X to its neighbourhood. We have $|I'_i \cap (X \cap J_i)| = |I'_i \cap X| = |I'_{i-1} \cap X| - 1 = b - 1$ as one token is moved from X. When $b = 1$, we get $I'_i \cap X = \emptyset$ and this budget constraint is not included in the i^{th} recursive step of the algorithm. For all other $(X', b') \in C$, we have $|I'_i \cap (X' \cap J_i)| = |I'_i \cap X'| = |I'_{i-1} \cap X'| = b'$ as none of the tokens in any X' are moved in the i^{th} step of the reconfiguration sequence. Therefore, $I'_i = Z$ satisfies all the constraints in C', as needed. ◀

We are now ready to prove our first main theorem.

▶ **Theorem 3.7.** *Token Sliding Optimization is fixed-parameter tractable parameterized by $k + \ell + d$ where d denotes the degeneracy of the graph.*

Proof. Let (G, S, T, k, ℓ) denote an instance of TSO, where G is d-degenerate. We first computed in time $O((kd)^{O(k)} \cdot (n + m) \log n)$ an independence covering family $\mathcal{F}(G, k)$ for (G, k) of size at most $O((kd)^{O(k)} \cdot \log n)$ (by Theorem 1.2). We then add S and T to $\mathcal{F}(G, k)$ (in case they do not already belong to $\mathcal{F}(G, k)$). Next, we "guess" a (iterate over every) sequence $\langle J_0, J_1, ..., J_\ell \rangle$ of elements of $\mathcal{F}(G, k)$ such that $J_0 = S$, $J_\ell = T$. Note that this guessing can be accomplished in time $O(((kd)^{O(k)} \cdot \log n)^{\ell+1})$, which by Proposition 2.1 is still FPT-time. Finally, we compute $\mathcal{C}_0$, $\mathcal{C}_1$, …, $\mathcal{C}_\ell$, which by Lemma 3.4 can also be done in FPT-time. To conclude, we simply need to check whether T satisfies at least one set of constraints in $\mathcal{C}_\ell$. The correctness of the algorithm follows from Lemma 3.5 and 3.6. ◀

▶ **Theorem 3.8.** *Token Sliding Optimization is fixed-parameter tractable parameterized by $|N| + k + \ell + d$ on graphs having a modulator N whose deletion leaves a d-degenerate graph (assuming N is given as part of the input).*

Proof. We proceed exactly as in the proof of Theorem 3.7 but we invoke Theorem 1.3 instead of Theorem 1.2. ◀

We conlude this section by showing how we can adapt the previous two results for the Token Jumping Optimization problem.

▶ **Theorem 3.9.** *Token Jumping Optimization is fixed-parameter tractable parameterized by $k + \ell + d$ where d denotes the degeneracy of the graph and fixed-parameter tractable parameterized by $|N| + k + \ell + d$ on graphs having a modulator N whose deletion leaves a d-degenerate graph (assuming N is given as part of the input).*

Proof. To allow tokens to jump to arbitrary vertices of the graph we only need to slightly modify our construction of the sets of sets of constraints $\mathcal{C}_1$, …, $\mathcal{C}_\ell$. In particular, we do the following:

- For each $C \in \mathcal{C}_{i-1}$
  - Initialize a constraint set C' obtained from C by replacing each (X, b) by $(X \cap J_i, b)$;
  - If $X \cap J_i \neq \emptyset$ and $|X| \geq b$ for all X then add C' to $\mathcal{C}_i$;
  - For each constraint $(X, b) \in C$
    1. Initialize a constraint set $C' = \emptyset$;
    2. If $b = 1$
       a. Add $((N(X) \cap J_i) \cup (J_i \setminus \cup_{(X,b) \in C} X), 1)$ to C';

b. Add $(X' \cap J_i, b')$ for all other constraints $(X', b') \in C$ to C';

3. Else

a. Add $(X \cap J_i, b-1)$ to C'.

b. Add $((N(X) \cap J_i) \cup (J_i \setminus \cup_{(X,b) \in C} X), 1)$ to C';

c. Add $(X' \cap J_i, b')$ for all other constraints $(X', b') \in C$ to C';

4. Add C' to $\mathcal{C}_i$;

5. If $|C| > 1$

a. Initialize $C' = C \setminus (X, b)$ when $b = 1$ and $C' = (C \setminus (X, b)) \cup (X, b-1)$ otherwise;

b. For each $(X', b') \in C'$ where $X' \neq X$

i. Create a new constraint set $C'' = (C' \setminus (X', b')) \cup (X', b'+1)$;

ii. If $|X'| \geq b'$ for all X add C'' to $\mathcal{C}_i$;

It is not hard to see that the correctness invariants remain true. Note that at every step we create at most $\mathcal{O}(k)$ constraint sets (each of size $\mathcal{O}(k)$) and so the size of the contraint sets and the time to compute them is bounded by a function of k and ℓ. ◀

4 FPT algorithm for parameter $|M| + \ell + \Delta$

In this section, we prove that TSO and TJO are fixed-parameter tractable parameterized by $|M| + \ell + \Delta$. Recall that an instance of either problem is denoted by (G, S, T, k, ℓ) where $V(G)$ can be partitioned into H and M and every vertex in H has degree at most Δ in G. Our algorithm is randomized and based on a variant of the color-coding technique [1] that is particularly useful in designing parameterized algorithms on graphs of bounded degree. The technique, known in the literature as random separation [8], boils down to a simple, but fruitful observation that in some cases, if we randomly color the vertex set of a graph using two colors, the solution or vertices we are looking for are appropriately colored with high probability. In our case, we want to make sure that the set of vertices involved in token slides or jumps gets highlighted. We note that our algorithm is an adaptation of the algorithm of Mouawad et al. [25] and it can easily be derandomized using standard techniques [9].

We start with an instance $(G = (H, M, E), S, T, k, \ell)$ of TSO; the algorithm is identical for TJO. We color independently every vertex of H using one of two colors, say red and blue (denoted by $\mathcal{R}$ and $\mathcal{B}$), with probability $\frac{1}{2}$. We let $\chi : H \to \{\mathcal{R}, \mathcal{B}\}$ denote the resulting random coloring. Suppose that (G, S, T, k, ℓ) is a yes-instance, and let σ denote a reconfiguration sequence from S to T of length at most ℓ. We say a vertex $v \in H$ is *touched* in σ whenever a token slides from a neighbor of v to v or from v to some neighbor of v. We let $V(\sigma)$ denote the set of vertices touched by σ. We say that the coloring χ is successful if both of the following conditions hold:

- Every vertex in $V(\sigma) \cap H$ is colored red; and
- Every vertex in $N_H(V(\sigma) \cap H)$ is colored blue.

Observe that $V(\sigma) \cap H$ and $N_H(V(\sigma) \cap H)$ are disjoint. Therefore, the two aforementioned conditions are independent. Moreover, since the maximum degree of $G[H]$ is Δ, we have $|V(\sigma) \cap H| + |N_H(V(\sigma) \cap H)| \leq 2\ell\Delta$. Consequently, the probability that χ is successful is at least:

$$\frac{1}{2^{|V(\sigma) \cap H| + |N_H(V(\sigma) \cap H)|}} \geq \frac{1}{2^{2\ell\Delta}} = \frac{1}{4^{\ell\Delta}}.$$

Let $H_{\mathcal{R}}$ denote the set of vertices of H colored red and $H_{\mathcal{B}}$ denote the set of vertices of H colored blue. Moreover, we let $C_1, \ldots, C_q$ denote the set of connected components of $G[H_{\mathcal{R}}]$. The main observation now is the following:

▶ **Lemma 4.1.** *If χ is successful then $V(\sigma)$ has a non-empty intersection with at most 2ℓ connected components of $G[H_{\mathcal{R}}]$, and each one of those components consists of at most 2ℓ vertices.*

Proof. Since $|V(\sigma)| \leq 2\ell$, we know that $G[(V(\sigma) \cup N_G(V(\sigma))) \cap H]$ consists of at most 2ℓ connected components (each of size at most $2\ell\Delta$) and $G[V(\sigma) \cap H]$ consists of at most 2ℓ components (each of size at most 2ℓ). Let C be a connected component of $G[H_{\mathcal{R}}]$ such that $|V(C)| > 2\ell$. Suppose to the contrary that $V(\sigma) \cap V(C) = Q \neq \emptyset$. Since χ is successful, it must be the case that every vertex in $N_H(Q)$ is colored blue. However, we know that there exists at least one vertex in $N_H(Q)$ that is colored red (since C is a connected component of $G[H_{\mathcal{R}}]$ and all at least $2\ell + 1$ vertices in C are colored red). As we have obtained a contradiction, we can conclude that when χ is successful, $V(\sigma)$ can intersect at most 2ℓ connected components of $G[H_{\mathcal{R}}]$, and none of those components can be of a size greater than 2ℓ, as needed. ◀

Given an instance $(G = (H, M, E), S, T, k, \ell)$ of TSO and a coloring χ of H, we know from Lemma 4.1 that when χ is successful every connected component of $G[H_{\mathcal{R}}]$ consists of at most 2ℓ vertices. We now construct a new (reduced) instance (G', S', T', k', ℓ) of TSO. We first guess the vertices of M that will be touched in a solution and we let M' denote this set. Note that this guessing can be accomplished in time 2^M-time. Starting from a copy of G we proceed as follows:

- If there exists $v \in (S \cap T) \cap H$ and v is colored blue then we delete v and its neighbors from the graph;
- If there exists $v \in (S \cap T) \cap (M \setminus M')$ then we delete v and its neighbors from the graph;
- If there exists $v \in (S \cap T) \cap H$, v is colored red, and v belongs to a red component C of $G[H_{\mathcal{R}}]$ such that $|V(C)| > 2\ell$ then we delete v and its neighbors from the graph;
- If there exists a blue vertex v which is not in $S \cap T$ then we delete v from the graph;
- If there exists a red vertex v which is not in $S \cap T$ and v belongs to a red component C of $G[H_{\mathcal{R}}]$ such that $|V(C)| > 2\ell$ then we delete v from the graph.

We adjust S, T, and k appropriately to obtain the new equivalent instance (G', S', T', k', ℓ). Note that in this new instance (assuming a successful coloring) no vertices are colored blue and (assuming a correct guess) all vertices of M' will be touched in a solution. In other words, G' can be partitioned into M' and H' where H' consists of (an unbounded number of) connected components each consisting of at most 2ℓ vertices. Note that when the number of connected components is constant then we are done since we can solve the problem via brute-force. In other words, we can simply enumerate all possible sequences of length at most ℓ and make sure that at least one of them is the required reconfiguration sequence from S' to T'. This brute-force testing can be accomplished in time $2^{\mathcal{O}(\ell \log \ell)} \cdot n^{\mathcal{O}(1)}$.

Let us now consider the general case when the number of components is not necessarily bounded. We say a component C of $G' - M'$ is *important* if $V(C) \cap ((S' \setminus T') \cup (T' \setminus S')) \neq \emptyset$. There are at most 2ℓ important components. Hence, we only need to bound the number of *unimportant* components. To that end, we partition the unimportant components of $G' - M'$ into equivalence classes with respect to the relation $\simeq$. For two graphs G_1, G_2 and two sets $X_1 \subseteq V(G)$, $X_2 \subseteq V(G_2)$, we say that (G_1, X_1) and (G_2, X_2) are *isomorphic* if the graphs G_1 and G_2 are isomorphic where vertices of X_1 and X_2 are now assigned the same color. Formally, a *c-colored graph* G is a tuple $(V, E, \mathcal{K})$ such that $\mathcal{K} = \{K_1, \ldots, K_c\}$ is a collection of subsets of $V(G)$ where each K_i is called a *color set*. Two colored graphs $G_1 = (V_1, E_1, \mathcal{K}_1)$ and $G_2 = (V_2, E_2, \mathcal{K}_2)$ are isomorphic if there is a *color-preserving isomorphism* $f : V_1(G_1) \to V_2(G_2)$ such that:

- for all $u, v \in V_1(G_1)$, $\{u, v\} \in E_1(G_1)$ if and only if $\{f(u), f(v)\} \in E_2(G_2)$; and
- for all $v \in V_1(G_1)$ and $K_i^1 \in \mathcal{K}_1$, $v \in K_i^1$ if and only if $f(v) \in K_i^2$.

Hence, (G_1, X_1) and (G_2, X_2) are isomorphic if the colored graphs $G_1 = (V_1, E_1, \{X_1\})$ and $G_2 = (V_2, E_2, \{X_2\})$ are isomorphic. Let C_1 and C_2 be two components in $G' - M'$ and let N_1 and N_2 be their respective neighborhoods in M'. We say C_1 and C_2 are *equivalent*, i.e., $C_1 \simeq C_2$, whenever $N_1 = N_2 = N$ and $(G[V(C_1) \cup N], V(C_1) \cap S' \cap T')$ is isomorphic to $(G[V(C_2) \cup N], V(C_2) \cap S' \cap T')$ by an isomorphism that fixes N point-wise.

▶ **Lemma 4.2.** *The total number of 2-colored graphs with at most 2ℓ vertices is at most $2^{\mathcal{O}(\ell^2)}$, and therefore, the equivalence relation $\simeq$ has at most $2^{\mathcal{O}(\ell^2)}$ equivalence classes.*

Assume that some equivalence class contains more than 2ℓ unimportant components. We claim that retaining only 2ℓ of them is enough. To see why, it is enough to note that $V(\sigma)$ intersects with at most 2ℓ of those components; they are all equivalent. Putting it all together, we know that we have at most $2^{\mathcal{O}(\ell^2)}$ equivalence classes, each with at most 2ℓ components, and each component is of size at most 2ℓ. Hence, we can guess the sequence from S' to T' in time $2^{\mathcal{O}(\ell^3 \log \ell)} \cdot n^{\mathcal{O}(1)}$ (testing whether two graphs with 2ℓ vertices are isomorphic can be accomplished naively in time $2^{\ell \log \ell}$).

We have proven that the probability that χ is successful is at least $4^{-\ell\Delta}$. Hence, to obtain a Monte Carlo algorithm with false negatives, we repeat the above procedure $4^{\ell\Delta}$ times and obtain the following result:

▶ **Theorem 4.3.** *There exists a one-sided error Monte Carlo algorithm with false negatives that solves TSO and TJO parameterized by $|M| + \ell + \Delta$ in time $\mathcal{O}(2^M \cdot 4^{\ell\Delta} \cdot 2^{\mathcal{O}(\ell^3 \log \ell)} \cdot n^{\mathcal{O}(1)})$.*

5 Hardness of TSO parameterized by ℓ on 2-degenerate graphs

In the Multicolored Clique problem, we are given an input graph G whose vertices are colored with k colors and the goal is to find a clique containing one vertex from each color. We show that TSO parameterized by ℓ is W[1]-hard on 2-degenerate graphs via a reduction from Multicolored Clique, known to be W[1]-hard.

We construct an instance $(G', S, T, \kappa, \ell = 8\binom{k}{2} + 2k)$ of TSO from an instance of Multicolored Clique denoted by $(G, k, (V_1, V_2, \ldots, V_k))$, where, w.l.o.g., we assume that there are no edges between two vertices of G of the same color.

Construction of G'. We subdivide all the edges in G. Let the vertex set of G be V. All the vertices corresponding to the edges in G are partitioned into $\binom{k}{2}$ sets of the form $\mathcal{E}_{ij}$, where $i = \{1, 2, \ldots, k\}$ and $j = \{1, 2, \ldots, k\}$ and $i \neq j$, such that $\mathcal{E}_{ij}$ contains all the vertices corresponding to the edges in G having one incident vertex of color i and the other incident vertex of color j. Let the union of all the sets $\mathcal{E}_{ij}$ be denoted by E.

We introduce two independent sets X and Y, each of size $\binom{k}{2}$. Let us label the vertices in X from 1 to $\binom{k}{2}$ and the sets $\mathcal{E}_{ij}$ from 1 to $\binom{k}{2}$. We add edges between vertex with label b in X and all the vertices in the $\mathcal{E}_{ij}$ with label b. Similarly, we label the vertices of Y and add edges from each vertex in Y to all the vertices in the $\mathcal{E}_{ij}$ having the same label. Each of these edges is further subdivided three times. Let the vertices on the subdivided edges from X to E, which are neither adjacent to some vertex in X nor E be denoted by U_1, and the vertices on the subdivided edges from Y to E, which are neither adjacent to some vertex in Y nor E be denoted by U_2. We take $U = U_1 \cup U_2$.

We also add a vertex corresponding to each vertex in V and add an edge between the two. Let this set of vertices be Z. The induced subgraph of G' having $V \cup Z$ as its

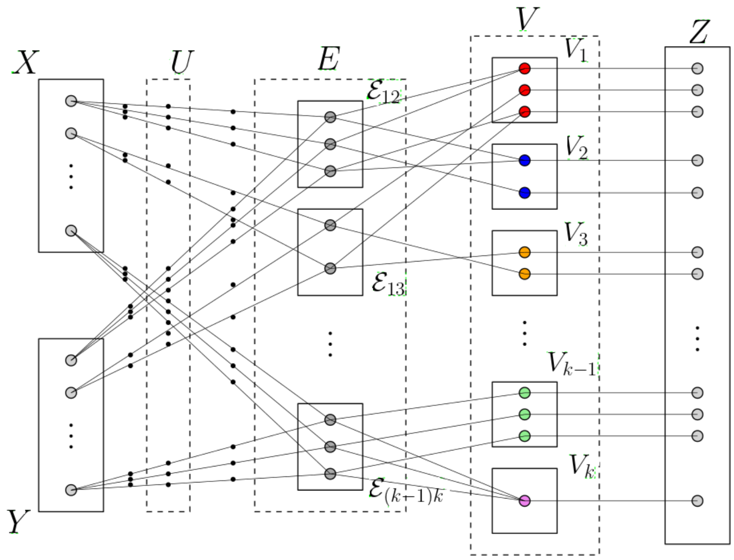

Figure 1 An illustration of the reduction from $(G, k, (V_1, V_2, \ldots, V_k))$ to $(G', S, T, \kappa, \ell = 8\binom{k}{2}+2k)$.

vertex set forms a perfect matching. Our initial independent set $S = V \cup X \cup U$ and our target independent set $T = V \cup Y \cup U$. Note that $|S| = |T| = n + \binom{k}{2} + |U| = \kappa$. We set $\ell = 8\binom{k}{2} + 2k$.

▶ **Lemma 5.1.** *The graph G' is 2-degenerate.*

Proof. Recall that a graph G' is 2-degenerate if every induced subgraph H of G' has a vertex of degree at most 2. Consider any induced subgraph H of G'. If H contains a vertex of Z or a vertex from the subdivided edges from $X \cup Y$ to E then we are done; as those vertices have degree at most two in G'. Otherwise, we know that H either contains an isolated vertex from $X \cup Y$ or a degree-two vertex from E, as needed. ◀

▶ **Lemma 5.2.** *If $(G, k, (V_1, V_2, \ldots, V_k))$ is a yes-instance of* Multicolored Clique *then there is a reconfiguration sequence of length at most ℓ from S to T in G'.*

Proof. Let the solution to the Multicolored Clique instance be $\{v_1, v_2, \ldots, v_k\} \subseteq V$. Consider the following reconfiguration sequence from S to T:

1. Slide each token on v_i to its matched neighbour in Z; for a total of k slides.
2. Since the vertices $\{v_1, v_2, \ldots, v_k\}$ form a clique in G, there are $\binom{k}{2}$ edges, each having distinct pair of colors on their incident vertices. So in G', all the vertices corresponding to the edges of the clique lie in distinct partitions $\mathcal{E}_{ij}$. We slide all the tokens from X to Y using these $\binom{k}{2}$ vertices. Consider the path from a vertex $v_x \in X$ to a vertex $v_y \in Y$, passing through one of these $\binom{k}{2}$ vertices, say v_i where $i \in [k]$. This path contains a vertex $u_1 \in U_1$ and a vertex $u_2 \in U_2$. Slide the token on u_2 to v_y (2 slides), the token on u_1 to u_2 through v_i (4 slides), and the token on v_x to u_1 along this path (2 slides); for a total of $8\binom{k}{2}$ slides.
3. Finally we slide the tokens in Z back to V; for a total of k slides.

The length of the reconfiguration sequence is $8\binom{k}{2} + 2k$. This completes the proof. ◀

▶ **Lemma 5.3.** *If there is a reconfiguration sequence of length at most ℓ from S to T in G' then $(G, k, (V_1, V_2, \ldots, V_k))$ is a yes-instance of* Multicolored Clique.

Proof. Let the reconfiguration sequence be $I_0, I_1, I_2, \ldots, I_\ell$ where $I_0 = S$, $I_\ell = T$, and $\ell \leq 8\binom{k}{2} + 2k$.

We need at least one step for moving out each token in X. This requires a total of $\binom{k}{2}$ slides. In order to move out a token from X, we need to move out a token on U_1 on at least one of the paths connecting that vertex in X to $\mathcal{E}_{ij}$. This again requires at least $\binom{k}{2}$ slides. The tokens moved out from U_1 need to be replaced, which requires at least $\binom{k}{2}$ slides.

Since in the initial configuration S all tokens are at a distance of at least 2 from the vertices in Y, we need at least 2 slides to bring a token into a vertex in Y. This amounts to a total of $2\binom{k}{2}$ slides. Now, consider the vertex v from which a token is moved to a vertex in Y. The neighbour of v, say u, originally had a token, which must have been moved out for placing a token on v. This token needs to be replaced by moving in a token from an adjacent vertex in $N(E)$, say v'. Bringing a token to v' and then moving it to u requires at least 2 slides. So, we get a total of at least $2\binom{k}{2}$ slides.

The only way to move tokens out of $X \cup N[U_1]$ is through E. Thus, $\binom{k}{2}$ tokens have to be moved out of $X \cup N[U_1]$ to E. Every $\mathcal{E}_{ij}$ has at least one token at some point of time. This requires another $\binom{k}{2}$ slides. In total we have taken up at least $8\binom{k}{2}$ slides. So, we are left with a budget of at most $2k$.

Any token moved out of V either needs to be brought back or replaced. Both of these require 2 slides at least. So, we can move out at most k tokens from V.

Whenever a token is to be moved out of $X \cup N[U_1]$ to a vertex v_e (corresponding to an edge e in G) in $\mathcal{E}_{ij}$ for some i and j, the tokens on the 2 vertices incident on e must be moved out of V. We need to move out all the $\binom{k}{2}$ tokens from $X \cup N[U_1]$, one token through each of the $\binom{k}{2}$ sets $\mathcal{E}_{ij}$. Therefore, we should move out the tokens from V which are adjacent to the vertices in $\mathcal{E}_{ij}$ sets to which the tokens from $X \cup N[U_1]$ are moved. Thus for the $\binom{k}{2}$ vertices in E (one in each $\mathcal{E}_{ij}$), we can have at most k neighbours in V.

Let us consider the subgraph induced in G by these set of vertices in V. It has at most k vertices and exactly $\binom{k}{2}$ edges. Now a graph having $\binom{k}{2}$ edges must have at least k vertices. So, the induced subgraph has exactly k vertices and forms a clique such that every edge has a distinct pair of colors on their incident vertices. This set of vertices in V give us a solution to the MULTICOLORED CLIQUE instance. ◀

The combination of Lemmas 5.2 and 5.3 give us the following:

▶ **Theorem 5.4.** *TOKEN SLIDING OPTIMIZATION parameterized by ℓ is W[1]-hard on 2-degenerate graphs.*

6 Hardness of TJO parameterized by ℓ on 2-degenerate graphs

We now show that TJO parameterized by ℓ is W[1]-hard on 2-degenerate graphs via a reduction from the CLIQUE problem, known to be W[1]-hard. We construct an instance $(G', S, T, \kappa, 2\binom{k}{2} + \binom{k}{2}^2 + 2k)$ of TJO starting from the a CLIQUE instance (G, k). The construction is quite similar to that of the sliding variant but with some adaptation to account for the possibility of tokens jumping anywhere in the graph.

Construction of G'. We subdivide all the edges in G. Let the vertex set of G be V. We let the set of vertices in G' corresponding to the edges be denoted by E. We introduce a biclique with parts L and R, each of size $\binom{k}{2}$. Next, we subdivide all the edges of the biclique twice. Let this entire set of vertices, i.e. $L \cup N(L) \cup N(R) \cup R$ be denoted by X. The vertices in X do not have edges with those in E or V. We also add a vertex corresponding

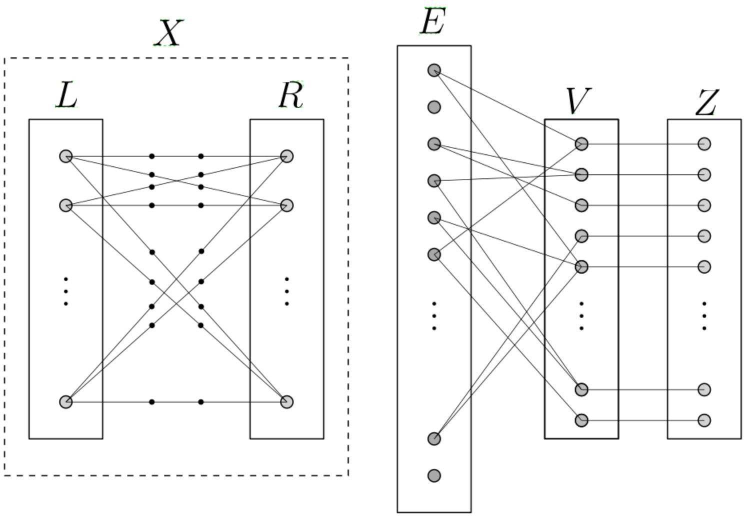

Figure 2 An illustration of the reduction from (G, k) to $(G', S, T, \kappa, 2\binom{k}{2} + \binom{k}{2}^2 + 2k)$.

to each vertex in V and add an edge between the two. Let this set of vertices be Z. The induced subgraph of G' having $V \cup Z$ as its vertex set forms a perfect matching. Our initial independent set $S = V \cup L \cup N(R)$ and our target independent set $T = V \cup R \cup N(L)$. Note that $|S| = |T| = \kappa$. We let $\ell = 2\binom{k}{2} + \binom{k}{2}^2 + 2k$.

▶ **Lemma 6.1.** *The graph G' is 2-degenerate.*

Proof. Consider any induced subgraph H of G'. If H contains a vertex of Z or a vertex from $N(L) \cup N(R)$ then we are done; as those vertices have degree at most two in G'. Otherwise, we know that H either contains an isolated vertex from $L \cup R$ or a degree-two vertex from E, as needed. ◀

▶ **Lemma 6.2.** *If (G, k) is a yes-instance of* Clique *then there is a reconfiguration sequence of length at most ℓ from S to T in G'.*

Proof. Let the solution to the Clique instance be $\{v_1, v_2, \ldots, v_k\} \subseteq V$. Consider the following reconfiguration sequence from S to T:

1. Jump each token on v_i to its matched neighbour in Z; for a total of k jumps.
2. Since the vertices $\{v_1, v_2, \ldots, v_k\}$ form a clique in G, there are $\binom{k}{2}$ edges in the subgraph induced on those vertices. Let the set of corresponding vertices in E be $E_{\mathcal{C}}$. We jump all the $\binom{k}{2}$ tokens from L to the vertices in $E_{\mathcal{C}}$; for a total of $\binom{k}{2}$ jumps.
3. Jump all the tokens in $N(R)$ to their adjacent vertex in $N(L)$; for a total of $\binom{k}{2}^2$ jumps.
4. Now jump all the tokens in $E_{\mathcal{C}}$ to R; for a total of $\binom{k}{2}$ jumps.
5. Finally we jump the tokens in Z back to V; for a total of k jumps.

The length of the reconfiguration sequence is $2\binom{k}{2} + \binom{k}{2}^2 + 2k$. This completes the proof. ◀

▶ **Lemma 6.3.** *The first time a token jumps to R, there can be at most $\binom{k}{2}^2$ tokens in the structure X.*

Proof. Let us assume that the first time a token jumps to R there are $\binom{k}{2}^2 + 1$ tokens in X. Also, let the vertex in R where a token is to be moved be v. If $y > 0$ of these tokens are in L

and none of them are in R then there cannot be any tokens on the vertices of the subdivided edges from those y vertices in L to v. So, we have at most y tokens on the vertices in L and those on the paths from them to v. We have $\binom{k}{2}^2 - y$ paths from the remaining vertices in L to the vertices in $R \setminus \{v\}$. These paths can have at most $\binom{k}{2}^2 - y$ tokens. Thus, in total we can have at most $y + \binom{k}{2}^2 - y = \binom{k}{2}^2$ tokens in X. This leads us to a contradiction, which completes the proof. ◀

▶ **Lemma 6.4.** *If there is a reconfiguration sequence of length at most ℓ from S to T in G' then (G, k) is a yes-instance of* Clique.

Proof. Let the reconfiguration sequence be $I_0, I_1, I_2, \ldots, I_\ell$, where $I_0 = S$, $I_\ell = T$, and $\ell \leq 2\binom{k}{2} + \binom{k}{2}^2 + 2k$. None of the tokens in X have the same position in both S and T. Hence, all of them have to jump at least once. This accounts for $\binom{k}{2} + \binom{k}{2}^2$ steps. From Lemma 6.3, we know that no tokens can be moved into R until we have no more than $\binom{k}{2}^2$ tokens left in X. This implies that at least $\binom{k}{2}$ of the tokens in X have to be moved out to either E, V, or Z. When we are about to move a token into R for the first time, we can have at most $\binom{k}{2}^2$ tokens in X. So, an extra $\binom{k}{2}$ tokens have to be moved into X, which takes at least $\binom{k}{2}$ steps. Therefore, we are left with a budget of at most $2k$. We consider the following three cases while jumping tokens out of X:

- **Case I:** If a token from X jumps to some vertex v_e in E, the tokens on the two neighbouring vertices of v_e in V should have been moved out to E or Z (we are not considering X, as moving a token from V to X in order to shift a token out of X does not help).
- **Case II:** If a token from X is to be moved to a vertex v_z in Z, the token on the neighbouring vertex of v_z in V needs to be moved out to E or Z. If that is to be moved to some vertex in Z, then the token on the neighbour of the matched vertex in V needs to be jumped out of V. Again if that token is to be moved out to Z, the sequence continues. At most $n - 1$ tokens can be jumped out of V to Z, because the initial token from X would occupy one vertex in Z. So, after at most $n - 1$ steps in the sequence, the token must be moved out of V to some vertex v_e in E. Now, the tokens on the two neighbours of v_e in V must have been moved out of V prior to the above sequence of jumps. Thus, we can consider a shorter reconfiguration sequence where the token from X is directly jumped to v_e after shifting the tokens on its neighbours in V. Then it becomes similar to Case I.
- **Case III:** If a token from X is to be moved to a vertex v_v in V, the token on v_v needs to be moved out to $E \setminus N(v_v)$ or $Z \setminus N(v_v)$. If that is to be moved to some vertex in $Z \setminus N(v_v)$, then the token on the neighbour of the matched vertex in V needs to be jumped out of V. Again if that token is to be moved out to Z, the sequence continues. At most $n - 1$ tokens can be jumped out of V to Z, because we cannot place a token on the neighbour of v_v in Z. So, after at most $n - 1$ steps in the sequence, the token must be moved out of V to some vertex v_e in E. Now, the tokens on the two neighbours of v_e in V must have been moved out of V prior to the above sequence of jumps. Thus, we can consider a shorter reconfiguration sequence where the token from X is directly jumped to v_e after shifting the tokens on its neighbours in V. Thus, it becomes similar to Case I.

By the above case analysis, it is sufficient to consider that Case I holds for our reconfiguration sequence. We need to move out at least $\binom{k}{2}$ tokens from X through the vertices in E. Therefore, we should move out the tokens from V which are adjacent to the vertices in E to which the tokens from X are moved. While we are moving tokens out of X, any token moved out of V eventually needs to be moved to X or brought back to V. Both of these take at least 2 steps because we directly cannot jump a token from V to X until X contains at most $\binom{k}{2}^2$ tokens. So, we can move out at most k tokens from V. Thus for the $\binom{k}{2}$ vertices in E, we can have at most k neighbours in V.

Let us consider the subgraph in G induced by this set of vertices in V. It has at most k vertices and exactly $\binom{k}{2}$ edges. Now a graph having $\binom{k}{2}$ edges must have at least k vertices. So, the induced subgraph has exactly k vertices and forms a clique. This set of vertices in V give us a solution to the Clique instance, as needed. ◀

The combination of Lemmas 6.2–6.4 give us the follwing:

▶ **Theorem 6.5.** *Token Jumping Optimization parameterized by ℓ is W[1]-hard on 2-degenerate graphs.*

7 FPT algorithm for Token Jumping Reachability parameterized by k

We propose a generalized scheme for solving Token Jumping Reachability parameterized by k on graphs having a small k-independence covering family, i.e., a family of size $\mathcal{O}(f(k) \cdot \text{poly}(n))$. Degenerate and nowhere dense graphs admit such independence covering families as shown in [24].

We remove all the sets in the covering family of size less than k. We find out if the independent sets S and T are a part of the independence covering family. If not, we add them to the family. Let the size of the resulting k-independence covering family $\mathcal{F}(G, k)$ be q. For denoting an independent set in the family, we will use capital letters like, $X, Y (\subseteq V(G))$. We construct a graph $\mathcal{G}$ with q vertices corresponding to the q sets in the family. Consider two independent sets I and I' in $\mathcal{F}(G, k)$. We add an edge between the vertices i and i' in $\mathcal{G}$ if and only if $|I \cap I'| \geq k - 1$. Note that for any two k-sized independent sets J and J' we can find a trivial reconfiguration sequence from J to J' if both of them are contained in some I in $\mathcal{F}(G, k)$.

In the algorithm, we find out if the vertices i_s and i_t in $\mathcal{G}$ are in the same connected component. If yes, then we know that S is reachable from T from the construction of $\mathcal{G}$. Otherwise no reconfiguration sequence from S to T exists.

▶ **Lemma 7.1.** *If there exists a path from i_s to i_t in $\mathcal{G}$ then there is a reconfiguration sequence from S to T in G.*

Proof. Let $i_s = i_0, i_1, i_2, \dots, i_\ell = i_t$ be the path from i_s to i_t. We start the reconfiguration sequence with S. For each pair of vertices i_j and i_{j+1} in the path, we have $|I_j \cap I_{j+1}| \geq k - 1$ according to the construction. Now, let $X_j \subseteq I_j$ be a k-sized independent set in the reconfiguration sequence and $Y_j \subseteq I_j \cap I_{j+1}$ be a $(k-1)$-sized set. Let u_j be a vertex in X_j. We can obtain a k-sized independent set $Z_j = Y_j \cup \{u_j\}$ from X_j by at most $k - 1$ token jumps. Next we jump the token on u_j to a vertex in $I_{j+1} \setminus Y_j$ to obtain a k-sized independent set $X_{j+1} \subseteq I_{j+1}$. This gives us a reconfiguration sequence from S to T, as needed. ◀

▶ **Lemma 7.2.** *If there is a reconfiguration sequence $S = I_0, I_1, I_2, \dots, I_\ell = T$ then there exists a path from i_s to i_t in $\mathcal{G}$.*

Proof. Let $I'_1, I'_2, \dots, I'_{\ell-1}$ be the sets in the covering family such that $I_i \subseteq I'_i$ for $i \in [\ell - 1]$. Since $|I_i \cap I_{i+1}| = k - 1$, we have $|I'_i \cap I'_{i+1}| \geq k - 1$. If I'_i and I'_{i+1} are the same set, then they correspond to the same vertex in $\mathcal{G}$. Otherwise, they are connected by an edge according to the construction of $\mathcal{G}$. We start from the vertex i_s and following the reconfiguration sequence, we reach i_t. This gives us a walk from i_s to i_t and a walk contains a path, as needed. ◀

The combination of Lemmas 7.1 and 7.2 give us the following:

▶ **Theorem 7.3.** *Token Jumping Reachability parameterized by k is fixed-parameter tractable on any graph class $\mathcal{C}$ for which we can, given any n-vertex graph $G \in \mathcal{C}$, compute a k-independence covering family $\mathcal{F}(G, k)$ of size $\mathcal{O}(f(k) \cdot n^{\mathcal{O}(1)})$ in time $\mathcal{O}(g(k) \cdot n^{\mathcal{O}(1)})$, where f and g are computable functions.*

References

1 Noga Alon, Raphael Yuster, and Uri Zwick. Color coding. In *Encyclopedia of Algorithms*, pages 335–338. 2016. doi:10.1007/978-1-4939-2864-4_76.

2 Valentin Bartier, Nicolas Bousquet, and Amer E. Mouawad. Galactic Token Sliding. *CoRR*, abs/2204.05549, 2022. URL: https://arxiv.org/abs/2204.05549, arXiv:2204.05549.

3 Rémy Belmonte, Eun Jung Kim, Michael Lampis, Valia Mitsou, Yota Otachi, and Florian Sikora. Token sliding on split graphs. *Theory Comput. Syst.*, 65(4):662–686, 2021. doi:10.1007/s00224-020-09967-8.

4 Marthe Bonamy and Nicolas Bousquet. Token sliding on chordal graphs. In Hans L. Bodlaender and Gerhard J. Woeginger, editors, *Graph-Theoretic Concepts in Computer Science - 43rd International Workshop, WG 2017, Eindhoven, The Netherlands, June 21-23, 2017, Revised Selected Papers*, volume 10520 of *Lecture Notes in Computer Science*, pages 127–139. Springer, 2017. doi:10.1007/978-3-319-68705-6_10.

5 Paul S. Bonsma, Marcin Kaminski, and Marcin Wrochna. Reconfiguring independent sets in claw-free graphs. In R. Ravi and Inge Li Gørtz, editors, *Algorithm Theory - SWAT 2014 - 14th Scandinavian Symposium and Workshops, Copenhagen, Denmark, July 2-4, 2014. Proceedings*, volume 8503 of *Lecture Notes in Computer Science*, pages 86–97. Springer, 2014. doi:10.1007/978-3-319-08404-6_8.

6 Nicolas Bousquet, Arnaud Mary, and Aline Parreau. Token jumping in minor-closed classes. In Ralf Klasing and Marc Zeitoun, editors, *Fundamentals of Computation Theory - 21st International Symposium, FCT 2017, Bordeaux, France, September 11-13, 2017, Proceedings*, volume 10472 of *Lecture Notes in Computer Science*, pages 136–149. Springer, 2017. doi:10.1007/978-3-662-55751-8_12.

7 Nicolas Bousquet, Amer E. Mouawad, Naomi Nishimura, and Sebastian Siebertz. A survey on the parameterized complexity of the independent set and (connected) dominating set reconfiguration problems. *CoRR*, abs/2204.10526, 2022. arXiv:2204.10526, doi:10.48550/arXiv.2204.10526.

8 Leizhen Cai, Siu Man Chan, and Siu On Chan. Random separation: A new method for solving fixed-cardinality optimization problems. In Hans L. Bodlaender and Michael A. Langston, editors, *Parameterized and Exact Computation, Second International Workshop, IWPEC 2006, Zürich, Switzerland, September 13-15, 2006, Proceedings*, volume 4169 of *Lecture Notes in Computer Science*, pages 239–250. Springer, 2006. doi:10.1007/11847250_22.

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

10 Erik D. Demaine, Martin L. Demaine, Eli Fox-Epstein, Duc A. Hoang, Takehiro Ito, Hirotaka Ono, Yota Otachi, Ryuhei Uehara, and Takeshi Yamada. Polynomial-time algorithm for sliding tokens on trees. In Hee-Kap Ahn and Chan-Su Shin, editors, *Algorithms and Computation - 25th International Symposium, ISAAC 2014, Jeonju, Korea, December 15-17, 2014, Proceedings*, volume 8889 of *Lecture Notes in Computer Science*, pages 389–400. Springer, 2014. doi:10.1007/978-3-319-13075-0_31.

11 Rodney G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness I: basic results. *SIAM J. Comput.*, 24(4):873–921, 1995. doi:10.1137/S0097539792228228.

12 Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer, 1999. doi:10.1007/978-1-4612-0515-9.

13 Eli Fox-Epstein, Duc A. Hoang, Yota Otachi, and Ryuhei Uehara. Sliding token on bipartite permutation graphs. In Khaled M. Elbassioni and Kazuhisa Makino, editors, *Algorithms and Computation - 26th International Symposium, ISAAC 2015, Nagoya, Japan, December 9-11, 2015, Proceedings*, volume 9472 of *Lecture Notes in Computer Science*, pages 237–247. Springer, 2015. doi:10.1007/978-3-662-48971-0_21.

14 Martin Grohe, Stephan Kreutzer, and Sebastian Siebertz. Deciding first-order properties of nowhere dense graphs. *J. ACM*, 64(3):17:1–17:32, 2017. doi:10.1145/3051095.

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

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

17 Takehiro Ito, Marcin Kaminski, Hirotaka Ono, Akira Suzuki, Ryuhei Uehara, and Katsuhisa Yamanaka. On the parameterized complexity for token jumping on graphs. In T. V. Gopal, Manindra Agrawal, Angsheng Li, and S. Barry Cooper, editors, *Theory and Applications of Models of Computation - 11th Annual Conference, TAMC 2014, Chennai, India, April 11-13, 2014. Proceedings*, volume 8402 of *Lecture Notes in Computer Science*, pages 341–351. Springer, 2014. doi:10.1007/978-3-319-06089-7_24.

18 Takehiro Ito and Yota Otachi. Reconfiguration of colorable sets in classes of perfect graphs. *Theor. Comput. Sci.*, 772:111–122, 2019. doi:10.1016/j.tcs.2018.11.024.

19 Marcin Kaminski, Paul Medvedev, and Martin Milanic. Complexity of independent set reconfigurability problems. *Theor. Comput. Sci.*, 439:9–15, 2012. doi:10.1016/j.tcs.2012.03.004.

20 Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972. doi:10.1007/978-1-4684-2001-2_9.

21 Stephan Kreutzer, Roman Rabinovich, and Sebastian Siebertz. Polynomial kernels and wideness properties of nowhere dense graph classes. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1533–1545. SIAM, 2017. doi:10.1137/1.9781611974782.100.

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

23 Daniel Lokshtanov, Amer E. Mouawad, Fahad Panolan, M. S. Ramanujan, and Saket Saurabh. Reconfiguration on sparse graphs. *J. Comput. Syst. Sci.*, 95:122–131, 2018. doi:10.1016/j.jcss.2018.02.004.

24 Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, Roohani Sharma, and Meirav Zehavi. Covering small independent sets and separators with applications to parameterized algorithms. *ACM Trans. Algorithms*, 16(3):32:1–32:31, 2020. doi:10.1145/3379698.

25 Amer E. Mouawad, Naomi Nishimura, Venkatesh Raman, and Sebastian Siebertz. Vertex cover reconfiguration and beyond. *Algorithms*, 11(2):20, 2018. doi:10.3390/a11020020.

26 Jaroslav Nesetril and Patrice Ossona de Mendez. Grad and classes with bounded expansion i. decompositions. *Eur. J. Comb.*, 29(3):760–776, 2008. doi:10.1016/j.ejc.2006.07.013.

27 Jaroslav Nesetril and Patrice Ossona de Mendez. On nowhere dense graphs. *Eur. J. Comb.*, 32(4):600–617, 2011. doi:10.1016/j.ejc.2011.01.006.

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

29 Geevarghese Philip, Venkatesh Raman, and Somnath Sikdar. Polynomial kernels for dominating set in graphs of bounded degeneracy and beyond. *ACM Trans. Algorithms*, 9(1):11:1–11:23, 2012. doi:10.1145/2390176.2390187.

30 Jan Arne Telle and Yngve Villanger. FPT algorithms for domination in biclique-free graphs. In Leah Epstein and Paolo Ferragina, editors, *Algorithms - ESA 2012 - 20th Annual European Symposium, Ljubljana, Slovenia, September 10-12, 2012. Proceedings*, volume 7501 of *Lecture Notes in Computer Science*, pages 802–812. Springer, 2012. doi:10.1007/978-3-642-33090-2_69.

31 Jan van den Heuvel. The complexity of change. In Simon R. Blackburn, Stefanie Gerke, and Mark Wildon, editors, *Surveys in Combinatorics 2013*, volume 409 of *London Mathematical Society Lecture Note Series*, pages 127–160. Cambridge University Press, 2013. doi:10.1017/CBO9781139506748.005.

32 Tom C. van der Zanden. Parameterized complexity of graph constraint logic. In Thore Husfeldt and Iyad A. Kanj, editors, *10th International Symposium on Parameterized and Exact Computation, IPEC 2015, September 16-18, 2015, Patras, Greece*, volume 43 of *LIPIcs*, pages 282–293. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPIcs.IPEC.2015.282.

33 Marcin Wrochna. Reconfiguration in bounded bandwidth and tree-depth. *J. Comput. Syst. Sci.*, 93:1–10, 2018. doi:10.1016/j.jcss.2017.11.003.

34 Takeshi Yamada and Ryuhei Uehara. Shortest reconfiguration of sliding tokens on subclasses of interval graphs. *Theor. Comput. Sci.*, 863:53–68, 2021. doi:10.1016/j.tcs.2021.02.019.