Independent-Set Reconfiguration Thresholds of Hereditary Graph Classes

Mark de Berg[1], Bart M. P. Jansen[2], and Debankur Mukherjee[3]

1 **Eindhoven University of Technology, Department of Computer Science, Eindhoven, The Netherlands**
`m.t.d.berg@tue.nl`
2 **Eindhoven University of Technology, Department of Computer Science, Eindhoven, The Netherlands**
`b.m.p.jansen@tue.nl`
3 **Eindhoven University of Technology, Department of Mathematics, Eindhoven, The Netherlands**
`d.mukherjee@tue.nl`

Abstract

Traditionally, reconfiguration problems ask the question whether a given solution of an optimization problem can be transformed to a target solution in a sequence of small steps that preserve feasibility of the intermediate solutions. In this paper, rather than asking this question from an algorithmic perspective, we analyze the combinatorial structure behind it. We consider the problem of reconfiguring one independent set into another, using two different processes: (1) exchanging exactly k vertices in each step, or (2) removing or adding one vertex in each step while ensuring the intermediate sets contain at most k fewer vertices than the initial solution. We are interested in determining the minimum value of k for which this reconfiguration is possible, and bound these threshold values in terms of several structural graph parameters. For hereditary graph classes we identify structures that cause the reconfiguration threshold to be large.

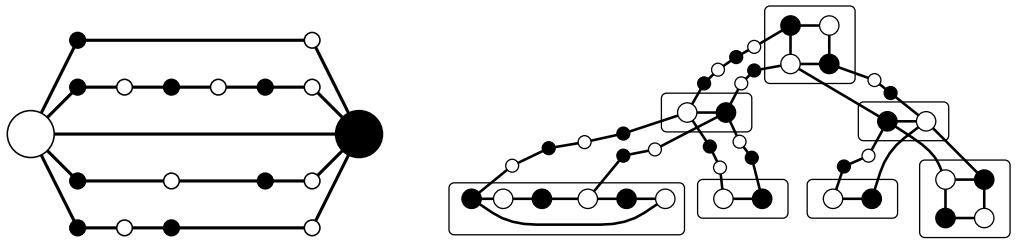
1998 ACM Subject Classification G.2.1 Combinatorics

Keywords and phrases reconfiguration, independent set, Token Addition Removal, Token Sliding

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2016.34

1 Introduction

Over the past decade, reconfiguration problems have drawn a lot of attention of researchers in algorithms and combinatorics [4, 5, 9, 13, 15, 16, 18, 22, 24]. In this framework, one asks the following question: Given two solutions I, J of a fixed optimization problem, can I be transformed into J by a sequence of small steps that maintain feasibility for all intermediate solutions? Such problems are practically motivated by the fact it may be impossible to adapt a new production strategy instantaneously if it differs too much from the strategy that is currently in use; changes have to be made in small steps, but production has to keep running throughout. From a theoretical perspective, the study of reconfiguration problems provides deep insights into the structure of the solution space. One of the well-studied examples is when the solution space consists of all the independent sets of a graph (optionally all having a prescribed size). In this case, three types of reconfiguration rules have been considered. These are naturally explained using *tokens* on vertices of the graph. In *Token Addition Removal* (TAR) [16, 22], there is a token on every vertex of the initial independent set, and there is a buffer of tokens, initially empty. A step consists of removing a token from a vertex



(a) A pumpkin of size 18.

(b) A graph of treewidth two with a complete binary tree T of depth two as a bipartite topological double minor.

Figure 1 The bipartite structures responsible for large MTJ and TAR reconfiguration thresholds, respectively. A *pumpkin* consists of odd-length vertex-disjoint paths between two vertices. The special form of topological *minor* represents each vertex of the tree T by an edge or even cycle in G, and each edge of T by two odd-length paths connecting vertices in opposite partite sets in G.

and placing it in the buffer, or placing a buffer token onto a vertex of the graph. The set of vertices with tokens must form an independent set at all times, and the goal is to move the tokens from the initial to the target independent set while ensuring the buffer size never exceeds a given threshold. In *Token Sliding* (TS) [18, 15], a step consists of replacing one vertex v in the independent set by a neighbor of v (the token slides along an edge). In *Token Jumping* (TJ) [18] a step also consists of replacing a single vertex, but the newly added vertex need not have any neighboring relation with the replaced vertex (the token jumps). Token jumping reconfiguration is equivalent to TAR reconfiguration with a buffer of size one.

These models have been analyzed in detail in the recent literature on algorithms [4, 5, 9, 13, 14, 21], complexity theory [15, 16, 18, 22], combinatorics [6, 12], and even statistical physics [17, 19, 23]. It is known that the reconfiguration problem under all the above three rules is PSPACE-complete for general graphs, perfect graphs, and planar graphs [15, 18, 16]. The TJ and TAR reconfiguration problems are PSPACE-complete even for bounded bandwidth graphs [24]. Further analyses on the complexity can be found in [4, 5, 9, 13, 21]. The constrained token-moving problems are related to pebbling games that have been studied in the literature, with applications to robot motion planning [1, 6, 12, 14].

As mentioned, the goal in reconfiguring independent sets is to go from one given independent I to another one J by a sequence of small steps. In the TS and TJ models, a step involves moving a single token. This is ideal, but unfortunately reconfiguration is often impossible in the TS or TJ model. Reconfiguration in the TAR model is always possible if one makes the buffer size sufficiently large. However, a large buffer size is undesirable. We are interested in determining the minimum buffer size that is sufficient to ensure any independent set in a given graph G can be reconfigured to any target independent set of the same size. We call this minimum the TAR *reconfiguration threshold* (precise definitions in Section 2). Our aim is to bound the threshold in terms of properties of the graph, and to identify the structures contained in hereditary graph classes that cause large thresholds. We also generalize the TJ model to *Multiple Token Jumping* (MTJ), where in each step a prescribed number of tokens may be moved simultaneously. In the MTJ model, the question becomes: What is the minimum number of simultaneously jumping tokens needed to ensure any reconfiguration is possible? This quantity is called the MTJ *reconfiguration threshold*.

Our contribution. We provide upper and lower bounds on the MTJ and TAR reconfiguration thresholds in terms of several graph parameters. Our bounds apply to the reconfiguration

thresholds of hereditary *graph classes*. The threshold of a graph class is the supremum of the threshold values of the graphs in that class: it is the smallest value k such that for any graph G in the class, any source independent set I in G can be reconfigured into any target independent set J using steps of size k (for MTJ) or a buffer of size k (for TAR).

The MTJ reconfiguration threshold of graphs that are structurally very simple, may nevertheless be very large. For example, an even cycle with $2n$ vertices can be partitioned into two independent sets I and J of size n each. Any MTJ reconfiguration of I into J requires a jump of n vertices, and this is trivially sufficient. Since a cycle has a feedback vertex set (FVS, see Section 2) of size one, the MTJ threshold cannot be bounded in terms of the size of a minimum feedback vertex set. However, we prove that the threshold is upper-bounded by the size of a minimum vertex cover of G. Although this bound is tight in the worst case, there are many graph classes with a small MTJ threshold even though they require a large vertex cover. Trees for example have MTJ threshold at most one. We therefore introduce the notion of *pumpkin*, which consists of two nodes connected by at least two vertex-disjoint paths of odd length (Figure 1a). The *size* of a pumpkin is its total number of vertices. We characterize the MTJ reconfiguration threshold of a hereditary graph class Π in terms of the size of the largest pumpkin it contains: the MTJ reconfiguration threshold is upper- and lower-bounded in terms of the largest pumpkin contained in a bipartite graph in Π.

TAR reconfiguration is more versatile than MTJ reconfiguration. In the concrete example of a $2n$-cycle discussed above, its MTJ threshold is n while any pair of independent sets can be reconfigured in the TAR model using a buffer of size two. Moreover, we show that any graph that has a feedback vertex set of size k has TAR reconfiguration threshold at most $k + 1$, and reconfiguring one side of the complete bipartite graph $K_{n,n}$ to the other side shows that this is tight. Our main result concerning TAR reconfiguration states that the TAR reconfiguration threshold of any graph is upper-bounded by its pathwidth. Somewhat surprisingly, there are graphs of constant treewidth (treewidth 2 suffices) for which the TAR reconfiguration threshold is arbitrarily large. We also introduce the concept of *bipartite topological double minor* (BTD-minor), see Figure 1b, and show using an isoperimetric inequality that any hereditary graph class containing a graph having a complete binary tree of depth d as a BTD-minor, has TAR reconfiguration threshold $\Omega(d)$. We conjecture that the TAR reconfiguration threshold can also be upper-bounded in terms of the depth of the largest complete binary tree BTD-minor, but we have not been able to prove this (see Section 6).

Applications. The MTJ and TAR reconfiguration thresholds play an important role in statistical physics and wireless communication networks. To understand the importance of the TAR reconfiguration threshold, consider the following process: In a graph G, nodes are trying to become *active* (transmit information) at some rate, independently of each other in a distributed manner. When a potential activation occurs at a node, it can only become active if none of its neighboring nodes are active at that moment (otherwise the transmissions would interfere). An active node deactivates at some rate independent of the other processes. At any point in time, the set of active nodes in this process forms an independent set of the graph. In statistical physics, this process is known as Glauber dynamics with *hard-core interaction*. This activity process on graphs has applications in various fields of study. Loosely speaking, when the activation rate is large, in the long run the above process always tries to stay in a maximum independent set. For graphs with more than one maximum independent set, it is interesting to study the time this process takes to reach a target independent set, starting from some specific independent set. This time depends crucially upon what we call the TAR reconfiguration threshold of the underlying

graph [23]. In particular, the mixing time of the Glauber dynamics on a graph increases exponentially with its TAR reconfiguration threshold, and hence the Glauber dynamics on the graph is fast mixing if and only if the TAR reconfiguration threshold is small.

2 Preliminaries

In this section we give the most important graph-theoretic definitions. Notions not defined here can be found in standard textbooks [7, 10]. Due to space restriction, proofs of statements marked (★) have been omitted; they can be found in the full version of the paper [8].

A graph is a pair $G = (V, E)$, where V is the set of vertices, and E is the set of edges. We also use $V(G)$ and $E(G)$ to refer to the vertex and edge set of G, when convenient. All graphs we consider are finite, simple, and undirected. For $U \subseteq V$ we denote by $G - U$ the graph obtained from G by removing the vertices in U and their incident edges. A set $U \subseteq V$ is an *independent set* of G if $\{u, v\} \notin E$ for any $u, v \in U$. The *symmetric difference* of two sets U and U' is $U \Delta U' := (U_1 \setminus U_2) \cup (U_2 \setminus U_1)$. A set $U \subseteq V$ is a *vertex cover* of G if every edge in E is incident with a vertex in U. The minimum cardinality of a vertex cover of G is denoted by $\textsc{vc}(G)$. A set $U \subseteq V$ is a *feedback vertex set* if $G - U$ is acyclic (a *forest*). The minimum cardinality of a feedback vertex set of G is denoted $\textsc{fvs}(G)$. For a vertex v, denote by $N_G(v)$ the set of its neighbors (excluding v itself). The *neighborhood* of a set $U \subseteq V$ is $N_G(U) := \bigcup_{s \in U} N_G(s) \setminus U$. We omit the subscript when it is clear from the context. A graph $G' = (V', E')$ is said to be a *subgraph* of G, if $V' \subseteq V$, and $E' \subseteq E$. It is an *induced subgraph* of G if $V' \subseteq V$ and for any $u, v \in V'$ we have $\{u, v\} \in E$ if and only if $\{u, v\} \in E'$. The subgraph of G induced by $U \subseteq V$ is denoted $G[U]$. A *graph class* is a (possibly infinite) collection of graphs. A graph class Π is said to be *hereditary* if given any graph $G \in \Pi$, any induced subgraph of G belongs to the class Π as well. A graph is *bipartite* if its vertex set can be partitioned into two independent sets I and J, which are also called the *partite sets*. We sometimes denote such a bipartite graph by $G = (I \cup J, E)$. A bipartite graph is *balanced* if $|I| = |J|$. A *matching* is a set of edges that do not share any endpoints. A matching is *perfect* if it spans the entire vertex set. A vertex v is a *cutvertex* in graph G if the removal of v increases the number of connected components. A *biconnected component* of G is a maximal connected subgraph of G that does not contain a cutvertex: removal of a single vertex from a biconnected component leaves the component connected.

We use the definitions of (nice) path decompositions as given in [7, §7.2]. For any path decomposition $\mathcal{P} = (X_1, X_2, \ldots, X_r)$ of $G = (V, E)$, and any vertex $v \in V$, define $l_{\mathcal{P}}(v) = \min\{i : v \in X_i\}$ and $r_{\mathcal{P}}(v) = \max\{i : v \in X_i\}$, that is, $l_{\mathcal{P}}(v)$ and $r_{\mathcal{P}}(v)$ respectively denote the index of the first and last bag containing v. Note that if $\mathcal{P}$ is nice, then $l_{\mathcal{P}}(\cdot)$ and $r_{\mathcal{P}}(\cdot)$ are injective maps over the set of vertices.

3 Definitions and Basic Facts for Reconfiguration

Multiple Token Jump (MTJ). Given any two independent sets I and J, with $|I| = |J|$, we say that I can be *k-MTJ reconfigured* to J, if there exists a finite sequence of independent sets $(I = W_0, W_1, W_2, \ldots, W_n, W_{n+1} = J)$ for some $n \geq 0$, such that for all $i \in \{0, \ldots, n+1\}$ the set W_i is an independent set, $|W_i| = |I| = |J|$, and $|W_{i+1} \setminus W_i| \leq k$. A step $W_i \to W_{i+1}$ in the reconfiguration process with $|W_i \setminus W_{i+1}| = k$ is called a k-TJ move. Given a graph $G = (V, E)$, define $\textsc{mtj}(G, s)$ as the minimum value of k such that any two independent sets of size s in G can be k-MTJ reconfigured to each other. Now define $\textsc{mtj}(G) := \max_{1 \leq s \leq |V|} \textsc{mtj}(G, s)$. Our goal is to characterize the value of $\textsc{mtj}(G)$ in terms of certain parameters of the graph G.

We call $\text{MTJ}(G)$ the *MTJ reconfiguration threshold* of the graph G. The MTJ reconfiguration threshold of a graph class Π is defined as $\text{MTJ}(\Pi) := \sup_{G\in\Pi} \text{MTJ}(G)$.

Token Addition Removal (TAR). Given any two independent sets I and J, with $|I| = |J|$, we say that I can be *k-TAR reconfigured* to J, if there exists a finite sequence of independent sets $(I = W_0, W_1, W_2, \ldots, W_n, W_{n+1} = J)$ for some $n \geq 0$, such that W_i is an independent set, $|I| - |W_i| \leq k$, and $|W_{i-1} \Delta W_i| \leq 1$ for all $i \in \{0, \ldots, n+1\}$. We refer to the quantity $B_i := |I| - |W_i|$ as the *buffer size* at step i: the tokens that were on the initial independent set and are not on the current independent set W_i, are placed in the buffer. Define $\text{TAR}(G, s)$ to be the smallest buffer size k such that any two independent sets of size s can be k-TAR reconfigured to each other. Define $\text{TAR}(G) := \max_{1\leq s\leq |V|} \text{TAR}(G, s)$. As before, we call $\text{TAR}(G)$ the *TAR reconfiguration threshold* of the graph G, and extend the terminology to graph classes Π by defining $\text{TAR}(\Pi) := \sup_{G\in\Pi} \text{TAR}(G)$.

Facts on Reconfiguration. Observe that for any graph G, it holds that $\text{MTJ}(G) = 1$ if and only if $\text{TAR}(G) = 1$. In general, the TAR reconfiguration threshold is at most the MTJ reconfiguration threshold. Indeed, each k-TJ move can be thought of as a sequence of $2k$ steps with maximum buffer size k. First, sequentially remove the tokens of the k vertices from which we are jumping, placing their tokens in the buffer; then sequentially place the buffer tokens on the k new vertices in the independent set.

▶ **Proposition 1 (★).** *Let G be a graph with independent sets I and J of equal size. If $I \setminus J$ can be k-TAR reconfigured (resp. k-MTJ reconfigured) to $J \setminus I$ in the graph $G[I\Delta J]$, then I can be k-TAR reconfigured (resp. k-MTJ reconfigured) to J in G.*

Proposition 1 shows that to upper-bound the TAR or MTJ reconfiguration threshold, it suffices to do so in balanced bipartite graphs where the source and target configurations are disjoint; note that $G[I\Delta J]$ is balanced bipartite and $I \setminus J$ and $J \setminus I$ are disjoint. We will frequently exploit this in our proofs. For any graph class Π, let Π_{bip} denote the set of bipartite graphs in Π. The following proposition shows that the reconfiguration threshold of a hereditary graph class is determined by the behavior of the bipartite graphs in the class. Note that for hereditary classes Π, the class Π_{bip} is hereditary as well.

▶ **Proposition 2 (★).** *For any hereditary graph class Π, we have $\text{MTJ}(\Pi) = \text{MTJ}(\Pi_{\text{bip}})$ and $\text{TAR}(\Pi) = \text{TAR}(\Pi_{\text{bip}})$.*

4 Thresholds for Multiple Token Jump Reconfiguration

We start our discussion of token jump reconfiguration by recalling the following known result.

▶ **Theorem 3** ([18, Theorem 7]). *Let the graph $G = (V, E)$ be a forest. Then $\text{MTJ}(G) \leq 1$.*

The intuition behind this result is that since a forest does not contain any cycle, one can start reconfiguring from the leaf nodes or the isolated vertices, each of which has at most one neighbor from the target configuration. For arbitrary graphs, the above procedure does not work since there may not be any leaves or isolated vertices. But if a graph G has a small vertex cover, then its MTJ reconfiguration threshold is again small.

▶ **Theorem 4 (★).** *Let $G = (V, E)$ be a graph. Then $\text{MTJ}(G) \leq \max(\text{VC}(G), 1)$.*

An even cycle of length $2n$ has MTJ reconfiguration threshold n. Since its vertex cover number is n, Theorem 4 is best-possible. Long cycles are not the only graphs whose MTJ reconfiguration threshold equals half the size of the vertex set. Bistable graphs (introduced below), of which the pumpkin structure defined in the introduction is a special case, also have this property. We bound the MTJ reconfiguration threshold of any graph G, in terms of the size of the largest induced bistable subgraph. The resulting bounds on the MTJ reconfiguration threshold are tight, but can be hard to apply to specific graph classes: it may be difficult to estimate the size of the largest induced bistable graph, or even to determine whether a given graph is bistable or not. We will therefore relate the size of the largest induced bistable subgraph to the size of the largest pumpkin subgraph. This will result in upper- and lower bounds on the MTJ reconfiguration threshold in terms of the largest pumpkin structure contained in the graph (class), which is arguably a more insightful parameter. The resulting bound will not be best-possible, however.

▶ **Definition 5** (Bistable graphs)**.** A graph is called *bistable* if it is connected, bipartite, and has exactly two distinct maximum independent sets formed by the two partite sets in its unique bipartition. The *rank* of a bistable graph is defined as the size of its maximum independent sets.

Let $\textsc{bi}(G)$ denote the rank of the largest induced bistable subgraph of G. If G contains no induced bistable subgraphs (which can only occur if G has no edges), then we define $\textsc{bi}(G)$ to be one. For a graph class Π we define $\textsc{bi}(\Pi) := \sup_{G\in\Pi} \textsc{bi}(G)$.

The pumpkin shown in Figure 1a forms an example of a bistable graph. Lemma 6 connects bistable graphs to independent-set reconfiguration. Consider the task of reconfiguring the J-partite set to the I-partite set in a balanced bipartite graph $G = (I \cup J, E)$. If we have a set $S \subseteq I$ such that $|S| \geq |N(S)|$, then one way to make progress in the reconfiguration is to select $|S|$ vertices from $N(S) \subseteq J$ and jump their tokens onto the vertices in S, resulting in a new independent set of the same size. The following lemma shows that when we consider a set S that is *minimal* with respect to being at least as large as its neighborhood, then the induced subgraph $G[N[S]]$ is bistable. Hence the cost of such a jump of $|S|$ vertices is bounded by $\textsc{bi}(G)$, which will allow us to bound the MTJ reconfiguration threshold.

▶ **Lemma 6 (★).** *Let $G = (I \cup J, E)$ be a balanced bipartite graph without isolated vertices and let $S \subseteq I$ be inclusion-wise minimal with the properties that $|S| \geq |N(S)|$ and S is not empty. Then $G[N[S]]$ is bistable.*

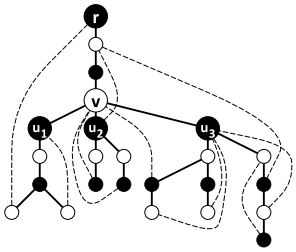
The next lemma states two key properties of bistable graphs. They will later be useful to relate the quantities $\textsc{pum}(G)$ and $\textsc{bi}(G)$.

▶ **Lemma 7 (★).** *Let $G = (I \cup J, E)$ be a bistable graph. Then the following holds:*
1. *G has a perfect matching covering I (and hence J).*
2. *G is biconnected.*

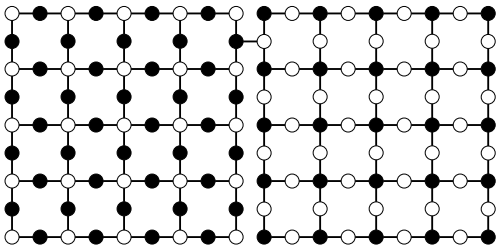
▶ **Theorem 8 (★).** *For any graph G it holds that $\textsc{mtj}(G) \leq \textsc{bi}(G)$. Moreover, if $G \neq K_1$, then there exists an induced subgraph G' of G with $\textsc{mtj}(G') \geq \textsc{bi}(G) \geq \textsc{bi}(G')$.*

The proof for the lower bound is straightforward, since by definition any induced bistable subgraph contains exactly two maximum independent sets. The upper bound follows by induction on the number of vertices in G, where in the induction step we make use of Lemma 6, and reconfigure the subgraph induced by the set $N[S]$.

The following corollary characterizes the MTJ reconfiguration threshold of hereditary graph classes. It follows easily from Theorem 8.



(a) DFS tree of a biconnected bipartite graph.



(b) Grid-like balanced bipartite graph with large treewidth and small TAR reconfiguration threshold.

Figure 2 (2a) Depth-first search tree of a bipartite biconnected graph. Tree-edges are drawn solid, while the remaining edges of G are drawn with dotted lines. The three children u_1, u_2, u_3 of v induce subtrees of types A, B, and C, respectively. (2b) Template for constructing graphs of large treewidth that can be TAR reconfigured with a buffer of size two. The treewidth is large due to the presence of a large grid minor.

▶ **Corollary 9 (★).** *For any hereditary graph class $\Pi \neq \{K_1\}$ it holds that* $\text{MTJ}(\Pi) = \text{BI}(\Pi)$.

We now formally introduce the pumpkin structure described in the introduction.

▶ **Definition 10** (Pumpkin)**.** A *pumpkin* is a graph consisting of two terminal vertices u and v linked by two or more vertex-disjoint paths with an odd number of edges, having no edges or vertices other than those on the paths. A path can consist of the single edge $\{u, v\}$. The *size* of the pumpkin is the total number of vertices.

For a graph G we denote by $\text{PUM}(G)$ the size of the largest (not necessarily induced) subgraph isomorphic to a pumpkin that is contained in G, or zero if G contains no pumpkin. For a graph class Π we define $\text{PUM}(\Pi) := \sup_{G \in \Pi} \text{PUM}(G)$.

The next theorem shows that the rank of the largest bistable induced subgraph of G can be upper-bounded in terms of the size of G's largest pumpkin subgraph.

▶ **Theorem 11 (★).** *For any bistable graph G we have $\text{BI}(G) \leq f(\text{PUM}(G))$, where the function f is defined as $f(k) = (k^3 + k^2)^{k^2+1} + 1$.*

Proof sketch. A bistable graph G is biconnected (Lemma 7). Biconnected graphs with a path of length more than L^2 have a cycle of length more than L [11], which forms a pumpkin since G is bipartite. So if we set $L := \text{PUM}(G)$, graph G cannot have a path of length more than L^2. Consequently, if we build a DFS tree T rooted at an arbitrary vertex, its depth will be at most L^2. Next we claim that each vertex v in T has at most $L^3 + L^2$ children, which in conjunction with the above bound on depth, yields the theorem. To prove the claim, we classify each child u of v into one of three types (Figure 2a) and prove that no type occurs often.

Type A: Some vertex in the subtree T_u rooted at u, has an edge in G to an ancestor w of v that does not belong to v's partite set. Through each such subtree, we obtain an odd-length path from v to w. If there are more than L such paths from v to w then they form a pumpkin of size more than L, which gives a contradiction. Hence each ancestor of v receives an edge from less than L type-A child trees. There are $\leq L^2$ ancestors, hence $\leq L^3$ type-A children.

Type B: Vertex u is not of type A and the vertices in the subtree T_u are not evenly balanced over the two partite sets. Then any perfect matching in G (which exists by Lemma 7) matches

a vertex in T_u to an ancestor of v. Since the depth is at most L^2, while matching partners are all distinct, there are at most L^2 type-B children.

Type C: Vertex u is not of type A and the vertices in the subtree T_u are evenly balanced over the two partite sets. If such a child exists, then we can build a maximum independent set that is not equal to either partite set in G: take the partite set that does not contain v, but replace its contents within T_u by the vertices from T_u in the other partite set. The case distinction ensures the result is a maximum independent set, contradicting bistability of G.

As this bounds the number of children of a vertex by $L^3 + L^2$, the theorem follows. ◀

The following theorem is our main result on the MTJ reconfiguration threshold. It bounds the MTJ reconfiguration threshold of a hereditary graph class Π in terms of the maximum size of a pumpkin subgraph of a graph in Π_{bip}, by combining Theorems 8 and 11. Recall that Π_{bip} contains the bipartite graphs in Π.

▶ **Theorem 12 (★).** *For any hereditary graph class Π, the following holds:*

$$g_1(\text{PUM}(\Pi_{\text{bip}})) \leq \text{MTJ}(\Pi) \leq g_2(\text{PUM}(\Pi_{\text{bip}})), \tag{1}$$

where $g_1,\ g_2 : \mathbb{N} \to \mathbb{N}$ are positive non-decreasing functions defined as $g_1(k) = k/2$ and $g_2(k) = (k^3 + k^2)^{k^2+1} + 1$. Moreover, for every graph G we have $\text{MTJ}(G) \leq g_2(\text{PUM}(G))$.

While the upper bound of Theorem 12 has room for improvement, the following proposition shows that the exponential dependency on the pumpkin size in the upper bound is unavoidable.

▶ **Proposition 13 (★).** *Let $\Pi_{\text{PUM}}(k) := \{G : \text{PUM}(G) \leq k\}$ be the class of all graphs G whose largest pumpkin subgraph has size at most k. Then $\text{MTJ}(\Pi_{\text{PUM}}(k)) = 2^{\Omega(k)}$.*

5 Thresholds for Token Addition Removal Reconfiguration

In this section we study the model of token addition removal. First observe that when G is a forest, we have $\text{MTJ}(G) \leq 1$ and therefore $\text{TAR}(G) \leq 1$ as well. Also, from Theorem 4 we get $\text{TAR}(G) \leq \max(\text{VC}(G), 1)$. But the inequality $\text{TAR}(G) \leq \text{MTJ}(G)$ tells us nothing about the behavior of the TAR reconfiguration threshold when the MTJ reconfiguration threshold is large. The next simple proposition immediately points towards this direction. Indeed, a large pumpkin (which has large MTJ reconfiguration threshold) can have a small feedback vertex set; this happens for even cycles, for example.

▶ **Proposition 14 (★).** *Let $G = (V, E)$ be a graph. Then $\text{TAR}(G) \leq \text{FVS}(G) + 1$.*

The proof is fairly straightforward by noting that for any graph G, if the size of the minimum feedback vertex set is k, then by definition, deletion of k vertices leaves an acyclic subgraph. Hence, we can essentially apply Theorem 3. One can see that the above bound is tight, by considering the TAR reconfiguration threshold of a complete balanced bipartite graph. Indeed, for $K_{n,n}$ the minimum size of a feedback vertex set is $n - 1$, and one can see that in order to include any one of the vertices of the target independent set, the reconfiguration must pass through the empty set. This shows that the TAR reconfiguration threshold is also n. As the main result of this section, we will show that the TAR reconfiguration threshold of a graph is also bounded in terms of its *pathwidth*. Before proving that statement, we present a structural lemma about path decompositions that will be useful in the proof.

▶ **Lemma 15 (★).** *Let $G = (I \cup J, E)$ be a bipartite graph with a nice path decomposition $\mathcal{P} = (X_1, \ldots, X_r)$ of width k. Let $S \subseteq J$ such that $|N(S)| \leq |S|$ while no non-empty subset of S has this property. If we order the vertices in S as $i_1, \ldots, i_t$ such that $r_{\mathcal{P}}(i_1) < r_{\mathcal{P}}(i_2) < \ldots < r_{\mathcal{P}}(i_t)$, then $|N(\{i_1, \ldots, i_{t'}\})| < t' + k$ for all $1 \leq t' \leq t$.*

Intuitively, the lemma says the following. Suppose a set $S \subseteq J$ is inclusion-wise minimal with respect to being no smaller than its neighborhood. Then ordering S according to the right endpoints of the intervals representing S in the path decomposition, we are guaranteed that every prefix of S has a fairly small neighborhood compared to its size: the neighborhood size exceeds the size of the prefix by less than the pathwidth. Note that since the lemma deals with bipartite graphs only, no vertex of S can belong to the neighborhood of any prefix of S. The ordering of the vertices is uniquely defined since the path decomposition is nice. The bound of Lemma 15 is best-possible. Consider a complete bipartite graph $K_{n,n}$, with pathwidth n. In any optimal path decomposition, for $t' = 1$ the first vertex in the ordering has a neighborhood of size n and so $n < t' + n = 1 + n$, but a better bound is not possible. Using Lemma 15 we bound the TAR reconfiguration threshold in terms of pathwidth.

▶ **Theorem 16.** *Let $G = (V, E)$ be a graph. Then* $\text{TAR}(G) \leq \max(\text{PW}(G), 1)$.

Proof. We prove this theorem using induction on the number of vertices. By Proposition 1, it is enough to consider $G = (V, E)$ and assume that the initial independent set I and target independent set J are such that $|I| = |J|$, and $I \cup J = V$ and $I \cap J = \emptyset$. We will show that $\text{PW}(G) \leq k$ implies that $\text{TAR}(G) \leq k$, using induction on the number of vertices n. For $n = 1$, the statement is trivially true. Now fix any $k \geq 1$, and assume the induction hypothesis that any graph G with n vertices satisfying $\text{PW}(G) \leq k$ has $\text{TAR}(G) \leq k$.

Now let G be a graph of $n+1$ vertices having pathwidth at most k. Let S be an inclusion-minimal subset of J for which $|S| \geq |N(S)|$. Such a set exists since $|J| = |I| \geq |N(J)|$. We will show that if we reconfigure the set S in a suitable order by moving tokens from $N(S)$ onto S, then the buffer size will not grow beyond k. There are enough vertices in S to accommodate all tokens on $N(S)$, and afterward we will invoke induction.

We first deal with a special case. If $S = \{v\}$ is a singleton set, then it has degree at most one since $|S| \geq |N(S)|$. Move the token from the neighbor u of v (or from an arbitrary vertex u, if v has no neighbors) into the buffer, and then onto v. By induction there exists a TAR reconfiguration from $I \setminus \{u\}$ to $J \setminus \{v\}$ in $G - \{u, v\}$ using a buffer of size at most $\max(\text{PW}(G - \{u, v\}), 1) \leq \max(\text{PW}(G), 1)$. When inserting the token move from u onto v at the beginning of this sequence, we get a TAR reconfiguration from I to J with the desired buffer size. In the remainder of the proof we can therefore assume $|S| \geq 2$. This implies that $|S| = |N(S)|$: if $|S| > |N(S)|$ and $|S| \geq 2$, then we can remove a vertex v from S to obtain $|S \setminus \{v\}| \geq |N(S \setminus \{v\})|$ for the nonempty set $S \setminus \{v\}$, contradicting minimality.

Let $\mathcal{P} = (X_1, X_2, \ldots, X_r)$ be a nice path decomposition of width at most k. If G has no edges, then S is a singleton set containing an isolated vertex. Since we already covered that case, we know G has at least one edge, so any path decomposition has width $k \geq 1$. Enumerate the vertices of S as $i_1, \ldots, i_m$ such that $r_{\mathcal{P}}(i_1) < \ldots < r_{\mathcal{P}}(i_m)$. In other words, the vertices are ordered by increasing rightmost endpoint of the interval of bags containing it.

In order to describe the reconfiguration procedure we suitably group several TAR reconfiguration steps together as one step in the algorithm. In particular, one reconfiguration step in the algorithm described below will consist of a run of successive removals of nodes, followed by a single node addition.

We use the notion of a *buffer set* B_t at the t^{th} step of the reconfiguration, such that $|B_t|$ will correspond to the number of tokens in the buffer at any particular time, and $\max_t |B_t| + 1$ will correspond to the maximum buffer size of the corresponding TAR reconfiguration sequence. The buffer set is a subset of vertices, showing where the tokens in the buffer came from. At time step $t = 0$, define $W_0 = I$ to be the independent set of vertices with a token, and let the buffer set B_0 be empty. We will define intermediate independent sets W_i and buffer sets B_i representing the grouped reconfiguration steps. The algorithm stops

when W_m contains all vertices in S; we will then invoke the induction hypothesis to finish the sequence. From the sequence $(W_0, W_1, \ldots, W_m)$ one obtains a formal reconfiguration sequence as defined in Section 3 by inserting "transitioning independent sets" in between W_i and W_{i+1} for all i. From W_i, repeatedly remove one vertex until arriving at $W_{i+1} \setminus W_i$, and then add the single vertex of $W_{i+1} \setminus W_i$ to the resulting set.

For $t \geq 1$, the transition from $t-1$ to t is obtained as follows. Let u_t be an arbitrary vertex from $B_{t-1} \cup (N(i_t) \cap W_{t-1})$. Intuitively, at step t we take the token from u_t (in the buffer set or on a neighbor of i_t) and move it onto vertex i_t, causing u_t to disappear from the buffer and adding i_t to the independent set. To ensure the resulting set is independent, tokens on neighbors of i_t are moved into the buffer beforehand. Observe that the above step is valid only if $B_{t-1} \cup (N(i_t) \cap W_{t-1})$ is nonempty. Below in Claim 17 we show that due to the choice of S, this is indeed the case for all $t \leq m$. Formally, we obtain the following:

▶ **Algorithm** (Reconfiguring graphs with small pathwidth). *Initialize with $B_0 = \emptyset$ and $W_0 = I$. We now recursively define B_t and W_t for $t \geq 1$.*

1. *The neighbors of i_t that have tokens (that is, the neighbors that are in the current independent set) are removed from the previous independent set W_{t-1}, making room to add i_t to the new independent set: $W_t = (W_{t-1} \setminus N(i_t)) \cup \{i_t\}$.*
2. *The neighbors of i_t belonging to the previous independent set W_t move to the buffer, while u_t is removed from the buffer since its token has moved onto i_t:*

$$B_t = (B_{t-1} \cup (N(i_t) \cap W_{t-1})) \setminus \{u_t\}. \tag{2}$$

As mentioned earlier, a step from W_t to W_{t+1} can be thought as a sequence of successive removals of the nodes in $N(i_{t+1}) \cap W_t$, and then addition of the node i_{t+1}. During this successive TAR reconfiguration sequence corresponding to the step W_t to W_{t+1}, the maximum buffer size is given by $|B_{t+1}| + 1$, since the buffer size will be $|B_{t-1} \cup (N(i_t) \cap W_{t-1})|$ just before the buffer token from u_t is moved onto i_t. Therefore, the maximum buffer size in the entire TAR reconfiguration sequence starting from W_0 and ending at W_m is given by $\max_{0 \leq t \leq m} |B_t| + 1$. Also, at the end of the algorithm, all vertices from the set S will be in the independent set and no vertex in the buffer set. This can be seen as follows. Initially all tokens were on the vertices belonging to the set $N(S) \subseteq I$, since $S \subseteq J$. At each step of the algorithm, essentially one token is selected from $N(S)$ as long as the number of such tokens is positive, and it is placed on some vertex in S. Now since $|S| \geq |N(S)|$, all the tokens in $N(S)$ must eventually exhaust before the algorithm terminates placing one token at each vertex of S. For the validity of the above algorithm we claim the following, which in turn also characterizes the size of the buffer set at all intermediate time steps.

▶ **Claim 17.** *For all $1 \leq t \leq m$ we have that $B_{t-1} \cup (N(i_t) \cap W_{t-1})$ is nonempty, and that $|B_t| = |N(\{i_1, \ldots, i_t\})| - t$.*

Proof. Suppose for a contradiction that there exists $t' \leq m$, such that $B_{t'-1} \cup (N(i_{t'}) \cap W_{t'-1})$ is empty for the first time. If $t' = 1$, then $B_{t'-1} \cup (N(i_{t'}) \cap W_{t'-1})$ is empty, and in particular $N(i_{t'}) = \emptyset$, so that $i_{t'} = i_1$ is an isolated vertex. But since $|S| \geq 2$ by our argument above, it follows that $S' = \{i_1\}$ is a nonempty strict subset with $|S'| \geq |N(S')|$; a contradiction. So in the remainder we consider $t' > 1$. We show that, for all $t < t'$, $|B_t| = |N(\{i_1, \ldots, i_t\})| - t$. Using this, we prove that $2 \leq t' \leq m$ leads to a contradiction.

Observe that for any $t < t'$, after the t^{th} step of the algorithm, the total number of distinct vertices that have been added to the buffer set is given by $|N(\{i_1, \ldots, i_t\})|$. Furthermore, for all $t'' \leq t < t'$, the set $B_{t''-1} \cup (N(i_{t''}) \cap W_{t''-1})$ has always been nonempty. This implies that at each step, precisely one token has been removed from the buffer, thus reducing the

size of the buffer set by moving a buffer token onto a vertex that is added to the independent set. Therefore, in total t times the size of the buffer set reduces by one. Since initially the buffer set was empty, for any $t < t'$ we have $|B_t| = |N(\{i_1, \ldots, i_t\})| - t$.

Since we have assumed that $B_{t'-1} \cup (N(i_{t'}) \cap W_{t'-1})$ is empty, we know $B_{t'-1}$ is empty, and therefore from the above argument $|B_{t'-1}| = |N(\{i_1, \ldots, i_{t'-1}\})| - (t'-1) = 0$.

Defining $S' := \{i_1, \ldots, i_{t'-1}\} \subsetneq S$, we have $|N(S')| \leq |S'|$. Since $t' \geq 2$ the set S' is nonempty, contradicting the minimality of S. This proves the first part of the claim. Since the buffer does not become empty until after step t, the given argument then also proves the second part of the claim. ◀

Note that in particular $|B_m| = |N(\{i_1, \ldots, i_m\})| - m = |N(S)| - |S| = 0$; the buffer empties for the first time only after reconfiguring the whole set.

It remains to show that throughout the process the buffer size will not grow beyond k, i.e. $|B_t| \leq k-1$, for all $t \leq m$. Claim 17 (ii) implies that $\max_{t \leq m} |B_t| \geq k$ if and only if there is a t, with $t \leq m$, such that $|N(\{i_1, \ldots, i_t\})| - t \geq k$. But this is not possible due to Lemma 15. Hence, throughout the algorithm the buffer size will never exceed k.

Since the buffer set empties out after reconfiguring the set S, after the execution of the algorithm we have $W_m \cap J = S$ and $W_m \cap I \subset V \setminus (S \cup N(S))$. Now define $G' := G - (S \cup N(S))$, and $I' := I \cap W_m$ and $J' := J \setminus S$. Observe that G' has pathwidth at most k, and $|I'| = |I \cap W_m| = |I| - |S| = |J'|$. Furthermore, since S is non-empty, $|V(G')| \leq n$. By the induction hypothesis, there exists a TAR reconfiguration sequence from I' to J' in G' using a buffer of size at most k. Since $N(S)$ is not in G', any independent set in G' remains to be an independent set in G when augmented with the set S. Therefore we can first apply the given reconfiguration from $N(S)$ to S, followed by the reconfiguration from I' to J', to reconfigure I to J with a buffer of size at most k. ◀

Observe by considering a complete balanced bipartite graph on $2n$ vertices $K_{n,n}$, that in general the above bound is tight. Indeed, $K_{n,n}$ has pathwidth equal to n [3], and as explained earlier, the TAR reconfiguration threshold is also n. Having proved Theorem 16, it is natural to ask whether pathwidth in some sense characterizes the TAR reconfiguration threshold: does large pathwidth of a graph imply that its TAR reconfiguration threshold is large? This is not the case: the pathwidth of a complete binary tree is proportional to its depth [20], but its reconfiguration threshold is 1 by Theorem 3. We now identify a graph structure which forces the TAR reconfiguration threshold to be large. First we formally introduce the special type of minor, illustrated in Figure 1b.

▶ **Definition 18** (Bipartite topological double minor)**.** Let $G = (I \cup J, E)$ be a bipartite graph and let H be an arbitrary graph. Then H is a *bipartite topological double minor* of G, if one can assign to every $v \in V(H)$ a subgraph $\varphi(v)$ of G, which is either an edge or an even cycle in G, and one can assign to each edge $e = \{u, v\} \in E(H)$ a pair of odd-length paths $\psi_1(e)$, $\psi_2(e)$ in G, such that the following holds:

- For any $u, v \in V(H)$ with $u \neq v$ the subgraphs $\varphi(u)$ and $\varphi(v)$ are vertex-disjoint.
- For any $v \in V(H)$ no vertex of $\varphi(v)$ occurs as an interior vertex of a path $\psi_1(e)$ or $\psi_2(e)$, for any $e \in E(H)$.
- For any $e, e' \in E(H)$ the paths $\psi_1(e)$ and $\psi_2(e')$ are internally vertex-disjoint.
- For any $e = \{u, v\} \in E(H)$ the paths $\psi_1(e)$ and $\psi_2(e)$ both have one endpoint in $\varphi(v)$ and one endpoint in $\varphi(u)$.
- For any $v \in V(H)$ and edge $\{u, v\} \in E(H)$, the attachment points of $\psi_1(e)$ and $\psi_2(e)$ in $\varphi(v)$ belong to different partite sets.

The triple $(\varphi, \psi_1, \psi_2)$ is a *BTD-minor model* of H in G. For an edge $e \in E(H)$ we define $\psi_1'(e), \psi_2'(e) \subseteq V(G)$ as the *interior* vertices of the paths $\psi_1(e)$ and $\psi_2(e)$. Note that we can have $\psi_1'(e) = \emptyset$ (and, similarly, $\psi_2'(e) = \emptyset$) when $\psi_1(e)$ (resp. $\psi_2(e)$) consists of a single edge.

Intuitively, H occurs as a bipartite topological double minor (or *BTD-minor*) if each vertex of H can be realized by an edge or even cycle, and every edge of H can be realized by two odd-length paths that connect an I-vertex of $\varphi(v)$ to a J-vertex of $\varphi(u)$ and the other way around, in such a way that these structures are vertex-disjoint except for the attachment of paths to cycles. The definition easily extends to bipartite graphs whose bipartition is not given, since a BTD-minor is contained within a single connected component of the graph, which has a unique bipartition.

▶ **Proposition 19 (★).** *Let $G = (I \cup J, E)$ be a bipartite graph having a connected graph H as a BTD-minor model $(\varphi, \psi_1, \psi_2)$, such that each vertex of G is in the image of φ, ψ_1, or ψ_2. Then G has a perfect matching with $|I| = |J|$ edges, and for any independent set W in G:*
1. *For each vertex v of H we have $|W \cap \varphi(v)| \leq |\varphi(v)|/2$.*
2. *For each edge e of H and $i \in \{1, 2\}$ we have $|W \cap \psi_i'(e)| \leq |\psi_i'(e)|/2$.*
For a maximum *independent set W, equality holds in all cases.*

For a bipartite graph G, let $\textsc{treeminor}(G)$ denote the largest integer k for which G contains a complete binary tree of depth k as a BTD-minor. For a class Π of bipartite graphs we define $\textsc{treeminor}(\Pi) := \sup_{G \in \Pi} \textsc{treeminor}(G)$.

▶ **Theorem 20.** *There exists a real constant $c > 0$ such that any hereditary graph class Π satisfies $\textsc{tar}(\Pi) \geq c \cdot \textsc{treeminor}(\Pi_{\text{bip}})$.*

Proof. As before, we consider a balanced bipartite graph $G \in \Pi_{\text{bip}}$ with bipartition $V(G) = I \cup J$ that has a complete binary tree T of depth d as a BTD-minor. Since the graph class is hereditary, for the lower bound we consider only the subgraph of G induced by $\bigcup_{v \in V(T)} \{\varphi(v)\} \cup \left(\bigcup_{e \in E(T)} \{\psi_1(e) \cup \psi_2(e)\} \right)$. With a slight abuse of notation we denote this subgraph by G from now on.

▶ **Fact 21** ([2]). *There is a universal constant $c_1 > 0$ such that if T is a complete binary tree of depth d, then $\max_{1 \leq i \leq |V(T)|} \min_{S \subseteq V(T); |S| = i} |N_T(S)| \geq c_1 \cdot d$.*

The above implies that there exists $i_0 \leq |V(T)|$, such that any size-i_0 subset of $V(T)$ has a neighborhood of size at least $c_1 \cdot d$. Let $I \cup J$ be the unique bipartition of the connected graph G, and consider an arbitrary TAR reconfiguration sequence from I and J. In this sequence $(I = W_0, W_1, \ldots, W_t = J)$ of independent sets in G, look at the reconfiguration step when for the first time there exists $S \subseteq V(T)$ with $|S| = i_0$, such that the intermediate independent set W at that step contains $\bigcup_{v \in S} (\varphi(v) \cap J)$, and for all $v \notin S$ it satisfies $(\varphi(v) \cap W \cap J) \subsetneq (\varphi(v) \cap J)$. We will prove that $|J| - |W| \geq c_1 \cdot d$, implying that from the initial independent set of $|I| = |J|$ tokens, at least $c_1 \cdot d$ tokens must reside in the buffer.

To prove the theorem, consider the intermediate independent set W, and the set $S \subseteq V(T)$ with $|S| = i_0$ satisfying the above criteria. The following claim shows that for each vertex in $N_T(S)$, the independent set W uses at least one vertex fewer than the maximum independent set J does.

▶ **Claim 22.** *Consider an edge $e = \{u, v\} \in E(T)$ with $u \in S$ and $v \notin S$, and let $Q_{e,v} \subseteq V(G)$ denote the vertices in $\varphi(v) \cup \psi_1'(e) \cup \psi_2'(e)$. The following holds:*

$$|W \cap Q_{e,v}| < |J \cap Q_{e,v}| = \frac{|Q_{e,v}|}{2}. \tag{3}$$

Proof. By Proposition 19, the maximum independent set J contains exactly half the vertices of $Q_{e,v}$. If $|W \cap \psi'_i(e)| < |\psi'_i(e)|/2$ for some $i \in \{1,2\}$, then we are done: by Proposition 19 the set W contains fewer vertices from $\psi'_i(e)$ that the maximum independent set J does, and this cannot be compensated within the other parts of the structure since J contains half the vertices there and no independent set contains more. In the remainder, we can assume that W contains exactly half the vertices from $\psi'_1(e)$ and $\psi'_2(e)$. Then the following are true:

- **(i)** All J-nodes of $\varphi(u)$ are in W (by our choice of W and since $u \in S$).
- **(ii)** Some J-node of $\varphi(v)$ is not in W (by our choice of W and since $v \notin S$).
- **(iii)** Some I-node of $\varphi(v)$ is not in W. To see this, let $i \in \{1,2\}$ such that $\psi_i(e)$ is an odd-length path from a J-node in $\varphi(u)$ to an I-node in $\varphi(v)$, which exists by Definition 18, and orient it in that direction. Since the first vertex on the path is a J-node in $\varphi(u)$, it is contained in W as shown above. Hence the second vertex on the path, the first interior vertex, is not in W. Since exactly half the interior vertices from $\psi_i(e)$ belong to W, every other interior vertex from $\psi_i(e)$ is in W. Since the path has an even number of interior vertices and the first interior vertex is not in W, the last interior vertex must be in W. But this prevents its I-node neighbor in $\varphi(v)$ from being in W.

Therefore, since $\varphi(v)$ is either an edge or an even cycle, we have $|W \cap \varphi(v)| < |\varphi(v)|/2$ by observing the following: the only independent sets in $\varphi(v)$ of size $|\varphi(v)|/2$ are $\varphi(v) \cap I$ and $\varphi(v) \cap J$, but $\varphi(v) \cap W$ is not equal to either of these sets since it avoids a J-node and an I-node. Hence $|W \cap \varphi(v)| < |\varphi(v)|/2 = |J \cap \varphi(v)|$, and Proposition 19 shows that this cannot be compensated in other parts of the minor model, implying $|W \cap Q_{e,v}| < |J \cap Q_{e,v}|$. ◀

Using Claim 22 we now finish the proof of Theorem 20. For each $v \in N_T(S)$, pick an edge $e = \{u, v\}$ such that $u \in S$. By Claim 22 the set W contains less than half the vertices of $Q_{e,v}$, while the maximum independent set J contains exactly half. Note that the sets $Q_{e,v}$ considered for different vertices $v \in N_T(S)$ are disjoint, while Proposition 19 shows that from the other pieces of the minor model W cannot use more vertices than J does. It follows that $|W| \leq |J| - |N_T(S)| \leq |J| - c_1 \cdot d$. Hence the buffer contains at least $c_1 \cdot d$ tokens. ◀

6 Conclusion

We considered two types of reconfiguration rules for independent set, involving simultaneously jumping tokens and reconfiguration with a buffer. For both models, we derived tight bounds on their reconfiguration thresholds in terms of several graph parameters like the minimum vertex cover size, the minimum feedback vertex set size, and the pathwidth. Many results in the literature concerning the parameter pathwidth can be extended to hold for the parameter treewidth as well. This is not the case here; the upper bound on the TAR reconfiguration threshold in terms of pathwidth (Theorem 16) cannot be strengthened to treewidth, since one can make arbitrarily deep complete binary trees as BTD-minors in bipartite graphs of treewidth only two (see Figure 1b). On the other hand, there are bipartite graphs of large treewidth with TAR reconfiguration threshold two (Figure 2b). To characterize the TAR reconfiguration threshold one therefore needs to combine graph connectivity (as measured by the width parameters) with notions that constrain the parity of the connections in the graph. This is precisely why we introduced BTD-minors. We conjecture that the converse of Theorem 20 holds, in the sense that any hereditary graph class having a large TAR reconfiguration threshold must contain a graph having a complete binary tree of large depth as a BTD-minor. Our belief is based partially on the fact that a BTD-minor model of a deep

complete binary tree is arguably the simplest graph of large pathwidth and feedback vertex number. Resolving this conjecture is our main open problem.

References

1 Aviv Adler, Mark de Berg, Dan Halperin, and Kiril Solovey. Efficient multi-robot motion planning for unlabeled discs in simple polygons. In *Algorithmic Foundations of Robotics XI: Selected Contributions of the Eleventh International Workshop on the Algorithmic Foundations of Robotics*, pages 1–17, 2015. doi:10.1007/978-3-319-16595-0_1.

2 B. V. Subramanya Bharadwaj and L. Sunil Chandran. Bounds on isoperimetric values of trees. *Discrete Mathematics*, 309(4):834–842, 2009. doi:10.1016/j.disc.2008.01.021.

3 Hans L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209(1-2):1–45, 1998. doi:10.1016/S0304-3975(97)00228-4.

4 Paul Bonsma. Independent set reconfiguration in cographs. In *Graph-Theoretic Concepts in Computer Science: 40th International Workshop, WG 2014, Nouan-le-Fuzelier, France, June 25-27, 2014. Revised Selected Papers*, pages 105–116, 2014. doi:10.1007/978-3-319-12340-0_9.

5 Paul Bonsma, Marcin Kamiński, and Marcin Wrochna. Reconfiguring independent sets in claw-free graphs. In *Algorithm Theory – SWAT 2014: 14th Scandinavian Symposium and Workshops, Copenhagen, Denmark, July 2-4, 2014. Proceedings*, pages 86–97, 2014. doi:10.1007/978-3-319-08404-6_8.

6 Gruia Calinescu, Adrian Dumitrescu, and János Pach. Reconfigurations in graphs and grids. In *LATIN 2006: Theoretical Informatics: 7th Latin American Symposium, Valdivia, Chile, March 20-24, 2006. Proceedings*, pages 262–273, 2006. doi:10.1007/11682462_27.

7 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Daniel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized algorithms*. Springer Publishing Company, Incorporated, 2015. doi:10.1007/978-3-319-21275-3.

8 Mark de Berg, Bart M. P. Jansen, and Debankur Mukherjee. Independent set reconfiguration thresholds of hereditary graph classes. *arXiv:1610.03766*, 2016. arXiv:1610.03766.

9 Erik D. Demaine, Martin L. Demaine, Eli Fox-Epstein, Duc A. Hoang, Takehiro Ito, Hirotaka Ono, Yota Otachi, Ryuhei Uehara, and Takeshi Yamada. Linear-time algorithm for sliding tokens on trees. *Theoretical Computer Science*, 600:132–142, 2015. doi:10.1016/j.tcs.2015.07.037.

10 Reinhard Diestel. *Graph theory*. Springer-Verlag Berlin Heidelberg, 2010.

11 G. A. Dirac. Some theorems on abstract graphs. *Proceedings of the London Mathematical Society*, s3-2(1):69–81, 1952. doi:10.1112/plms/s3-2.1.69.

12 Adrian Dumitrescu and János Pach. Pushing squares around. *Graphs and Combinatorics*, 22(1):37–50, 2006. doi:10.1007/s00373-005-0640-1.

13 Eli Fox-Epstein, Duc A. Hoang, Yota Otachi, and Ryuhei Uehara. Sliding token on bipartite permutation graphs. In *Algorithms and Computation: 26th International Symposium, ISAAC 2015, Nagoya, Japan, December 9-11, 2015, Proceedings*, pages 237–247, 2015. doi:10.1007/978-3-662-48971-0_21.

14 Gilad Goraly and Refael Hassin. Multi-color pebble motion on graphs. *Algorithmica*, 58(3):610–636, 2010. doi:10.1007/s00453-009-9290-7.

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. *Theoretical Computer Science*, 412(12-14):1054–1065, 2011. doi:10.1016/j.tcs.2010.12.005.

17 Libin Jiang, Mathieu Leconte, Jian Ni, R. Srikant, and Jean Walrand. Fast mixing of parallel Glauber dynamics and low-delay CSMA scheduling. *IEEE Transactions on Information Theory*, 58(10):6541–6555, 2012. doi:10.1109/TIT.2012.2204032.

18 Marcin Kamiński, Paul Medvedev, and Martin Milanič. Complexity of independent set reconfigurability problems. *Theoretical Computer Science*, 439:9–15, 2012. doi:10.1016/j.tcs.2012.03.004.

19 Prashant Kataria and Rajarshi Roy. Parallel Glauber dynamics-based scheduling in static wireless grid networks with polynomially fading interference. *Electronics Letters*, 51(23):1948–1950, 2015. doi:10.1049/el.2014.3385.

20 Nancy G. Kinnersley and Michael A. Langston. Obstruction set isolation for the gate matrix layout problem. *Discrete Applied Mathematics*, 54(2-3):169–213, 1994. doi:10.1016/0166-218X(94)90021-3.

21 Daniel Lokshtanov, Amer E. Mouawad, Fahad Panolan, M. S. Ramanujan, and Saket Saurabh. Reconfiguration on sparse graphs. In *Algorithms and Data Structures – 14th International Symposium, WADS 2015, Victoria, BC, Canada, August 5-7, 2015. Proceedings*, pages 506–517, 2015. doi:10.1007/978-3-319-21840-3_42.

22 Amer E. Mouawad, Naomi Nishimura, Venkatesh Raman, Narges Simjour, and Akira Suzuki. On the parameterized complexity of reconfiguration problems. In *Parameterized and Exact Computation: 8th International Symposium, IPEC 2013, Sophia Antipolis, France, September 4-6, 2013, Revised Selected Papers*, pages 281–294, 2013. doi:10.1007/978-3-319-03898-8_24.

23 Francesca R. Nardi, Alessandro Zocca, and Sem C. Borst. Hitting time asymptotics for hard-core interactions on grids. *Journal of Statistical Physics*, 162(2):522–576, 2016. doi:10.1007/s10955-015-1391-x.

24 Marcin Wrochna. Reconfiguration in bounded bandwidth and treedepth. *arXiv:1405.0847*, 2014. arXiv:1405.0847.