Independent Set Reconfiguration Parameterized by Modular-Width*

Rémy Belmonte[1][0000−0001−8043−5343], Tesshu Hanaka[2][0000−0001−6943−856X], Michael Lampis[3][0000−0002−5791−0887], Hirotaka Ono[4][0000−0003−0845−3947], and Yota Otachi[5][0000−0002−0087−853X]

[1] The University of Electro-Communications, Chofu, Tokyo, 182-8585, Japan
remy.belmonte@uec.ac.jp
[2] Chuo University, Bunkyo-ku, Tokyo, 112-8551, Japan
hanaka.91t@g.chuo-u.ac.jp
[3] Université Paris-Dauphine, PSL University, CNRS, LAMSADE 75016, Paris, France
michail.lampis@dauphine.fr
[4] Nagoya University, Nagoya, 464-8601, Japan
ono@nagoya-u.jp
[5] Kumamoto University, Kumamoto, 860-8555, Japan
otachi@cs.kumamoto-u.ac.jp

Abstract. Independent Set Reconfiguration is one of the most well-studied problems in the setting of combinatorial reconfiguration. It is known that the problem is PSPACE-complete even for graphs of bounded bandwidth. This fact rules out the tractability of parameterizations by most of well-studied structural parameters as most of them generalize bandwidth. In this paper, we study the parameterization by modular-width, which is not comparable with bandwidth. We show that the problem parameterized by modular-width is fixed-parameter tractable under all previously studied rules TAR, TJ, and TS. The result under TAR resolves an open problem posed by Bonsma [WG 2014, JGT 2016].

Keywords: reconfiguration · independent set · modular-width.

1 Introduction

In a reconfiguration problem, we are given an instance of a search problem together with two feasible solutions. The algorithmic task there is to decide whether one solution can be transformed to the other by a sequence of prescribed local modifications while keeping the feasibility of intermediate states. Recently, reconfiguration versions of many search problems have been studied (see [14,23]).

* Partially supported by JSPS and MAEDI under the Japan-France Integrated Action Program (SAKURA) Project GRAPA 38593YJ, and by JSPS/MEXT KAKENHI Grant Numbers JP24106004, JP17H01698, JP18K11157, JP18K11168, JP18K11169, JP18H04091, JP18H06469.

Independent Set Reconfiguration is one of the most well-studied reconfiguration problems. In this problem, we are given a graph and two independent sets. Our goal is to find a sequence of independent sets that represents a step-by-step modification from one of the given independent sets to the other. There are three local modification rules studied in the literature: Token Addition and Removal (TAR) [3,19,22], Token Jumping (TJ) [4,5,16,17,18], and Token Sliding (TS) [1,2,8,10,13,15,21]. Under TAR, given a threshold k, we can remove or add any vertices as long as the resultant independent set has size at least k. (When we want to specify the threshold k, we call the rule TAR(k).) TJ allows to swap one vertex in the current independent set with another vertex not dominated by the current independent set. TS is a restricted version of TJ that additionally asks the swapped vertices to be adjacent.

It is known that Independent Set Reconfiguration is PSPACE-complete under all three rules for general graphs [16], for perfect graphs [19], and for planar graphs of maximum degree 3 [13] (see [4]). For claw-free graphs, the problem is solvable under all three rules [4]. For even-hole-free graphs (graphs without induced cycles of even length), the problem is known to be polynomial-time solvable under TAR and TJ [19], while it is PSPACE-complete under TS even for split graphs [1]. Under TS, forests [8] and interval graphs [2] form maximal known subclasses of even-hole-free graphs for which Independent Set Reconfiguration is polynomial-time solvable. For bipartite graphs, the problem is PSPACE-complete under TS and, somewhat surprisingly, it is NP-complete under TAR and TJ [21].

Independent Set Reconfiguration is studied also in the setting of parameterized computation. (See the recent textbook [7] for basic concepts in parameterized complexity.) It is known that there is a constant b such that the problem is PSPACE-complete under all three rules even for graphs of bandwidth at most b [25]. Since bandwidth is a lower bound of well-studied structural parameters such as pathwidth, treewidth, and clique-width, this result rules out FPT (and even XP) algorithms with these parameters. Given this situation, Bonsma [3] asked whether Independent Set Reconfiguration parameterized by modular-width is tractable under TAR and TJ. The main result of this paper is to answer this question by presenting an FPT algorithm for Independent Set Reconfiguration under TAR and TJ parameterized by modular-width. We also show that under TS the problem allows a much simpler FPT algorithm.

Our results in this paper can be summarized as follows:[6]

Theorem 1.1. *Under all three rules* TAR*,* TJ*, and* TS*,* Independent Set Reconfiguration *parameterized by modular-width* mw *can be solved in time* $O^*(2^{\mathrm{mw}})$.

In Section 3, we give our main result for TAR (Theorem 3.9), which implies the result for TJ (Corollary 3.10). The FPT algorithm under TS is given in Section 4 (Theorem 4.7).

The proofs marked with ★'s are omitted due to the space limitation.

[6] The $O^*(\cdot)$ notation suppresses factors polynomial in the input size.

2 Preliminaries

Let $G = (V, E)$ be a graph. For a set of vertices $S \subseteq V$, we denote by $G[S]$ the subgraph induced by S. For a vertex set $S \subseteq V$, we denote by $G - S$ the graph $G[V \setminus S]$. For a vertex $u \in V$, we write $G - u$ instead of $G - \{u\}$. For $u, v \in S$, we denote $S \cup \{u\}$ by $S + u$ and $S \setminus \{v\}$ by $S - v$, respectively. We use $\alpha(G)$ to denote the size of a maximum independent set of G. For two sets S, R we use $S \triangle R$ to denote their symmetric difference, that is, the set $(S \setminus R) \cup (R \setminus S)$. For an integer k we use $[k]$ to denote the set $\{1, \ldots, k\}$. For a vertex $v \in V$, its (*open*) *neighborhood* is denoted by $N(v)$. The *open neighborhood* of a set $S \subseteq V$ of vertices is defined as $N(S) = \bigcup_{v \in S} N(v) \setminus S$. A *component* of G is a maximal vertex set $S \subseteq V$ such that G contains a path between each pair of vertices in S.

In the rest of this section, we are going to give definitions of the terms used in the following formalization of the main problem:

Problem: Independent Set Reconfiguration under $\mathsf{TAR}(k)$
Input: A graph G, an integer k, and independent sets S and S' of G.
Parameter: The modular-width of the input graph $\mathrm{mw}(G)$.
Question: Does $S \leftrightsquigarrow_k S'$ hold?

2.1 TAR(k) rule

Let S and S' be independent sets in a graph G and k an integer. Then we write $S \overset{G}{\leftrightarrow}_k S'$ if $|S \triangle S'| \leq 1$ and $\min\{|S|, |S'|\} \geq k$. If G is clear from the context we simply write $S \leftrightarrow_k S'$. Here $S \leftrightarrow_k S'$ means that S and S' can be reconfigured to each other in one step under the $\mathsf{TAR}(k)$ rule, which stands for "Token Addition and Removal", under the condition that no independent set contains fewer than k vertices (tokens). We write $S \overset{G}{\leftrightsquigarrow}_k S'$, or simply $S \leftrightsquigarrow_k S'$ if G is clear, if there exists $\ell \geq 0$ and a sequence of independent sets $S_0, \ldots, S_\ell$ with $S_0 = S$, $S_\ell = S'$ and for all $i \in [\ell]$ we have $S_{i-1} \leftrightarrow_k S_i$. If $S \leftrightsquigarrow_k S'$ we say that S' is *reachable* from S under the $\mathsf{TAR}(k)$ rule.

We recall the following basic facts.

Observation 2.1. *For all integers k the relation defined by $\leftrightsquigarrow_k$ is an equivalence relation on independent sets of size at least k. For any graph G, integer k, and independent sets S, R, if $S \leftrightsquigarrow_k R$, then $S \leftrightsquigarrow_{k-1} R$. For any graph G and independent sets S, R we have $S \leftrightsquigarrow_0 R$.*

2.2 TJ and TS rules

Under the TJ rule, one step is formed by a removal of a vertex and an addition of a vertex. As this rule does not change the size of the independent set, we assume that the given initial and target independent sets are of the same size. In other words, two independent sets S and S' with $|S| = |S'|$ can be reconfigured to each other in one step under the TJ rule if $|S \triangle S'| = 2$. It is known that the TJ reachability can be seen as a special case of TAR reachability as follows.

Proposition 2.2 ([19]). *Let S and R be independent sets of G with $|S| = |R|$. Then, S and R are reachable to each other under* TJ *if and only if $S \leftrightsquigarrow_{|S|-1} R$.*

One step under the TS rule first has to be a TJ step, and additionally has to satisfy that the removed and added vertices are adjacent. Intuitively, one step in a TS sequence "slides" a token along an edge. We postpone the introduction of notation for TS until Section 4 to avoid any confusions.

2.3 Modular-width

In a graph $G = (V, E)$ a module is a set of vertices $M \subseteq V$ with the property that for all $u, v \in M$ and $w \in V \setminus M$, if $\{u, w\} \in E$, then $\{v, w\} \in E$. In other words, a module is a set of vertices that have the same neighbors outside the module. A graph $G = (V, E)$ has *modular-width* at most k if it satisfies at least one of the following conditions (i) $|V| \leq k$, or (ii) there exists a partition of V into at most k sets $V_1, V_2, \ldots, V_s$, such that $G[V_i]$ has modular-width at most k and V_i is a module in G, for all $i \in [s]$. We will use $\mathrm{mw}(G)$ to denote the minimum k for which G has modular-width at most k. We recall that there is a polynomial-time algorithm which, given a graph $G = (V, E)$ produces a non-trivial partition of V into at most $\mathrm{mw}(G)$ modules [6,12,24] and that deleting vertices from G can only decrease the modular-width. We also recall that Maximum Independent Set is solvable in time $O^*(2^{\mathrm{mw}})$. Indeed, a faster algorithm with running time $O^*(1.7347^{\mathrm{mw}})$ is known [9].

A graph has neighborhood diversity at most k if its vertex set can be partitioned into k modules, such that each module induces either a clique or an independent set. We use $\mathrm{nd}(G)$ to denote the minimum neighborhood diversity of G, and recall that $\mathrm{nd}(G)$ can be computed in polynomial time [20] and that $\mathrm{nd}(G) \geq \mathrm{mw}(G)$ for all graphs G [11].

It can be seen that the modular-width of a graph is not smaller than its clique-width. On the other hand, we can see that treewidth, pathwidth, and bandwidth are not comparable to modular-width. To see this, observe that the complete graph of n vertices has treewidth $n-1$ and modular-width 2, and that the path of n vertices has treewidth 1 and modular-width n for $n \geq 4$. Our positive result and the hardness result by Wrochna [25] together give Fig. 1 that depicts a map of structural graph parameters with a separation the complexity of Independent Set Reconfiguration.

3 FPT Algorithm for Modular-Width under TAR

In this section we present an FPT algorithm for the TAR(k)-reachability problem parameterized by modular-width. The main technical ingredient of our algorithm is a sub-routine which solves a related problem: given a graph G, an independent set S, and an integer k, what is the largest independent set that is reachable from S under TAR(k)? This sub-routine relies on dynamic programming: we present (in Lemma 3.4) an algorithm which answers this "maximum extensibility" question, if we are given tables with answers for the same question for all the modules

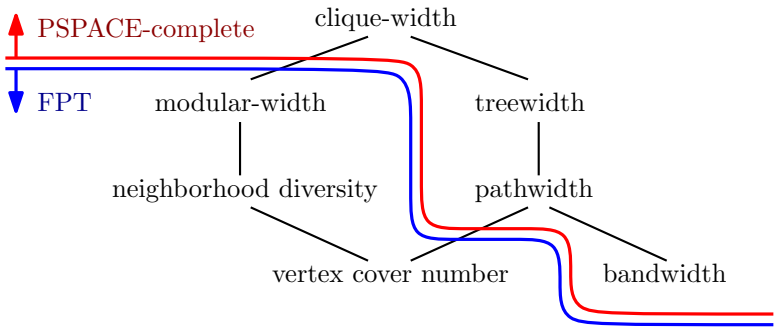

Fig. 1. The complexity of Independent Set Reconfiguration under TAR, TJ, and TS parameterized by structural graph parameters.

in a non-trivial partition of the input graph. This results in an algorithm (Theorem 3.5) that solves this problem on graphs of small modular-width, which we then put to use in Section 3.2 to solve the reconfiguration problem.

3.1 Computing the Largest Reachable Set

In this section we present an FPT algorithm (parameterized by modular-width) which computes the following value:

Definition 3.1. *Given a graph G, an independent set S, and an integer k, we define $\lambda(G, S, k)$ as the size of the largest independent set S' such that $S \leftrightsquigarrow_k S'$.*

In particular, we will present a *constructive* algorithm which, given G, S, k will return an independent set S' such that $|S'| = \lambda(G, S, k)$, as well as a reconfiguration sequence proving that $S \leftrightsquigarrow_k S'$.

We begin by tackling an easier case: the case when the parameter is the neighborhood diversity.

Lemma 3.2 (★). *There is an algorithm which, given a graph G, an independent set S, and an integer k, returns an independent set S', with $|S'| = \lambda(G, S, k)$, and a reconfiguration sequence proving that $S \leftrightsquigarrow_k S'$, in time $O^*(2^{\mathrm{nd}(G)})$.*

Before presenting the main algorithm of this section, let us also make a useful observation: once we are able to reach a configuration that contains a sufficiently large number of vertices from a module, we can safely delete a vertex from the module (bringing us closer to the case where Lemma 3.2 will apply).

Lemma 3.3 (★). *Let G be a graph, S be an independent set of G, k an integer, and M a module of G. Suppose there exists an independent set $A \subset M$ such that $(S \cap M) \subseteq A$ and $|A| = \alpha(G[M])$. Then, for all $u \in M \setminus A$ we have $\lambda(G, S, k) = \lambda(G - u, S, k)$.*

We are now ready to present our main dynamic programming procedure.

Lemma 3.4 (★). *Suppose we are given the following input:*

1. *A graph $G = (V, E)$, an integer k, and an independent set S with $|S| \geq k$.*
2. *A partition of V into $r \leq \mathrm{mw}(G)$ non-empty modules, $V_1, \ldots, V_r$.*
3. *For each $i \in [r]$, for each $j \in [|S \cap V_i|]$ an independent set $R_{i,j}$, such that $|R_{i,j}| = \lambda(G[V_i], S \cap V_i, j)$, and a transformation sequence proving that $(S \cap V_i) \overset{G[V_i]}{\leftrightsquigarrow}_j R_{i,j}$.*

Then, there exists an algorithm which returns an independent set R of G, such that $|R| = \lambda(G, S, k)$, and a transformation sequence proving that $S \leftrightsquigarrow_k R$, running in time $O^*(2^{\mathrm{mw}(G)})$.

We thus arrive to the main theorem of this section.

Theorem 3.5. *There exists an algorithm which, given a graph G, an independent set S, and an integer k, runs in time $O^*(2^{\mathrm{mw}(G)})$ and outputs an independent set S' such that $|S'| = \lambda(G, S, k)$ and a $\mathsf{TAR}(k)$ transformation $S \leftrightsquigarrow_k S'$.*

Proof. We perform dynamic programming using Lemma 3.4. More precisely, our goal is, given G and S, to produce for each value of $j \in [|S|]$ an independent set R_j such that $S \leftrightsquigarrow_k R_j$ and $|R_j| = \lambda(G, S, j)$. Clearly, if we can solve this more general problem in time $O^*(2^{\mathrm{mw}(G)})$ we are done.

Our algorithm works as follows: first, it computes a modular decomposition of G of minimum width, which can be done in time at most $O(n^2)$ [12]. If $|V(G)| \leq \mathrm{mw}(G)$ then the problem can be solved in $O^*(2^{\mathrm{mw}(G)})$ by brute force (enumerating all independent sets of G), or even by Lemma 3.2. We therefore assume that G has a non-trivial partition into $r \leq \mathrm{mw}(G)$ modules $V_1, \ldots, V_r$. We call our algorithm recursively for each $G[V_i]$, and obtain for each $i \in [r]$ and $j \in [|S \cap V_i|]$ a set $R_{i,j}$ such that $|R_{i,j}| = \lambda(G[V_i], S \cap V_i, j)$ and a transformation $(S \cap V_i) \overset{G[V_i]}{\leftrightsquigarrow}_j R_{i,j}$. We use this input to invoke the algorithm of Lemma 3.4 for each value of $j \in [|S|]$. This allows us to produce the sets R_j and the corresponding transformations.

Suppose that $\beta \geq 2$ is a constant such that the algorithm of Lemma 3.4 runs in time at most $O(2^{\mathrm{mw}(G)} n^\beta)$. Our algorithm runs in time at most $O(2^{\mathrm{mw}(G)} n^{\beta+2})$. This can be seen by considering the tree representing a modular decomposition of G. In each node of the tree (that represents a module of G) our algorithm makes at most n calls to the algorithm of Lemma 3.4. Since the modular decomposition has at most $O(n)$ nodes, the running time bound follows. □

3.2 Reachability

In this section we will apply the algorithm of Theorem 3.5 to obtain an FPT algorithm for the $\mathsf{TAR}(k)$ reconfiguration problem parameterized by modular-width. The main ideas we will need are that (i) using the algorithm of Theorem 3.5 we can decide if it is possible to arrive at a configuration where a module is empty of tokens (Lemma 3.6) (ii) if a module is empty in both the initial

and target configurations, we can replace it by an independent set (Lemma 3.7) and (iii) the reconfiguration problem is easy on graphs with small neighborhood diversity (Lemma 3.8). Putting these ideas together we obtain an algorithm which can always identify an irrelevant vertex which we can delete if the input graph is connected. If the graph is disconnected, we can use ideas similar to those of [3] to reduce the problem to appropriate sub-instances in each component.

Lemma 3.6. *There is an algorithm which, given a graph G, an independent set S, a module M of G, and an integer k, runs in time $O^*(2^{\mathrm{mw}(G)})$ and either returns a set S' with $S' \cap M = \emptyset$ and $S \leftrightsquigarrow_k S'$ or correctly concludes that no such set exists.*

Proof. We assume that $S \cap M \neq \emptyset$ (otherwise we simply return S).

Let H be the graph obtained by deleting from G all vertices of $V \setminus M$ that have a neighbor in M. We invoke the algorithm of Theorem 3.5 to compute a set S' in H such that $S \overset{H}{\leftrightsquigarrow}_k S'$ and $|S'| = \lambda(H, S, k)$. If $|S' \setminus M| \geq k$ then we return as solution the set $S' \setminus M$, and as transformation the transformation sequence returned by the algorithm, to which we append moves that delete all vertices of $S' \cap M$. If $|S' \setminus M| < k$ we answer that no such set exists.

Let us now argue for correctness. If the algorithm returns a set S', it also returns a $\mathsf{TAR}(k)$ transformation from S to S' in H; this is also a transformation in G, and since $S' \cap M = \emptyset$, the solution is correct.

Suppose then that the algorithm returns that no solution exists, but for the sake of contradiction there exists a T with $S \overset{G}{\leftrightsquigarrow}_k T$ and $T \cap M = \emptyset$. Among all such sets T select the one at minimum reconfiguration distance from S and let $S_0 = S, S_1, \ldots, S_\ell = T$ be a shortest reconfiguration sequence. We claim that this is also a valid reconfiguration sequence in H. Indeed, for all $j \in [\ell - 1]$, the set S_j contains a vertex from M (otherwise we would have a shorter sequence), therefore may not contain any deleted vertex. As a result, if a solution T exists, then $S \overset{H}{\leftrightsquigarrow}_k T$. Let A be a maximum independent set of $G[M]$. We observe that (i) $|T \setminus M| \geq k$ since T is reachable with $\mathsf{TAR}(k)$ moves and $T \cap M = \emptyset$ (ii) $T \overset{H}{\leftrightsquigarrow}_k (T \cup A)$. However, this gives a contradiction, because we now have $S \overset{H}{\leftrightsquigarrow}_k (T \cup A)$ and this set is strictly larger than the set returned by the algorithm of Theorem 3.5 when computing $\lambda(H, S, k)$. ◻

Lemma 3.7. *Let G be a graph, k an integer, M a module of G, and S, T two independent sets of G such that $S \cap M = T \cap M = \emptyset$. Let A be a maximum independent set of $G[M]$. Then, for all $u \in M \setminus A$ we have $S \overset{G}{\leftrightsquigarrow}_k T$ if and only if $S \overset{G-u}{\leftrightsquigarrow}_k T$.*

Proof. The proof is similar to that of Lemma 3.3. Specifically, since $u \notin S$ and $u \notin T$, it is easy to see that $S \overset{G-u}{\leftrightsquigarrow}_k T$ implies $S \overset{G}{\leftrightsquigarrow}_k T$. Suppose then that $S \overset{G}{\leftrightsquigarrow}_k T$ and we have a sequence $S_0 = S, S_1, \ldots, S_\ell = T$. We construct a sequence $S'_0 = S, S'_1, \ldots, S'_\ell$ such that for all $i \in [\ell]$ we have $|S_i| = |S'_i|$, $S_i \setminus M = S'_i \setminus M$, and $S'_i \subseteq A$. This can be done inductively: for S'_0 the desired

properties hold; and for all $i \in [\ell]$ we can prove that if the properties hold for S'_{i-1}, then we can construct S'_i in the same way as in the proof of Lemma 3.3 (namely, we perform the same moves as S_i outside of M, and pick an arbitrary vertex of A when S_i adds a vertex of M). □

Lemma 3.8. *There is an algorithm which, given a graph G, an integer k, and two independent sets S, T, decides if $S \leftrightsquigarrow_k T$ in time $O^*(2^{\mathrm{nd}(G)})$.*

Proof. The proof is similar to that of Lemma 3.2, but we need to carefully handle some corner cases. We are given a partition of G into $r \leq \mathrm{nd}(G)$ sets $V_1, \ldots, V_r$, such that each V_i induces a clique or an independent set. Suppose V_i induces a clique. We use the algorithm of Lemma 3.6 with input (G,S,V_i,k) and with input (G,T,V_i,k) to decide if it is possible to empty V_i of tokens. If the algorithm gives different answers we immediately reject, since there is a configuration that is reachable from S but not from T. If the algorithm returns S', T' with $S' \cap V_i = T' \cap V_i = \emptyset$, then the problem reduces to deciding if $S' \leftrightsquigarrow_k T'$. However, by Lemma 3.7 we can delete all the vertices of V_i except one and this does not change the answer. Finally, if the algorithm responds that V_i cannot be empty in any configuration reachable from S or T then, if $S \cap V_i \neq T \cap V_i$ we immediately reject, while if $S \cap V_i = T \cap V_i$ we delete from the input V_i and all its neighbors and solve the reconfiguration problem in the instance $(G[V \setminus N[V_i]],\ k-1,\ S \setminus V_i,\ T \setminus V_i)$.

After this preprocessing all sets V_i are independent. We now construct an auxiliary graph G' as in Lemma 3.2, namely, our graph has a vertex for every independent set S of G with $|S| \geq k$ such that for all $i \in [r]$ either $S \cap V_i = \emptyset$ or $V_i \subseteq S$. Again, we have an edge between S_1, S_2 if $S_1 \vartriangle S_2 = V_i$ for some $i \in [r]$. We can assume without loss of generality that S, T are represented in this graph (if there exists V_i such that $0 < |S \cap V_i| < |V_i|$ we add to S all remaining vertices of V_i). Now, $S \leftrightsquigarrow T$ if and only if S is reachable from T in G', and this can be checked in time linear in the size of G'. □

Theorem 3.9 (TAR)**.** *There is an algorithm which, given a graph G, an integer k, and two independent sets S, T, decides if $S \leftrightsquigarrow_k T$ in time $O^*(2^{\mathrm{mw}(G)})$.*

Proof. Our algorithm considers two cases: if G is connected we will attempt to simplify G in a way that eventually produces either a graph with small neighborhood diversity or a disconnected graph; if G is disconnected we will recursively solve an appropriate subproblem in each component.

First, suppose that G is connected. We compute a modular decomposition of G which gives us a partition of V into $r \leq \mathrm{mw}(G)$ modules $V_1, \ldots, V_r$. If for all $i \in [r]$ we have that $G[V_i]$ is an independent set, then $\mathrm{nd}(G) \leq r$ and we invoke the algorithm of Lemma 3.8. Suppose then that for some $i \in [r]$, $G[V_i]$ contains at least one edge. We invoke the algorithm of Lemma 3.6 on input (G, S, V_i, k) and on input (G, T, V_i, k). If the answers returned are different, we decide that S is not reachable from T in G, because from one set we can reach a configuration that contains no vertex of V_i and from the other we cannot.

If the algorithm of Lemma 3.6 returned to us two sets S', T' with $S' \cap V_i = T' \cap V_i = \emptyset$ then by transitivity we know $S \leftrightsquigarrow T$ if and only if $S' \leftrightsquigarrow T'$. We compute a maximum independent set A of $G[V_i]$ and delete from our graph a vertex $u \in V_i \setminus A$. Such a vertex exists, since $G[V_i]$ is not an independent set. By Lemma 3.7 deleting u does not affect whether $S' \leftrightsquigarrow T'$, so we call our algorithm with input $(G - u, k, S', T')$, and return its response.

On the other hand, if the algorithm of Lemma 3.6 concluded that no set reachable from either S or T has empty intersection with V_i, we find a vertex $u \in V \setminus V_i$ that has a neighbor in V_i and delete it, that is, we call our algorithm with input $(G - u, k, S, T)$. Such a vertex u exists because G is connected. This recursive call is correct because any configuration reachable from S or T contains some vertex of V_i, which is a neighbor of u, so no reachable configuration uses u.

We note that if G is connected, all the cases described above will make a single recursive call on an input that has strictly fewer vertices.

Suppose now that G is not connected and there are s connected components $C_1, C_2, \ldots, C_s$. We will assume that $|S| = \lambda(G, S, k)$ and $|T| = \lambda(G, T, k)$. This is without loss of generality, since we can invoke the algorithm of Theorem 3.5 and in case $|S| < \lambda(G, S, k)$ replace S with the set S' returned by the algorithm while keeping an equivalent instance (similarly for T).

As a result, we can assume that $|S| = |T|$, otherwise the answer is trivially no. More strongly, if there exists a component C_i such that $|S \cap C_i| \neq |T \cap C_i|$ we answer no. To see that this is correct, we argue that for all S' such that $S \leftrightsquigarrow_k S'$ we have $|S' \cap C_i| \leq |S \cap C_i|$. Indeed, suppose there exists S' such that for some $i \in [s]$ we have $|S' \cap C_i| > |S \cap C_i|$ and $S \leftrightsquigarrow S'$. Among such configurations S' select one that is at minimum reconfiguration distance from S and let $S_0 = S, S_1, \ldots, S_\ell = S'$ be a shortest reconfiguration from S to S'. Then for all $j \in [\ell]$ we have $|S \setminus C_i| \geq |S_j \setminus C_i|$ (otherwise we would have an S' that is at shorter reconfiguration distance from S). This means that the sequence $S_0 \cap C_i, S_1 \cap C_i, \ldots, S_\ell \cap C_i$ is a $\mathsf{TAR}(k - |S \setminus C_i|)$ transformation of $S \cap C_i$ to $S' \cap C_i$ in $G[C_i]$. But this transformation proves that the set $(S \setminus C_i) \cup (S' \cap C_i)$ is $\mathsf{TAR}(k)$ reachable from S in G, and since this set is larger than S we have a contradiction.

For each $i \in [s]$ we now consider the reconfiguration instance given by the following input: $(G[C_i], k - |S \setminus C_i|, S \cap C_i, T \cap C_i)$. We call our algorithm recursively for each such instance. If the answer is yes for all these instances we reply that S is reachable from T, otherwise we reply that the sets are not reachable.

To argue for correctness we use induction on the depth of the recursion. Suppose that the algorithm correctly concludes that the answer to all sub-instances is yes. Then, there does indeed exist a transformation $S \leftrightsquigarrow T$ as follows: starting from S, for each $i \in [s]$ we keep $S \setminus C_i$ constant and perform in $G[C_i]$ the transformation $(S \cap C_i) \overset{G[C_i]}{\leftrightsquigarrow}_{k-|S\setminus C_i|} (T \cap C_i)$. At each step this gives a configuration where S and T agree in more components. Furthermore, since $|S \cap C_i| = |T \cap C_i|$ for all $i \in [s]$, this is a valid $\mathsf{TAR}(k)$ reconfiguration.

Suppose now that the answer is no for the instance $(G[C_i], k - |S \setminus C_i|, S \cap C_i, T \cap C_i)$. Suppose also, for the sake of contradiction, that there exists a $\mathsf{TAR}(k)$ reconfiguration $S_0 = S, S_1, \ldots, S_\ell = T$. As argued above, any configuration S' reachable from S has $|S' \cap C_{i'}| \leq |S \cap C_{i'}|$ for all $i' \in [s]$. This means that $|S \setminus C_i| \geq |S_j \setminus C_i|$ for all $j \in [\ell]$. Hence, the sequence $S_0 \cap C_i, S_1 \cap C_i, \ldots, S_\ell \cap C_i$ gives a valid $\mathsf{TAR}(k - |S \setminus C_i|)$ reconfiguration in $G[C_i]$, which is a contradiction.

Finally, it is not hard to see that the algorithm runs in time $O^*(2^{\mathrm{mw}(G)})$, because in the case of disconnected graphs we make a single recursive call for each component. □

Theorem 3.9 and Proposition 2.2 give an FPT algorithm with the same running time for TJ.

Corollary 3.10 (TJ). *There is an algorithm which, given a graph G and two independent sets S, T, decides the TJ reachability between S and T in time $O^*(2^{\mathrm{mw}(G)})$.*

4 FPT Algorithm for Modular-Width under TS

We now present an FPT algorithm deciding the TS-reachability parameterized by modular-width. The problem under TS is much easier than the one under TAR since we can reduce the problem to a number of constant-size instances that can be considered separately. To see this, we first observe that the components can be considered separately. We then further observe that we only need to solve the case where each maximal nontrivial module contains at most one vertex of the current independent set. Finally, we show that the reachability problem on the reduced case thus far is equivalent to a generalized reachability problem on a graph of order at most $\mathrm{mw}(G)$, where G is the original graph.

Let S and S' be independent sets of G with $|S| = |S'|$. Recall that S and S' can be reachable by one step under TS if $|S \vartriangle S'| = 2$ and the two vertices in $S \vartriangle S'$ are adjacent. We denote this relation by $S \overset{G}{\leftrightarrow} S'$, or simply by $S \leftrightarrow S'$ if G is clear from the context. We write $S \overset{G}{\leftrightsquigarrow} S'$ (or simply $S \leftrightsquigarrow S'$) if there exists $\ell \geq 0$ and a sequence of independent sets $S_0, \ldots, S_\ell$ with $S_0 = S$, $S_\ell = S'$ and for all $i \in [\ell]$ we have $S_{i-1} \leftrightarrow S_i$. If $S \leftrightsquigarrow S'$ we say that S' is *reachable* from S under the TS rule. Observe that the relation defined by $\leftrightsquigarrow$ is an equivalence relation on independent sets.

The first easy observation is that the TS rule cannot move a token to a different component since a TS step is always along an edge (and thus within a component). This is formalized as follows.

Observation 4.1. *Let G be a graph, S, S' independent sets of G, and $C_1, \ldots, C_c$ the components of G. Then, $S \overset{G}{\leftrightsquigarrow} S'$ if and only if $(S \cap V(C_i)) \overset{G[V(C_i)]}{\leftrightsquigarrow} (S' \cap V(C_i))$ for all $i \in [c]$.*

The next lemma, which is still an easy one, is a key tool in our algorithm.

Lemma 4.2 (★). *Let G be a graph, M a module of G, and S an independent set in G such that $|S \cap M| \geq 2$. Then, for every independent set S' in G, $S \overset{G}{\leftrightsquigarrow} S'$ if and only if $S' \cap N(M) = \emptyset$ and $S \overset{G-N(M)}{\leftrightsquigarrow} S'$.*

Lemma 4.2 implies that S with $|S \cap M| \geq 2$ and S' with $|S' \cap M| \leq 1$ are not reachable to each other. This fact in the following form will be useful later.

Corollary 4.3. *Let G be a graph, M a module of G, and S an independent set in G such that $|S \cap M| \leq 1$. Then, for every independent set S' in G such that $S \leftrightsquigarrow S'$, it holds that $|S' \cap M| \leq 1$.*

We now show that a module sharing at most one vertex with both initial and target independent sets can be replaced with a single vertex, under an assumption that we may solve a slightly generalized reachability problem (which is still trivial on a graph of constant size).

Lemma 4.4 (★). *Let G be a graph, M a module of G with $|M| \geq 2$, and S, S' independent sets of G with $|S| = |S'|$. If $|M \cap (S \cup S')| \leq 1$, then $S \overset{G}{\leftrightsquigarrow} S'$ if and only if $S \overset{G-v}{\leftrightsquigarrow} S'$ for every $v \in M \setminus (S \cup S')$.*

Lemma 4.5 (★). *Let G be a graph, M a module of G, and S, S' independent sets of G with $|S| = |S'|$. If $M \cap S = \{u\}$, $M \cap S' = \{v\}$, $u \neq v$, and u and v are in the same component of $G[M]$, then $S \overset{G}{\leftrightsquigarrow} S'$ if and only if $S \overset{G-v}{\leftrightsquigarrow} S' - v + u$.*

Lemma 4.6 (★). *Let G be a graph, M a module of G, and S, S' independent sets of G with $|S| = |S'|$. If $M \cap S = \{u\}$, $M \cap S' = \{v\}$, $u \neq v$, and u and v are in different components of $G[M]$, then $S \overset{G}{\leftrightsquigarrow} S'$ if and only if $S \overset{G-v}{\leftrightsquigarrow} S' - v + u$ and there is an independent set T in $G - v$ such that $T \cap M = \emptyset$ and $S \overset{G-v}{\leftrightsquigarrow} T$.*

Now we are ready to present our algorithm for the TS-reachability problem.

Theorem 4.7 (TS, ★). *There is an algorithm which, given a graph G and two independent sets S, T, decides if $S \leftrightsquigarrow T$ in time $O^*(2^{\mathrm{mw}(G)})$.*

References

1. R. Belmonte, E. J. Kim, M. Lampis, V. Mitsou, Y. Otachi, and F. Sikora. Token sliding on split graphs. In *STACS*, LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2019.
2. M. Bonamy and N. Bousquet. Token sliding on chordal graphs. In *WG 2017*, volume 10520 of *LNCS*, pages 127–139, 2017.
3. P. S. Bonsma. Independent set reconfiguration in cographs and their generalizations. *Journal of Graph Theory*, 83(2):164–195, 2016.
4. P. S. Bonsma, M. Kaminski, and M. Wrochna. Reconfiguring independent sets in claw-free graphs. In *SWAT*, volume 8503 of *LNCS*, pages 86–97. Springer, 2014.
5. N. Bousquet, A. Mary, and A. Parreau. Token jumping in minor-closed classes. In *FCT*, volume 10472 of *LNCS*, pages 136–149. Springer, 2017.

6. A. Cournier and M. Habib. A new linear algorithm for modular decomposition. In *CAAP*, volume 787 of *LNCS*, pages 68–84. Springer, 1994.
7. M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer, 2015.
8. E. D. Demaine, M. L. Demaine, E. Fox-Epstein, D. A. Hoang, T. Ito, H. Ono, Y. Otachi, R. Uehara, and T. Yamada. Linear-time algorithm for sliding tokens on trees. *Theor. Comput. Sci.*, 600:132–142, 2015.
9. F. V. Fomin, M. Liedloff, P. Montealegre, and I. Todinca. Algorithms parameterized by vertex cover and modular width, through potential maximal cliques. *Algorithmica*, 80(4):1146–1169, 2018.
10. E. Fox-Epstein, D. A. Hoang, Y. Otachi, and R. Uehara. Sliding token on bipartite permutation graphs. In *ISAAC*, volume 9472 of *LNCS*, pages 237–247. Springer, 2015.
11. J. Gajarský, M. Lampis, and S. Ordyniak. Parameterized algorithms for modular-width. In *IPEC*, volume 8246 of *LNCS*, pages 163–176. Springer, 2013.
12. M. Habib and C. Paul. A survey of the algorithmic aspects of modular decomposition. *Computer Science Review*, 4(1):41–59, 2010.
13. R. A. Hearn and E. 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.
14. J. van den Heuvel. The complexity of change. In S. R. Blackburn, S. Gerke, and M. Wildon, editors, *Surveys in Combinatorics 2013*, volume 409 of *London Mathematical Society Lecture Note Series*, pages 127–160. Cambridge University Press, 2013.
15. D. A. Hoang and R. Uehara. Sliding tokens on a cactus. In *ISAAC*, volume 64 of *LIPIcs*, pages 37:1–37:26. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.
16. T. Ito, E. D. Demaine, N. J. A. Harvey, C. H. Papadimitriou, M. Sideri, R. Uehara, and Y. Uno. On the complexity of reconfiguration problems. *Theor. Comput. Sci.*, 412(12–14):1054–1065, 2011.
17. T. Ito, M. Kaminski, H. Ono, A. Suzuki, R. Uehara, and K. Yamanaka. On the parameterized complexity for token jumping on graphs. In *TAMC*, volume 8402 of *LNCS*, pages 341–351. Springer, 2014.
18. T. Ito, M. J. Kaminski, and H. Ono. Fixed-parameter tractability of token jumping on planar graphs. In *ISAAC*, volume 8889 of *LNCS*, pages 208–219. Springer, 2014.
19. M. Kamiński, P. Medvedev, and M. Milanič. Complexity of independent set reconfigurability problems. *Theor. Comput. Sci.*, 439:9–15, 2012.
20. M. Lampis. Algorithmic meta-theorems for restrictions of treewidth. *Algorithmica*, 64(1):19–37, 2012.
21. D. Lokshtanov and A. E. Mouawad. The complexity of independent set reconfiguration on bipartite graphs. In *SODA 2018*, pages 185–195, 2018.
22. A. E. Mouawad, N. Nishimura, V. Raman, N. Simjour, and A. Suzuki. On the parameterized complexity of reconfiguration problems. *Algorithmica*, 78(1):274–297, 2017.
23. N. Nishimura. Introduction to reconfiguration. *Algorithms*, 11(4):52, 2018.
24. M. Tedder, D. G. Corneil, M. Habib, and C. Paul. Simpler linear-time modular decomposition via recursive factorizing permutations. In *ICALP (1)*, volume 5125 of *LNCS*, pages 634–645. Springer, 2008.
25. M. Wrochna. Reconfiguration in bounded bandwidth and tree-depth. *J. Comput. Syst. Sci.*, 93:1–10, 2018.

A Omitted Proofs in Section 3

Proof (Lemma 3.2). Assume that $G = (V, E)$ is partitioned into r sets $V_1, V_2, \ldots, V_r$ such that each set induces a clique or an independent set. In fact, we may assume without loss of generality that V_i is an independent set for each $i \in [r]$, because if V_i is a clique we can delete all but one of its vertices without affecting the answer.

Consider now the auxiliary graph G' which has a vertex for each independent set S of G that satisfies the following: (i) $|S| \geq k$ (ii) for all $i \in [r]$ either $V_i \subseteq S$ or $S \cap V_i = \emptyset$. There are at most 2^r vertices in G'. We add an edge between S_1, S_2 in G' if $S_1 \triangle S_2 = V_i$ for some $i \in [r]$. As a result, G' has at most $r2^r$ edges.

We observe that the set S' we seek is represented by a vertex of G' (S' must be maximal, therefore it fully contains all modules it intersects). Furthermore, we may assume that the set S we have been given is also represented by a vertex of G' (because if $0 < |S \cap V_i| < |V_i|$ we may add to S the remaining vertices of V_i and we still have a set that is reachable from S). We note that $S \leftrightsquigarrow_k S'$ if and only if there is a path from S to S' in G, and it is not hard to construct a reconfiguration sequence in G given a path in G'. As a result, the problem reduces to determining the vertices of G' which are reachable from S, and then determining which among these represents the largest independent set, both of which can be solved in time linear in the size of G'. □

Proof (Lemma 3.3). We assume that there exists a $u \in M \setminus A$ (otherwise the claim is vacuously true). We have $u \notin S$, since $(S \cap M) \subseteq A$ and $u \notin A$. Therefore, $\lambda(G, S, k) \geq \lambda(G - u, S, k)$, since any transformation which can be performed in $G - u$ can also be performed in G. We therefore need to argue that $\lambda(G - u, S, k) \geq \lambda(G, S, k)$.

Let T be an independent set of G such that $S \overset{G}{\leftrightsquigarrow}_k T$ and $|T| = \lambda(G, S, k)$. Consider a shortest $\mathsf{TAR}(k)$ reconfiguration from S to T, say $S_0 = S, S_1, \ldots, S_\ell = T$. We construct a $\mathsf{TAR}(k)$ reconfiguration $S'_0 = S, S'_1, \ldots, S'_\ell$ with the property that for all $i \in [\ell]$ we have $|S_i| = |S'_i|$, $S_i \setminus M = S'_i \setminus M$, and $(S'_i \cap M) \subseteq A$. If we achieve this we are done, since we have that $S \overset{G-u}{\leftrightsquigarrow}_k S'_\ell$ and $|S'_\ell| = |T| = \lambda(G, S, k)$.

We will construct the new reconfiguration sequence inductively. First, $S'_0 = S$ satisfies the desired properties. So, suppose that for some $i \in [\ell]$ we have $|S'_{i-1}| = |S_{i-1}|$, $S_{i-1} \setminus M = S'_{i-1} \setminus M$, and $(S'_{i-1} \cap M) \subseteq A$. We now consider the four possible cases corresponding to single reconfiguration moves from S_{i-1} to S_i. If $S_i = S_{i-1} \setminus \{v\}$, for some $v \in S_{i-1} \setminus M$, we set $S'_i = S'_{i-1} \setminus \{v\}$; this is a valid move since $v \in S'_{i-1}$. If $S_i = S_{i-1} \setminus \{v\}$, for some $v \in S_{i-1} \cap M$, then it must be the case that $S'_{i-1} \cap M \neq \emptyset$. Select an arbitrary $v' \in S'_{i-1} \cap M$ and set $S'_i = S'_{i-1} \setminus \{v'\}$. If $S_i = S_{i-1} \cup \{v\}$ for $v \in V \setminus M$, we set $S'_i = S'_{i-1} \cup \{v\}$; it is not hard to see that S'_i is still an independent set and satisfies the desired properties. Finally, if $S_i = S_{i-1} \cup \{v\}$ for $v \in M$, we observe that $|S_{i-1} \cap M| < \alpha(G[M]) = |A|$ (otherwise S_i would not be independent). Since $|S_{i-1} \cap M| = |S'_{i-1} \cap M|$ there exists $v' \in A \setminus S'_{i-1}$. We therefore set $S'_i = S'_{i-1} \cup \{v'\}$. □

Proof (Lemma 3.4). We describe an iterative procedure which maintains the following variables:

- A working graph H, and a working independent set R of H.
- A partition of $V(H)$ into $r+1$ sets $M_0, M_1, \ldots, M_r$, some of which may be empty.
- A tuple of $r+1$ non-negative integer "thresholds", t_i, for $i \in \{0, \ldots, r\}$.

Informally, the meaning of these variables is the following: H is a working copy of G where we may have deleted some vertices which we have found to be irrelevant (using Lemma 3.3); R represents a working independent set which is reachable from the initial set S in G (and we have a transformation to prove this reachability); M_0 represents the union of initial modules which we have "processed" using Lemma 3.3, and therefore turned into independent sets, which implies that $H[M_0]$ is a graph with small neighborhood diversity; and t_i represents a threshold above which we are allowed to perform internal transformations inside the set M_i without violating the size constraints (that is, while keeping $|R| \geq k$).

To make this more precise we will maintain the following invariants.

1. H is an induced subgraph of G and $M_0, M_1, \ldots, M_r$ is a partition of $V(H)$.
2. We have a transformation proving that $S \overset{G}{\rightsquigarrow}_k R$.
3. $\lambda(G, S, k) = \lambda(H, R, k)$.
4. $\mathrm{nd}(H[M_0]) \leq |\{i \in [r] \mid M_i = \emptyset\}| \leq \mathrm{mw}(G)$.
5. For all $i \in [r]$ we have either $M_i = V_i$ or $M_i = \emptyset$. If $M_i \neq \emptyset$ then $M_i \cap R \neq \emptyset$.
6. For all $i \in \{0, \ldots, r\}$ such that $M_i \neq \emptyset$ we have $k - |R \setminus M_i| \leq t_i \leq |R \cap M_i|$.
7. For each $i \in [r]$ such that $M_i \neq \emptyset$ we have a transformation $(S \cap V_i) \overset{G[V_i]}{\rightsquigarrow}_{t_i} (R \cap M_i)$.
8. For each $i \in [r]$ such that $M_i \neq \emptyset$ we have $\lambda(G[V_i], S \cap V_i, t_i) = |R \cap M_i|$.

Informally, invariants 1-3 state that we may have deleted some vertices of G and reconfigured the independent set, but this has not changed the answer. Invariants 4-5 state that H can be thought of as having two parts: the low neighborhood diversity part induced by M_0 and the rest which includes some unchanged modules of G. Finally for each such non-empty module M_i invariant 6 states that it is safe to perform $\mathsf{TAR}(t_i)$ moves inside M_i, and invariants 7,8 and state that the current configuration is reachable and best possible under such moves, inside the module.

In the remainder, when we say that we "perform" a $\mathsf{TAR}(k)$ transformation from a set R to a set R', what we mean is that our algorithm appends this transformation to the transformation which we already have from S to R (by invariant 2), to obtain a transformation from S to the new set R'.

Preprocessing: Our algorithm begins by performing some preprocessing steps which ensure that all invariants are satisfied. First, for each $i \in [r]$ we compute a maximum independent set A_i of $G[V_i]$ (this takes time $O^*(2^{\mathrm{mw}(G)})$). We initialize $H := G$ and $R := S$. For each $i \in [r]$ such that $V_i \cap S = \emptyset$ we set $M_i := \emptyset$, $t_i := 0$

we add all vertices of A_i to M_0, and we delete from H all vertices of $M_i \setminus A_i$. After this step we have satisfied invariants 1, 2 (trivially, since $R = S$), 4 (because M_0 is composed of the at most r modules which had an empty intersection with S, each of which now induces an independent set), and 5. To see that invariant 3 is satisfied we invoke Lemma 3.3 repeatedly: we see that the lemma trivially applies if $S \cap V_i = \emptyset$ and allows us to delete all vertices of a module which are outside a maximum independent set.

For the second preprocessing step we do the following for each $i \in [r]$ for which $S \cap V_i \neq \emptyset$: we set $M_i := V_i$ and $t_i = |S \cap V_i|$. We recall that we have been given in the input a set $R_{i,j}$ for $j = |S \cap V_i|$ such that $\lambda(G[V_i], S \cap V_i, j) = |R_{i,j}|$, and a transformation $(S \cap V_i) \overset{G[V_i]}{\rightsquigarrow}_j R_{i,j}$; we perform this transformation in H, leaving $R \setminus M_i$ unchanged, to obtain an independent set R such that $R \cap M_i = R_{i,j}$. This is a valid $\mathsf{TAR}(k)$ transformation in H since inside V_i the transformation maintains an independent set with at least t_i tokens at all times. We observe that after this step is applied for all $i \in [r]$, all invariants are satisfied. This preprocessing step is performed in polynomial time, because the sets $R_{i,j}$ and the transformations leading to them have been given to us as input. Thus, all our preprocessing steps take a total time of $O^*(2^{\mathrm{mw}(G)})$.

For the main part of its execution our algorithm will enter a loop which attempts to apply some rules (given below) which either delete some vertex of H or produce a new (larger) independent set R, while maintaining all invariants. Once this is no longer possible the algorithm returns the current set R as the solution.

Rule 1 (Irrelevant Vertices): Check if there exists $i \in [r]$ such that $H[M_i]$ induces at least one edge and $|R \cap M_i| = \alpha(H[M_i])$. Then, delete all vertices of $M_i \setminus R$ from H, set $M_0 := M_0 \cup (M_i \cap R)$, $t_i := 0$, and $M_i := \emptyset$.

Rule 2a (Configuration Improvement in M_0): Let $F_0 \subseteq M_0$ be the set of vertices of M_0 that have no neighbors in $\cup_{i \in [r]} M_i$. Let $\lambda_0 := \lambda(H[F_0], R \cap F_0, k - |R \setminus F_0|)$. If $\lambda_0 > |R \cap F_0|$ then set $t_0 := \max\{k - |R \setminus F_0|, 0\}$ and perform a transformation that leaves $R \setminus F_0$ unchanged and results in $|R \cap F_0| = \lambda_0$.

Rule 2b (Configuration Improvement in $V \setminus M_0$): For each $i \in [r]$ such that $M_i \neq \emptyset$ let $\lambda_i := \lambda(H[M_i], R \cap M_i, k - |R \setminus M_i|)$. If there exists i such that $\lambda_i > |R \cap M_i|$, then set $t_i := \max\{k - |R \setminus M_i|, 0\}$ and perform a transformation that leaves $R \setminus M_i$ unchanged and results in $|R \cap M_i| = \lambda_i$.

Claim A.1. Rule 1 maintains all invariants and can be applied in time $O^*(2^{\mathrm{mw}(G)})$.

Proof. First, it is not hard to see that the rule can be applied in the claimed running time, since the only non-polynomial step is computing $\alpha(H[M_i])$, which can be done in $O^*(2^{\mathrm{mw}(G)})$.

Let us argue why the rule maintains all invariants. Invariants 1, 2, 5, 6, 7, 8 remain trivially satisfied if they were true before applying the rule. For invariant 4, we observe that $H[M_0]$ is composed of the (at most r) modules which have been turned into independent sets, therefore its neighborhood diversity is at most $\mathrm{mw}(G)$. For invariant 3, we invoke Lemma 3.3. The lemma applies because

$R \cap M_i$ is itself a maximum independent set of $H[M_i]$, and the lemma states that we can safely delete vertices of M_i outside this independent set. ♢

Claim A.2. Rule 2a maintains all invariants and can be applied in time $O^*(2^{\mathrm{mw}(G)})$.

Proof. Let us first make the useful observation that $R \cap M_0 = R \cap F_0$, which is a consequence of invariant 5: if R contained a vertex $u \in M_0 \setminus F_0$ then this vertex would have a neighbor in some non-empty M_i, therefore u would be connected to all of M_i (since M_i is a module), and R would not be independent since $R \cap M_i \neq \emptyset$.

By invariant 4 we have that $\mathrm{nd}(H[F_0]) \leq \mathrm{mw}(G)$. By Lemma 3.2 we can therefore compute λ_0 in time $O^*(2^{\mathrm{mw}(G)})$. Furthermore, the algorithm of the lemma returns to us a transformation $(R \cap F_0) \overset{H[F_0]}{\rightsquigarrow}_{t_0} R'$, with $|R'| = \lambda_0$. We perform the same transformation in H, keeping $R \setminus M_0 = R \setminus F_0$ unchanged and resulting in $R \cap F_0 = R'$. This is a $\mathsf{TAR}(k)$ transformation in H because by invariant 6, $t_0 + |R \setminus M_0| \geq k$. We therefore maintain invariant 2. Invariants 1,3,4,5,7,8 are unaffected by this rule (so remain satisfied). For invariant 6 observe that we have set $t_0 := \max\{k - |R \setminus F_0|, 0\} \geq k - |R \setminus M_0|$ and that $R \cap M_0$ increases and t_0 does not increase when we apply the rule. ♢

Claim A.3. Rule 2b maintains all invariants and can be applied in polynomial time.

Proof. First, observe that invariants 1,3,4,5 are unaffected. To ease notation, let $j := k - |R \setminus M_i|$.

Let us first deal with the easy case $j \leq 0$. Recall that during the preprocessing step we have calculated a maximum independent set of $G[V_i]$, call it A_i, and that by invariant 5 $H[M_i] = G[V_i]$. As a result, $\lambda_i = \lambda(G[V_i], R \cap M_i, 0) = |A_i|$ is a known value. We have set $t_i = 0$, so invariants 6,7 are trivially satisfied no matter how we modify $R \cap M_i$. We perform a (trivial) transformation that leaves $R \setminus M_i$ unchanged and first removes all tokens from $R \cap M_i$ and then adds all tokens of A_i. This is a $\mathsf{TAR}(k)$ transformation in G, as $|R \setminus M_i| \geq k$, so invariant 2 is satisfied, and invariant 8 is satisfied because A_i is a maximum independent set of $G[V_i]$. We therefore assume in the remainder that $j > 0$.

Let us first argue why λ_i can be computed in polynomial time. Again $H[M_i] = G[V_i]$ by invariant 5, so we want to compute $\lambda_i = \lambda(G[V_i], R \cap M_i, j)$. By invariant 7, we have a transformation $(R \cap M_i) \overset{G[V_i]}{\rightsquigarrow}_{t_i} (S \cap V_i)$ and by invariant 6 we have $t_i \geq k - |R \setminus M_i| = j$. Therefore, using Observation 2.1 we have $(R \cap M_i) \overset{G[V_i]}{\rightsquigarrow}_{j} (S \cap V_i)$. This means that $\lambda_i = \lambda(G[V_i], S \cap V_i, j)$. Therefore, we can find the value λ_i in the input we have been supplied. Furthermore, we have in the input an independent set $R_{i,j}$ that is $\mathsf{TAR}(j)$ reachable from $S \cap V_i$ in $G[V_i]$ and has $|R_{i,j}| = \lambda_i$; this set is also $\mathsf{TAR}(j)$ reachable from $R \cap M_i$, so we can perform a transformation that leaves $R \setminus M_i$ unchanged and results in $R \cap M_i = R_{i,j}$. This maintains invariant 2, as well as invariants 7 and 8. Finally, invariant 6 is satisfied by the new value of t_i since t_i may only decrease and $R \cap M_i$ increases. ♢

We are now ready to argue for the algorithm's correctness. First, we observe that by Claims A.1, A.2, A.3, all rules can be applied in time at most $O^*(2^{\mathrm{mw}(G)})$. Since all rules either delete a vertex of the graph or increase $|R|$, the algorithm runs in time $O^*(2^{\mathrm{mw}(G)})$. By invariant 2, the independent set R returned by the algorithm (when no rule can be applied) is $\mathsf{TAR}(k)$ reachable from S in G, therefore $\lambda(G, S, k) \geq |R|$. We therefore need to argue that $|R| \geq \lambda(G, S, k)$. By invariant 3, this is equivalent to $|R| \geq \lambda(H, R, k)$. In other words, we want to argue that in the graph H on which the algorithm may no longer apply any rules, it is impossible to reach an independent set T using $\mathsf{TAR}(k)$ moves such that $|T| > |R|$.

Let $M_0, M_1, \ldots, M_r$ be the partition of $V(H)$ at the last iteration of our algorithm. For an independent set T of H we will say that T is *interesting* if it satisfies at least one of the following two conditions: (a) for some $i \in \{0, \ldots, r\}$ we have $|T \cap M_i| > |R \cap M_i|$ or (b) for some $i \in [r]$ we have $R \cap M_i \neq \emptyset$ and $T \cap M_i = \emptyset$. In other words, an independent set is interesting if it manages to place more tokens than R in a module M_i (or in M_0), or if it manages to remove all tokens from a module M_i that is non-empty in R.

We will make two claims: (i) no interesting set is reachable from R with $\mathsf{TAR}(k)$ moves in H; (ii) if for an independent set T of H we have $|T| > |R|$, then T is interesting. Together the two claims imply that $|R| \geq \lambda(H, R, k)$, since all sets which are strictly larger than R are not reachable.

For claim (ii) suppose that T is not interesting, therefore for each $i \in \{0, \ldots, r\}$ $|T \cap M_i| \leq |R \cap M_i|$. Since $M_0, M_1, \ldots, M_r$ is a partition of $V(H)$ we have $|T| \leq |R|$.

For claim (i) suppose that T is interesting and $R \leftrightsquigarrow_k T$. Among all such sets T consider one whose shortest reconfiguration sequence from R has minimum length. Let $R_0 = R, R_1, \ldots, R_\ell = T$ be such a shortest reconfiguration sequence. Therefore, we have $\ell \geq 1$ and $R_0, \ldots, R_{\ell-1}$ are not interesting. We now consider the two possible reasons for which R_ℓ may be interesting.

In case there exists $i \in \{0, \ldots, r\}$ such that $|R_\ell \cap M_i| > |R \cap M_i|$ we construct a transformation from $R \cap M_i$ to $R_\ell \cap M_i$ in $H[M_i]$ by considering the sets $R_0 \cap M_i, R_1 \cap M_i, \ldots, R_\ell \cap M_i$. We note that this is a $\mathsf{TAR}(k - |R \setminus M_i|)$ transformation, since for all $j < \ell$ we have $|R_j \setminus M_i| \leq |R \setminus M_i|$ and $|R_j| \geq k$. But then Rule 2 could have been applied. In particular, if $i \neq 0$ the transformation proves that $\lambda_i = \lambda(H[M_i], R \cap M_i, k - |R \setminus M_i|) \geq |R_\ell \cap M_i| > |R \cap M_i|$ so we could have applied Rule 2b. If $i = 0$ we first make the observation that for all $j \in \{0, \ldots, \ell\}$ we have $R_j \cap M_0 = R_j \cap F_0$ (where F_0 is defined in Rule 2a). To see this we argue (as in Claim A.2) that if there exists $u \in R_j \cap (M_0 \setminus F_0)$ then u is connected to all of a module M_i which has $R \cap M_i \neq \emptyset$, and therefore $R_j \cap M_i \neq \emptyset$ (as R_j is not interesting for $j < \ell$) which contradicts the independence of R_j. We now have $\lambda_0 = \lambda(H[F_0], R \cap F_0, k - |R \setminus M_0|) \geq |R_\ell \cap F_0| > |R \cap F_0|$ and Rule 2a could have been applied.

In the case there exists $i \in [r]$ such that $R \cap M_i \neq \emptyset$ and $R_\ell \cap M_i = \emptyset$, this means that $|R_{\ell-1} \setminus M_i| \geq k$, which implies that $|R \setminus M_i| \geq k$, because $R_{\ell-1}$ is not interesting. If $|R \cap M_i| = \alpha(G[V_i])$, then rule 1 would have been applied,

so we assume $|R \cap M_i| < \alpha(G[V_i])$. However, $k - |R \setminus M_i| \leq 0$, so we have $\lambda_i = \lambda(H[M_i], R \cap M_i, k - |R \setminus M_i|) = \alpha(G[V_i]) > |R \setminus M_i|$, therefore rule 2b should have been applied.

Thus, in both cases we see that a rule could have been applied and we have a contradiction. Therefore, the set R returned is optimal. □

B Omitted Proofs in Section 4

Proof (Lemma 4.2). The if direction holds as a TS sequence in an induced subgraph is always valid in the original graph.

To show the only-if direction, assume that $S \overset{G}{\leftrightsquigarrow} S'$. Let $S_0, \ldots, S_\ell$ be a TS sequence from $S_0 = S$ to $S_\ell = S'$. It suffices to show that no independent set in the sequence contains a vertex in $N(M)$. Suppose to the contrary that i is the first index such that $S_i \cap N(M) \neq \emptyset$. Since $|S \cap M| \geq 2$, we have $S_i \cap M \neq \emptyset$. As M is a module, all vertices in M are adjacent to all vertices in $N(M)$. This contradicts that S_i is an independent set. Therefore, $S_i \cap N(M) = \emptyset$ for all i. □

Proof (Lemma 4.4). If $S \overset{G-v}{\leftrightsquigarrow} S'$, then $S \overset{G}{\leftrightsquigarrow} S'$ since $G-v$ is an induced subgraph of G.

To prove the only-if part, assume that $S \overset{G}{\leftrightsquigarrow} S'$. Let $S_0, \ldots, S_\ell$ be a TS sequence from $S_0 = S$ to $S_\ell = S'$. If $M \cap (S \cup S')$ is nonempty, let u be the unique vertex in $M \cap (S \cup S')$; otherwise, let u be an arbitrary vertex $M - v$. For $0 \leq i \leq \ell$, we define T_i as follows: if $S_i \cap M = \emptyset$, then $T_i = S_i$; otherwise, $T_i = (S_i \setminus M) + u$. Observe that $T_0 = S$ and $T_\ell = S'$. We show that $T_{i-1} \overset{G-v}{\leftrightsquigarrow} T_i$ for each $i \in [\ell]$. By Corollary 4.3, $|M \cap S_i| \leq 1$ holds for all i. This implies that $|T_i| = |S_i|$ and that T_i is an independent set of $G - v$ since $N_G(w) \setminus M = N_G(u) \setminus M$ for all $w \in M$.

If $|S_{i-1} \cap M| = |S_i \cap M| = 1$, then $T_{i-1} = (S_{i-1} \setminus M) + u = (S_i \setminus M) + u = T_i$, and thus $T_{i-1} \overset{G-v}{\leftrightsquigarrow} T_i$. If $S_{i-1} \cap M = S_i \cap M = \emptyset$, then $S_{i-1} = T_{i-1}$ and $S_i = T_i$, and thus $T_{i-1} \overset{G-v}{\leftrightarrow} T_i$. In the remaining case, we may assume by symmetry that $|S_{i-1} \cap M| = 1$ and $S_i \cap M = \emptyset$. Let $S_{i-1} \setminus S_i = x$ and $S_i \setminus S_{i-1} = y$. Observe that $x \in M$, $y \notin M$, and $\{x, y\} \in E(G)$. Since $T_{i-1} \setminus T_i = u$ and $T_i \setminus T_{i-1} = y$, we have $\{u, y\} \in E(G - v)$. Therefore, $T_{i-1} \overset{G-v}{\leftrightarrow} T_i$. □

Proof (Lemma 4.5). Let $P = (p_0, \ldots, p_q)$ be a u–v path in $G[M]$, where $p_0 = u$ and $p_q = v$. Let $T_0 = S' - v + u$ and $T_i = T_{i-1} - p_{i-1} + p_i$ for $i \geq 1$. Clearly, $|T_{i-1} \vartriangle T_i| = 2$ for $i \in [q]$. Each T_i is an independent set of G since T_0 is an independent set of G, $T_i = T_0 - u + p_i$, and $N_G(u) \setminus M = N_G(p_i) \setminus M$ as u and p_i are in the same module M. Since $T_{i-1} \vartriangle T_i = \{p_{i-1}, p_i\}$ and $\{p_{i-1}, p_i\} \in E(G)$ for each $i \in [q]$, we have $T_{i-1} \overset{G}{\leftrightarrow} T_i$. As $T_0 = S' - v + u$ and $T_q = S'$, we have $S' - v + u \overset{G}{\leftrightsquigarrow} S'$. Hence, we can conclude that $S \overset{G}{\leftrightsquigarrow} S'$ if and only if $S \overset{G}{\leftrightsquigarrow} S' - v + u$. On the other hand, since $|M| \geq 2$ and $M \cap (S \cup (S' - v + u))) = \{u\}$, Lemma 4.4 implies that $S \overset{G}{\leftrightsquigarrow} S' - v + u$ if and only if $S \overset{G-v}{\leftrightsquigarrow} S' - v + u$. Putting them together, we obtain that $S \overset{G}{\leftrightsquigarrow} S'$ if and only if $S \overset{G-v}{\leftrightsquigarrow} S' - v + u$. □

Proof (Lemma 4.6). We first show the only-if part. Assume that $S \overset{G}{\rightsquigarrow} S'$. Let $S_0, \ldots, S_\ell$ be a TS sequence from $S_0 = S$ to $S_\ell = S'$. Observe that there is an index i such that $S_i \cap M = \emptyset$. This is because, otherwise the vertex in $S_i \cap M$ is always in the component of $G[M]$ that contains u and thus cannot reach v. We set $T = S_i$. By Lemma 4.4, we have $S \overset{G-v}{\rightsquigarrow} T$. We also have $T \overset{G-v}{\rightsquigarrow} S - v + u$ by Lemma 4.4, and hence $S \overset{G-v}{\rightsquigarrow} S - v + u$.

To show the if part, assume that $S \overset{G-v}{\rightsquigarrow} S' - v + u$ and $S \overset{G-v}{\rightsquigarrow} T$ for some independent set T in $G - v$ with $T \cap M = \emptyset$. These assumptions imply that $S \overset{G}{\rightsquigarrow} T$ and $T \overset{G}{\rightsquigarrow} S' - v + u$. Let $S_0, \ldots, S_\ell$ be a TS sequence from $S_0 = T$ to $S_\ell = S' - v + u$ in $G - v$. For each i, we set $T_i = S_i$ if $S_i \cap M = \emptyset$; otherwise, we set $T_i = (S_i \setminus M) + v$. Note that $T_0 = T$ and $T_\ell = S'$. We show that $T_{i-1} \overset{G}{\rightsquigarrow} T_i$ for each $i \in [\ell]$. Corollary 4.3 implies that $|M \cap S_i| \leq 1$ holds for all i, and thus $|T_i| = |S_i|$. We can see that T_i is an independent set of G since $N_G(v) \setminus M = N_G(w) \setminus M$ for all $w \in M$.

If $|S_{i-1} \cap M| = |S_i \cap M| = 1$, then $T_{i-1} = T_i$, and thus $T_{i-1} \overset{G}{\rightsquigarrow} T_i$. If $S_{i-1} \cap M = S_i \cap M = \emptyset$, then $S_{i-1} = T_{i-1}$ and $S_i = T_i$, and thus $T_{i-1} \overset{G-v}{\leftrightarrow} T_i$. Now assume that $|S_{i-1} \cap M| = 1$ and $S_i \cap M = \emptyset$. Let $S_{i-1} \setminus S_i = x$ and $S_i \setminus S_{i-1} = y$. Observe that $x \in M$, $y \notin M$, and $\{x, y\} \in E(G)$. Since $T_{i-1} \setminus T_i = v$ and $T_i \setminus T_{i-1} = y$, we have $\{v, y\} \in E(G)$, and hence $T_{i-1} \overset{G}{\leftrightarrow} T_i$. Therefore, $T = T_0 \overset{G}{\rightsquigarrow} T_\ell = S'$ holds. Since $S \overset{G}{\rightsquigarrow} T$, we conclude that $S \overset{G}{\rightsquigarrow} S$. □

Proof (Theorem 4.7). We first check the size of each input and return "no" if $|S| \neq |T|$. We return "yes" if $|S| = |T| = 0$. Otherwise, we check the connectivity of G. If G is not connected, then we can solve each component independently by Observation 4.1. We return "yes" if and only if all executions on the components return "yes."

Now assume that G is connected. We compute a modular decomposition of G which gives a partition of V into $r \leq \mathrm{mw}(G)$ modules $M_1, \ldots, M_r$. We then check whether there is $i \in [r]$ such that $|S \cap M_i| \geq 2$. If such an i exists, then by Lemma 4.2, $S \overset{G}{\rightsquigarrow} T$ if and only if $T \cap N(M) = \emptyset$ and $S \overset{G-N(M)}{\rightsquigarrow} T$. Hence, if $T \cap N(M) \neq \emptyset$, then we return "no"; otherwise, we recursively check whether $S \overset{G-N(M)}{\rightsquigarrow} T$.

In the following, we assume that the instance is not caught by the tests above. That is, G is connected and $|S \cap M_i| \leq 1$ for each $i \in [r]$. We also assume that $|T \cap M_i| \leq 1$ holds for each $i \in [r]$, since otherwise the answer is "no" by Corollary 4.3.

We then exhaustively apply Lemmas 4.4, 4.5, 4.6 to remove "irrelevant" vertices. When we apply Lemma 4.6, we remember which module is involved as we need it later for the generalized reachability test. After all, we end up with a reduced instance where each module is of size 1 and the list of modules that are used for applying Lemma 4.6. Let H be the reduced graph with the vertex set $\{v_1, \ldots, v_r\}$ where $v_i \in M_i$, S', T' the modified independent sets, and $\mathcal{L}$ the list of modules used in Lemma 4.6. We construct the auxiliary graph $\mathcal{G}$ as follows:

we set the vertex set $V(\mathcal{G})$ to be the set of all size-$|S'|$ independent sets in H. Two sets $X, Y \in V(\mathcal{G})$ are adjacent in $\mathcal{G}$ if and only if $X \overset{H}{\leftrightarrow} Y$. We then compute the component $\mathcal{C}$ of $\mathcal{G}$ that contains S'. We return "yes" if

- $\mathcal{C}$ contains T', and
- for each module M_i in $\mathcal{L}$, there is $Z \in \mathcal{C}$ with $Z \cap M_i = \emptyset$ (equivalently, $v_i \notin Z$).

Otherwise, we return "no." Note that $|V(\mathcal{G})| \leq \binom{r}{|S'|} \leq 2^r$ and $|E(\mathcal{G})| \leq |V(\mathcal{G})| \cdot |V(H)| \leq 2^r r$.

The correctness of the algorithm follows directly from the correctness of facts used (that is, Observation 4.1, Lemmas 4.2, 4.4, 4.5, 4.6, and Corollary 4.3). We next consider the running time. The algorithm first reduces the instance to a collection of instances of size $O(r)$. This phase runs in time polynomial in the number of vertices of G. Then the algorithm solves each instance in time $O^*(2^r)$. Thus the total running time is $O^*(2^r)$. □