Reconfiguration over tree decompositions

Amer E. Mouawad[1⋆], Naomi Nishimura[1⋆], Venkatesh Raman[2], and Marcin Wrochna[3]

[1] David R. Cheriton School of Computer Science
University of Waterloo, Ontario, Canada.
{aabdomou, nishi}@uwaterloo.ca
[2] Institute of Mathematical Sciences
Chennai, India.
vraman@imsc.res.in
[3] Institute of Computer Science
Uniwersytet Warszawski, Warsaw, Poland.
mw290715@students.mimuw.edu.pl

Abstract. A vertex-subset graph problem Q defines which subsets of the vertices of an input graph are feasible solutions. The reconfiguration version of a vertex-subset problem Q asks whether it is possible to transform one feasible solution for Q into another in at most ℓ steps, where each step is a vertex addition or deletion, and each intermediate set is also a feasible solution for Q of size bounded by k. Motivated by recent results establishing W[1]-hardness of the reconfiguration versions of most vertex-subset problems parameterized by ℓ, we investigate the complexity of such problems restricted to graphs of bounded treewidth. We show that the reconfiguration versions of most vertex-subset problems remain PSPACE-complete on graphs of treewidth at most t but are fixed-parameter tractable parameterized by $\ell + t$ for all vertex-subset problems definable in monadic second-order logic (MSOL). To prove the latter result, we introduce a technique which allows us to circumvent cardinality constraints and define reconfiguration problems in MSOL.

1 Introduction

Reconfiguration problems allow the study of structural and algorithmic questions related to the solution space of computational problems, represented as a *reconfiguration graph* where feasible solutions are represented by nodes and adjacency by edges [6, 16, 18]; a path is equivalent to the step-by-step transformation of one solution into another as a *reconfiguration sequence* of *reconfiguration steps*.

Reconfiguration problems have so far been studied mainly under classical complexity assumptions, with most work devoted to deciding whether it is possible to find a path between two solutions. For several problems, this question has been shown to be PSPACE-complete [4, 18, 19], using reductions that construct

⋆ Research supported by the Natural Science and Engineering Research Council of Canada.

examples where the length ℓ of reconfiguration sequences can be exponential in the size of the input graph. It is therefore natural to ask whether we can achieve tractability if we allow the running time to depend on ℓ or on other properties of the problem, such as a bound k on the size of feasible solutions. These results motivated Mouawad et al. [22] to study reconfiguration under the *parameterized complexity* framework [12], showing the W[1]-hardness of Vertex Cover Reconfiguration (VC-R), Feedback Vertex Set Reconfiguration (FVS-R), and Odd Cycle Transversal Reconfiguration (OCT-R) parameterized by ℓ, and of Independent Set Reconfiguration (IS-R), Induced Forest Reconfiguration (IF-R), and Induced Bipartite Subgraph Reconfiguration (IBS-R) parameterized by $k + \ell$ [22].

Here we focus on reconfiguration problems restricted to $\mathcal{C}_t$, the class of graphs of treewidth at most t. In Section 3, we show that a large number of reconfiguration problems, including the six aforementioned problems, remain PSPACE-complete on $\mathcal{C}_t$, answering a question left open by Bonsma [5]. The result is in fact stronger in that it applies to graphs of bounded bandwidth and even to the question of finding a reconfiguration sequence of *any* length.

In Section 4, using an adaptation of Courcelle's cornerstone result [8], we present a meta-theorem proving that the reconfiguration versions of all vertex-subset problems definable in monadic second-order logic become tractable on $\mathcal{C}_t$ when parameterized by $\ell + t$. Since the running times implied by our meta-theorem are far from practical, we consider the reconfiguration versions of problems defined in terms of hereditary graph properties in Section 5. In particular, we first introduce signatures to succinctly represent reconfiguration sequences and define "generic" procedures on signatures which can be used to exploit the structure of nice tree decompositions. We use these procedures in Section 5.2 to design algorithms solving VC-R and IS-R in $\mathcal{O}^\star(4^\ell(t+1)^\ell)$ time (the $\mathcal{O}^\star$ notation suppresses factors polynomial in n, ℓ, and t). In Section 5.4, we extend the algorithms to solve OCT-R and IBS-R in $\mathcal{O}^\star(2^{\ell t}4^\ell(t+1)^\ell)$ time, as well as FVS-R and IF-R in $\mathcal{O}^\star(t^{\ell t}4^\ell(t+1)^\ell)$ time. We further demonstrate in Section 5.3 that VC-R and IS-R parameterized by ℓ can be solved in $\mathcal{O}^\star(4^\ell(3\ell+1)^\ell)$ time on planar graphs by an adaptation of Baker's shifting technique [1].

2 Preliminaries

For general graph theoretic definitions, we refer the reader to the book of Diestel [11]. We assume that each input graph G is a simple undirected graph with vertex set $V(G)$ and edge set $E(G)$, where $|V(G)| = n$ and $|E(G)| = m$. The *open neighborhood* of a vertex v is denoted by $N_G(v) = \{u \mid uv \in E(G)\}$ and the *closed neighborhood* by $N_G[v] = N_G(v) \cup \{v\}$. For a set of vertices $S \subseteq V(G)$, we define $N_G(S) = \{v \notin S \mid uv \in E(G), u \in S\}$ and $N_G[S] = N_G(S) \cup S$. We drop the subscript G when clear from context. The subgraph of G induced by S is denoted by $G[S]$, where $G[S]$ has vertex set S and edge set $\{uv \in E(G) \mid u, v \in S\}$. Given two sets $S_1, S_2 \subseteq V(G)$, we let $S_1 \Delta S_2 = \{S_1 \setminus S_2\} \cup \{S_2 \setminus S_1\}$ denote the symmetric difference of S_1 and S_2.

We say a graph problem Q is a *vertex-subset* problem whenever feasible solutions for Q on input G correspond to subsets of $V(G)$. Q is a *vertex-subset minimization (maximization)* problem whenever feasible solutions for Q correspond to subsets of $V(G)$ of size at most (at least) k, for some integer k. The *reconfiguration graph* of a vertex-subset minimization (maximization) problem Q, $R_{\text{MIN}}(G,k)$ ($R_{\text{MAX}}(G,k)$), has a node for each $S \subseteq V(G)$ such that $|S| \leq k$ ($|S| \geq k$) and S is a feasible solution for Q. We say k is the *maximum (minimum) allowed capacity* for $R_{\text{MIN}}(G,k)$ ($R_{\text{MAX}}(G,k)$). Nodes in a reconfiguration graph are adjacent if they differ by the addition or deletion of a single vertex.

Definition 1. *For any vertex-subset problem Q, graph G, positive integers k and ℓ, $S_s \subseteq V(G)$, and $S_t \subseteq V(G)$, we define four decision problems:*

- Q-MIN(G,k)*: Is there $S \subseteq V(G)$ such that $|S| \leq k$ and S is a feasible solution for Q?*
- Q-MAX(G,k)*: Is there $S \subseteq V(G)$ such that $|S| \geq k$ and S is a feasible solution for Q?*
- Q-MIN-R(G,S_s,S_t,k,ℓ)*: For $S_s, S_t \in V(R_{\text{MIN}}(G,k))$, is there a path of length at most ℓ between the nodes for S_s and S_t in $R_{\text{MIN}}(G,k)$?*
- Q-MAX-R(G,S_s,S_t,k,ℓ)*: For $S_s, S_t \in V(R_{\text{MAX}}(G,k))$, is there a path of length at most ℓ between the nodes for S_s and S_t in $R_{\text{MAX}}(G,k)$?*

For ease of description, we present our positive results for paths of length exactly ℓ, as all our algorithmic techniques can be generalized to shorter paths. Throughout, we implicitly consider reconfiguration problems as parameterized problems with ℓ as the parameter. The reader is referred to the books of Downey and Fellows [12], Flum and Grohe [15], and Niedermeier [23] for more on parameterized complexity.

In Section 5, we consider problems that can be defined using graph properties, where a *graph property* Π is a collection of graphs closed under isomorphism, and is *non-trivial* if it is non-empty and does not contain all graphs. A graph property is *polynomially decidable* if for any graph G, it can be decided in polynomial time whether G is in Π. The property Π is *hereditary* if for any $G \in \Pi$, any induced subgraph of G is also in Π. For a graph property Π, $R_{\text{MAX}}(G,k)$ has a node for each $S \subseteq V(G)$ such that $|S| \geq k$ and $G[S]$ has property Π, and $R_{\text{MIN}}(G,k)$ has a node for each $S \subseteq V(G)$ such that $|S| \leq k$ and $G[V(G) \setminus S]$ has property Π. We use Π-MIN-R and Π-MAX-R instead of Q-MIN-R and Q-MAX-R, respectively, to denote *reconfiguration problems for* Π; examples include VC-R, FVS-R, and OCT-R for the former and IS-R, IF-R, and IBS-R for the latter, for Π defined as the collection of all edgeless graphs, forests, and bipartite graphs, respectively.

Due to space limitations, most proofs have been omitted from the current version of the paper. The affected propositions, lemmas, and theorems have been marked with a star.

Proposition 1. *Given Π and a collection of graphs $\mathcal{C}$, if Π-MIN-R parameterized by ℓ is fixed-parameter tractable on $\mathcal{C}$ then so is Π-MAX-R.*

Proof. Given an instance (G, S_s, S_t, k, ℓ) of Π-Max-R, where $G \in \mathcal{C}$, we solve the Π-Min-R instance $(G, V(G)\setminus S_s, V(G)\setminus S_t, n-k, \ell)$. Note that the parameter ℓ remains unchanged.

It is not hard to see that there exists a path between the nodes corresponding to S_s and S_t in $R_{\text{MAX}}(G,k)$ if and only if there exists a path of the same length between the nodes corresponding to $V(G) \setminus S_s$ and $V(G) \setminus S_t$ in $R_{\text{MIN}}(G, n-k)$. □

We obtain our results by solving Π-Min-R, which by Proposition 1 implies results for Π-Max-R. We always assume Π to be non-trivial, polynomially decidable, and hereditary.

Our algorithms rely on dynamic programming over graphs of bounded treewidth. A *tree decomposition* of a graph G is a pair $\mathcal{T} = (T, \chi)$, where T is a tree and χ is a mapping that assigns to each node $i \in V(T)$ a vertex subset X_i (called a *bag*) such that: (1) $\bigcup_{i \in V(T)} X_i = V(G)$, (2) for every edge $uv \in E(G)$, there exists a node $i \in V(T)$ such that the bag $\chi(i) = X_i$ contains both u and v, and (3) for every $v \in V(G)$, the set $\{i \in V(T) \mid v \in X_i\}$ forms a connected subgraph (subtree) of T. The *width* of any tree decomposition $\mathcal{T}$ is equal to $\max_{i \in V(T)} |X_i| - 1$. The *treewidth* of a graph G, $tw(G)$, is the minimum width of a tree decomposition of G.

For any graph of treewidth t, we can compute a tree decomposition of width t and transform it into a nice tree decomposition of the same width in linear time [21], where a rooted tree decomposition $\mathcal{T} = (T, \chi)$ with root *root* of a graph G is a *nice tree decomposition* if each of its nodes is either (1) a leaf node (a node i with $|X_i| = 1$ and no children), (2) an introduce node (a node i with exactly one child j such that $X_i = X_j \cup \{v\}$ for some vertex $v \notin X_j$; v is said to be *introduced* in i), (3) a forget node (a node i with exactly one child j such that $X_i = X_j \setminus \{v\}$ for some vertex $v \in X_j$; v is said to be *forgotten* in i), or (4) a join node (a node i with two children p and q such that $X_i = X_p = X_q$). For node $i \in V(T)$, we use T_i to denote the subtree of T rooted at i and V_i to denote the set of vertices of G contained in the bags of T_i. Thus $G[V_{root}] = G$.

3 PSPACE-completeness

We define a simple intermediary problem that highlights the essential elements of a PSPACE-hard reconfiguration problem. Given a pair $H = (\Sigma, E)$, where Σ is an alphabet and $E \subseteq \Sigma^2$ a binary relation between symbols, we say that a word over Σ is an *H-word* if every two consecutive symbols are in the relation. If one looks at H as a digraph (possibly with loops), a word is an H-word if and only if it is a walk in H. The H-Word Reconfiguration problem asks whether two given H-words of equal length can be transformed into one another (in any number of steps) by changing one symbol at a time so that all intermediary steps are also H-words.

A *Thue system* is a pair (Σ, R), where Σ is a finite alphabet and $R \subseteq \Sigma^* \times \Sigma^*$ is a set of rules. A rule can be applied to a word by replacing one subword by

the other, that is, for two words $s, t \in \Sigma^*$, we write $s \leftrightarrow_R t$ if there is a rule $\{\alpha, \beta\} \in R$ and words $u, v \in \Sigma^*$ such that $s = u\alpha v$ and $t = u\beta v$. The reflexive transitive closure of this relation defines an equivalence relation $\leftrightarrow_R^*$, where words s, t are equivalent if and only if one can be reached from the other by repeated application of rules. The *word problem* of R is the problem of deciding, given two words $s, t \in \Sigma^*$, whether $s \leftrightarrow_R^* t$. A Thue system is called *c-balanced* if for each $\{\alpha, \beta\} \in R$ we have $|\alpha| = |\beta| = c$. The following fact is a folklore variant [2] of the classic proof of undecidability for general Thue systems [24].

Lemma 1 (*). *There exists a 2-balanced Thue system whose word problem is PSPACE-complete.*

A simple but technical reduction from Lemma 1 allows us to show the PSPACE-completeness of H-Word Reconfiguration. The simplicity of the problem statement allows for easy reductions to various reconfiguration problems, as exemplified in Theorem 1. Similar reductions apply to the reconfiguration versions of, e.g., k-Coloring [7] and Shortest Path [20] – a comprehensive discussion is available in an online manuscript by the fourth author [25].

Lemma 2 (*). *There exists a digraph H for which H-Word Reconfiguration is PSPACE-complete.*

Theorem 1. *There exists an integer b such that* VC-R*,* FVS-R*,* OCT-R*,* IS-R*,* IF-R*, and* IBS-R *are PSPACE-complete even when restricted to graphs of treewidth at most b.*

Proof. Let $H = (\Sigma, R)$ be the digraph obtained from Lemma 2. We show a reduction from H-Word Reconfiguration to VC-R.

For an integer n, we define G_n as follows. The vertex set contains vertices v_i^a for all $i \in \{1, \ldots, n\}$ and $a \in \Sigma$. Let $V_i = \{v_i^a \mid a \in \Sigma\}$ for $i \in \{1, \ldots, n\}$. The edge set of G_n contains an edge between every two vertices of V_i for $i \in \{1, \ldots, n\}$ and an edge $v_i^a v_{i+1}^b$ for all $(a, b) \notin R$ and $i \in \{1, \ldots, n-1\}$. The sets $V_i \cup V_{i+1}$ give a tree decomposition of width $b = 2|\Sigma|$.

Let $k = n \cdot (|\Sigma| - 1)$ and consider a vertex cover S of G_n of size k. For all i, since $G_n[V_i]$ is a clique, S contains all vertices of V_i except at most one. Since $|S| = \sum_i (|V_i| - 1)$, S contains all vertices except exactly one from each set V_i, say $v_i^{s_i}$ for some $s_i \in \Sigma$. Now $s_1 \ldots s_n$ is an H-word ($s_i s_{i+1} \in R$, as otherwise $v_i^{s_i} v_{i+1}^{s_{i+1}}$ would be an uncovered edge) and any H-word can be obtained in a similar way, giving a bijection between vertex covers of G_n of size k and H-words of length n.

Consider an instance $s, t \in \Sigma^*$ of H-Word Reconfiguration. We construct the instance $(G_n, S_s, S_t, k+1, \ell)$ of VC-R, where $n = |s| = |t|$, $\ell = 2^{n|\Sigma|}$ (that is, we ask for a reconfiguration sequence of any length) and S_s and S_t are the vertex covers of size k that correspond to s and t, respectively. Any reconfiguration sequence between such vertex covers starts by adding a vertex (since G_n has no vertex cover of size $k-1$) and then removing another (since vertex covers larger than $k+1$ are not allowed), which corresponds to changing one symbol of

an H-word. This gives a one-to-one correspondence between reconfiguration sequences of H-words and reconfiguration sequences (of exactly twice the length) between vertex covers of size k. The instances are thus equivalent.

This proof can be adapted to FVS-R and OCT-R by replacing edges with cycles, e.g. triangles [22]. For IS-R, IF-R, and IBS-R, we simply need to consider set complements of solutions for VC-R, FVS-R, and OCT-R, respectively. □

4 A meta-theorem

In contrast to Theorem 1, in this section we show that a host of reconfiguration problems definable in monadic second-order logic (MSOL) become fixed-parameter tractable when parameterized by $\ell + t$. First, we briefly review the syntax and semantics of MSOL over graphs. The reader is referred to the excellent survey by Martin Grohe [17] for more details.

We have an infinite set of *individual variables*, denoted by lowercase letters x, y, and z, and an infinite set of *set variables*, denoted by uppercase letters X, Y, and Z. A *monadic second-order formula (MSOL-formula)* ϕ over a graph G is constructed from *atomic formulas* $\mathcal{E}(x, y)$, $x \in X$, and $x = y$ using the usual Boolean connectives as well as existential and universal quantification over individual and set variables. We write $\phi(x_1, \ldots, x_r, X_1, \ldots, X_s)$ to indicate that ϕ is a formula with free variables $x_1, \ldots, x_r$ and $X_1, \ldots, X_s$, where free variables are variables not bound by quantifiers.

For a formula $\phi(x_1, \ldots, x_r, X_1, \ldots, X_s)$, a graph G, vertices $v_1, \ldots, v_r$, and sets $V_1, \ldots, V_r$, we write $G \models \phi(v_1, \ldots, v_r, V_1, \ldots, V_r)$ if ϕ is satisfied in G when $\mathcal{E}$ is interpreted by the adjacency relation $E(G)$, the variables x_i are interpreted by v_i, and variables X_i are interpreted by V_i. We say that a vertex-subset problem Q is *definable in monadic second-order logic* if there exists an MSOL-formula $\phi(X)$ with one free set variable such that $S \subseteq V(G)$ is a feasible solution of problem Q for instance G if and only if $G \models \phi(S)$. For example, an independent set is definable by the formula $\phi_{\text{IS}}(X) = \forall_x \forall_y (x \in X \wedge y \in X) \rightarrow \neg\mathcal{E}(x, y)$.

Theorem 2 (Courcelle [8]). *There is an algorithm that given a MSOL-formula $\phi(x_1, \ldots, x_r, X_1, \ldots, X_s)$, a graph G, vertices $v_1, \ldots, v_r \in V(G)$, and sets $V_1, \ldots, V_s \subseteq V(G)$ decides whether $G \models \phi(v_1, \ldots, v_r, V_1, \ldots, V_s)$ in $\mathcal{O}(f(tw(G), |\phi|) \cdot n)$ time, for some computable function f.*

Theorem 3. *If a vertex-subset problem Q is definable in monadic second-order logic by a formula $\phi(X)$, then Q-Min-R and Q-Max-R parameterized by $\ell + tw(G) + |\phi|$ are fixed-parameter tractable.*

Proof. We provide a proof for Q-Min-R as the proof for Q-Max-R is analogous. Given an instance (G, S_s, S_t, k, ℓ) of Q-Min-R, we build an MSOL-formula $\omega(X_0, X_\ell)$ such that $G \models \omega(S_s, S_t)$ if and only if the corresponding instance is a yes-instance. Since the size of ω will be bounded by a function of $\ell + |\phi|$, the statement will follow from Theorem 2.

As MSOL does not allow cardinality constraints, we overcome this limitation using the following technique. We let $L \subseteq \{-1,+1\}^\ell$ be the set of all sequences of length ℓ over $\{-1,+1\}$ which do not violate the maximum allowed capacity. In other words, given S_s and k, a sequence σ is in L if and only if for all $\ell' \leq \ell$ it satisfies $|S_s| + \sum_{i=1}^{\ell'} \sigma[i] \leq k$, where $\sigma[i]$ is the i^{th} element in sequence σ. We let $\omega = \bigvee_{\sigma \in L} \omega_\sigma$ and

$$\omega_\sigma(X_0, X_\ell) = \exists_{X_1,\ldots,X_{\ell-1}} \bigwedge_{0 \leq i \leq \ell} \phi(X_i) \wedge \bigwedge_{1 \leq i \leq \ell} \psi_{\sigma[i]}(X_{i-1}, X_i)$$

where $\psi_{-1}(X_{i-1}, X_i)$ means X_i is obtained from X_{i-1} by removing one element and $\psi_{+1}(X_{i-1}, X_i)$ means it is obtained by adding one element. Formally, we have:

$$\psi_{-1}(X_{i-1}, X_i) = \exists_x\ x \in X_{i-1} \wedge x \notin X_i \wedge \forall y\,(y \in X_i \leftrightarrow (y \in X_{i-1} \wedge y \neq x))$$

$$\psi_{+1}(X_{i-1}, X_i) = \exists_x\ x \notin X_{i-1} \wedge x \in X_i \wedge \forall y\,(y \in X_i \leftrightarrow (y \in X_{i-1} \vee y = x))$$

It is easy to see that $G \models \omega_\sigma(S_s, S_t)$ if and only if there is a reconfiguration sequence from S_s to S_t (corresponding to $X_0, X_1, \ldots, X_\ell$) such that the i^{th} step removes a vertex if $\sigma[i] = -1$ and adds a vertex if $\sigma[i] = +1$. Since $|L| \leq 2^\ell$, the size of the MSOL-formula ω is bounded by an (exponential) function of $\ell + |\phi|$. □

5 Dynamic programming algorithms

Throughout this section we will consider one fixed instance (G, S_s, S_t, k, ℓ) of Π-Min-R and a nice tree decomposition $\mathcal{T} = (T, \chi)$ of G. Moreover, similarly to the previous section, we will ask, for a fixed sequence $\sigma \in \{-1,+1\}^\ell$, whether $G \models \omega_\sigma(S_s, S_t)$ holds. That is, we ask whether there is a reconfiguration sequence which at the i^{th} step removes a vertex when $\sigma[i] = -1$ and adds a vertex when $\sigma[i] = +1$. The final algorithm then asks such a question for every sequence σ which does not violate the maximum allowed capacity: $|S_s| + \sum_{i=1}^{\ell'} \sigma[i] \leq k$ for all $\ell' \leq \ell$. This will add a factor of at most 2^ℓ to the running time.

5.1 Signatures as equivalence classes

A reconfiguration sequence can be described as a sequence of steps, each step specifying which vertex is being removed or added. To obtain a more succinct representation, we observe that in order to propagate information up from the leaves to the root of a nice tree decomposition, we can ignore vertices outside of the currently considered bag (X_i) and only indicate whether a step has been used by a vertex in any previously processed bags, i.e. a vertex in $V_i \setminus X_i$.

Definition 2. *A* signature τ *over a set* $X \subseteq V(G)$ *is a sequence of steps* $\tau[1], \ldots, \tau[\ell] \in X \cup \{\texttt{used}, \texttt{unused}\}$. *Steps from* X *are called* vertex steps.

The total number of signatures over a bag X of at most t vertices is $(t+2)^\ell$. Our dynamic programming algorithms start by considering a signature with only `unused` steps in each leaf node, specify when a vertex may be added/removed in introduce nodes by replacing `unused` steps with vertex steps ($\tau[i] =$ `unused` becomes $\tau[i] = v$ for the introduced vertex v), merge signatures in join nodes, and replace vertex steps with `used` steps in forget nodes.

For a set $S \subseteq V(G)$ and a bag X, we let $\tau(i, S) \subseteq S \cup X$ denote the set of vertices obtained after executing the first i steps of τ: the i^{th} step adds $\tau[i]$ if $\tau[i] \in X$ and $\sigma[i] = +1$, removes it if $\tau[i] \in X$ and $\sigma[i] = -1$, and does nothing if $\tau[i] \in \{\texttt{used}, \texttt{unused}\}$.

A valid signature must ensure that no step deletes a vertex that is absent or adds a vertex that is already present, and that the set of vertices obtained after applying reconfiguration steps to $S_s \cap X$ is the set $S_t \cap X$. Additionally, because Π is hereditary, we can check whether this property is at least locally satisfied (in $G[X]$) after each step of the sequence. More formally, we have the following definition.

Definition 3. *A signature τ over X is* valid *if*

(1) $\tau[i] \in \tau(i-1, S_s \cap X)$ *whenever* $\tau[i] \in X$ *and* $\sigma[i] = -1$,
(2) $\tau[i] \notin \tau(i-1, S_s \cap X)$ *whenever* $\tau[i] \in X$ *and* $\sigma[i] = +1$,
(3) $\tau(\ell, S_s \cap X) = S_t \cap X$, *and*
(4) $G[X \setminus \tau(i, S_s \cap X)] \in \Pi$ *for all* $i \leq \ell$.

It is not hard to see that a signature τ over X is valid if and only if $\tau(0, S_s \cap X), \ldots, \tau(\ell, S_s \cap X)$ is a well-defined path between $S_s \cap X$ and $S_t \cap X$ in $R_{\text{MIN}}(G[X], n)$. We will consider only valid signatures. The dynamic programming algorithms will enumerate exactly the signatures that can be extended to valid signatures over V_i in the following sense:

Definition 4. *A signature π over V_i* extends *a signature π over X_i if it is obtained by replacing some* `used` *steps with vertex steps from $V_i \setminus X_i$*

However, for many problems, the fact that S is a solution for $G[X]$ for each bag X does not imply that S is a solution for G, and checking this 'local' notion of validity will not be enough – the algorithm will have to maintain additional information. One such example is the OCT-R problem, which we discuss in Section 5.4.

5.2 An algorithm for VC-R

To process nodes of the tree decomposition, we now define ways of generating signatures from other signatures. The introduce operation determines all ways that an introduced vertex can be represented in a signature, replacing `unused` steps in the signature of its child.

Definition 5. *Given a signature τ over X and a vertex $v \notin X$, the* introduce operation*, introduce(τ, v) returns the following set of signatures over $X \cup \{v\}$:*

for every subset I of indices i for which $\tau[i] = \texttt{unused}$, consider a copy τ' of τ where for all $i \in I$ we set $\tau'[i] = v$, check if it is valid, and if so, add it to the set.

In particular $\tau \in \mathit{introduce}(\tau, v)$ and $|\mathit{introduce}(\tau, v)| \leq 2^\ell$. All signatures obtained through the introduce operation are valid, because of the explicit check.

Definition 6. *Given a signature τ over X and a vertex $v \in X$, the* forget operation*, returns a new signature $\tau' = \mathit{forget}(\tau, v)$ over $X \setminus \{v\}$ such that for all $i \leq \ell$, we have $\tau'[i] = \texttt{used}$ if $\tau[i] = v$ and $\tau'[i] = \tau[i]$ otherwise.*

Since $\tau'(i, S_s \cap X \setminus \{v\}) = \tau(i, S_s \cap X) \setminus \{v\}$, it is easy to check that the forget operation preserves validity.

Definition 7. *Given two signatures τ_1 and τ_2 over $X \subseteq V(G)$, we say τ_1 and τ_2 are* compatible *if for all $i \leq \ell$:*
(1) $\tau_1[i] = \tau_2[i] = \texttt{unused}$,
(2) $\tau_1[i] = \tau_2[i] = v$ for some $v \in X$, or
(3) either $\tau_1[i]$ or $\tau_2[i]$ is equal to used *and the other is equal to* unused*.*
For two compatible signatures τ_1 and τ_2, the join operation *returns a new signature $\tau' = \mathit{join}(\tau_1, \tau_2)$ over X such that for all $i \leq \ell$ we have, respectively:*
(1) $\tau'[i] = \texttt{unused}$,
(2) $\tau'[i] = v$, and
(3) $\tau'[i] = \texttt{used}$.

Since $\tau' = \mathit{join}(\tau_1, \tau_2)$ is a signature over the same set as τ_1 and differs from τ_1 only by replacing some unused steps with used steps, the join operation preserves validity, that is, if two compatible signatures τ_1 and τ_2 are valid then so is $\tau' = \mathit{join}(\tau_1, \tau_2)$.

Let us now describe the algorithm. For each $i \in V(T)$ we assign an initially empty table A_i. All tables corresponding to internal nodes of T will be updated by simple applications of the introduce, forget, and join operations.

Leaf nodes. Let i be a leaf node, that is $X_i = \{v\}$ for some vertex v. We let $A_i = \mathit{introduce}(\tau, v)$, where τ is the signature with only unused steps.

Introduce nodes. Let j be the child of an introduce node i, that is $X_i = X_j \cup \{v\}$ for some $v \notin X_j$. We let $A_i = \bigcup_{\tau \in A_j} \mathit{introduce}(\tau, v)$.

Forget nodes. Let j be the child of a forget node i, that is $X_i = X_j \setminus \{v\}$ for some $v \in X_j$. We let $A_i = \{\mathit{forget}(\tau, v) \mid \tau \in A_j\}$.

Join nodes. Let j and h be the children of a join node i, that is $X_i = X_j = X_h$. We let $A_i = \{\mathit{join}(\tau_j, \tau_h) \mid \tau_j \in A_j, \tau_h \in A_h,$ and τ_j is compatible with $\tau_h\}$.

The operations were defined so that the following lemma holds by induction. The theorem then follows by making the algorithm accept when A_{root} contains a signature τ such that no step of τ is unused.

Lemma 3. *For $i \in V(T)$ and a signature τ over X_i, $\tau \in A_i$ if and only if τ can be extended to a signature over V_i that is valid.*

Theorem 4 (*). VC-R *and* IS-R *can be solved in $\mathcal{O}^\star(4^\ell(t+1)^\ell)$ time on graphs of treewidth t.*

5.3 VC-R in planar graphs

Using an adaptation of Baker's approach for decomposing planar graphs [1], also known as the *shifting technique* [3, 10, 13], we show a similar result for VC-R and IS-R on planar graphs. The idea is that at most ℓ elements of a solution will be changed, and thus if we divide the graph into $\ell + 1$ parts, one of these parts will be unchanged throughout the reconfiguration sequence. The shifting technique allows the definition of the $\ell+1$ parts so that removing one (and replacing it with simple gadgets to preserve all needed information) yields a graph of treewidth at most $3\ell - 1$.

Theorem 5 (*). VC-R *and* IS-R *are fixed-parameter tractable on planar graphs when parameterized by ℓ. Moreover, there exists an algorithm which solves both problems in $\mathcal{O}^\star(4^\ell(3\ell+1)^\ell)$ time.*

We note that, by a simple application of the result of Demaine et al. [9], Theorem 5 generalizes to H-minor-free graphs and only the constants of the overall running time of the algorithm are affected.

5.4 An algorithm for OCT-R

In this section we show how known dynamic programming algorithms for problems on graphs of bounded treewidth can be adapted to reconfiguration. The general idea is to maintain a view of the reconfiguration sequence just as we did for VC-R and in addition check if every reconfiguration step gives a solution, which can be accomplished by maintaining (independently for each step) any information that the original algorithm would maintain. We present the details for OCT-R (where Π is the collection of all bipartite graphs) as an example.

In a dynamic programming algorithm for VC on graphs of bounded treewidth, it is enough to maintain information about what the solution's intersection with the bag can be. This is not the case for OCT. One algorithm for OCT works in time $\mathcal{O}^\star(3^t)$ by additionally maintaining a bipartition of the bag (with the solution deleted) [14, 15]. That is, at every bag X_i, we would maintain a list of assignments $X \to \{\texttt{used}, \texttt{left}, \texttt{right}\}$ with the property that there exists a subset S of V_i and a bipartition L, R of $G[V_i \setminus S]$ such that $X_i \cap S, X_i \cap L$, and $X_i \cap R$ are the `used`, `left`, and `right` vertices, respectively. A signature for OCT-R will hence additionally store a bipartition for each step (except for the first and last sets S_s and S_t, as we already assume them to be solutions).

Definition 8. *An* OCT-signature τ *over a set $X \subseteq V(G)$ is a sequence of steps $\tau[1], \ldots, \tau[\ell] \in X \cup \{\texttt{used}, \texttt{unused}\}$ together with an entry $\tau[i, v] \in \{\texttt{left}, \texttt{right}\}$ for every $1 \leq i \leq \ell - 1$ and $v \in X \setminus \tau(i, S_s \cap X)$.*

There are at most $(t+2)^\ell 2^{t(\ell-1)}$ different OCT-signatures. In the definition of validity, we replace the last condition with the following, stronger one:

(4) For all $1 \leq i \leq \ell - 1$, the sets $\{v \mid \tau[i, v] = \texttt{left}\}$ and $\{v \mid \tau[i, v] = \texttt{right}\}$ give a bipartition of $G[X \setminus \tau(p, S_s \cap X)]$.

In the definition of the join operation, we additionally require two signatures to have equal $\tau[i, v]$ entries (whenever defined) to be considered compatible; the operation copies them to the new signature. In the definition of the forget operation, we delete any $\tau[i, v]$ entries, where v is the vertex being forgotten. In the introduce operation, we consider (and check the validity of) a different copy for each way of replacing `unused` steps with v steps and each way of assigning $\{\texttt{left}, \texttt{right}\}$ values to new $\tau[i, v]$ entries, where v is the vertex being introduced. As before, to each node we assign an initially empty table of OCT-signatures and fill them bottom-up using these operations. Lemma 3, with the new definitions, can then be proved again by induction.

Theorem 6 (*). OCT-R *and* IBS-R *can be solved in* $\mathcal{O}^\star(2^{t\ell}4^\ell(t+1)^\ell)$ *time on graphs of treewidth* t*.*

Similarly, using the classical $\mathcal{O}^\star(2^{\mathcal{O}(t \log t)})$ algorithm for FVS and IF (which maintains what partition of X_i the connected components of V_i can produce), we get the following running times for reconfiguration variants of these problems.

Theorem 7 (*). FVS-R *and* IF-R *can be solved in* $\mathcal{O}^\star(t^{\ell t}4^\ell(t+1)^\ell)$ *time on graphs of treewidth* t*.*

6 Conclusion

We have seen in Section 5.4 that, with only minor modifications, known dynamic programming algorithms for problems on graphs of bounded treewidth can be adapted to reconfiguration. It is therefore natural to ask whether the obtained running times can be improved via more sophisticated algorithms which exploit properties of the underlying problem or whether these running times are optimal under some complexity assumptions. Moreover, it would be interesting to investigate whether the techniques presented for planar graphs can be extended to other problems or more general classes of sparse graphs. In particular, the parameterized complexity of "non-local" reconfiguration problems such as FVS-R and OCT-R remains open even for planar graphs.

References

1. B. S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *J. ACM*, 41(1):153–180, Jan. 1994.
2. G. Bauer and F. Otto. Finite complete rewriting systems and the complexity of the word problem. *Acta Informatica*, 21(5):521–540, Dec. 1984.
3. H. L. Bodlaender and A. M. C. A. Koster. Combinatorial optimization on graphs of bounded treewidth. *Comput. J.*, 51(3):255–269, May 2008.
4. P. Bonsma. The complexity of rerouting shortest paths. In *Proc. of Mathematical Foundations of Computer Science*, pages 222–233, 2012.
5. P. Bonsma. Rerouting shortest paths in planar graphs. In *FSTTCS 2012*, volume 18 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 337–349, 2012.

6. L. Cereceda, J. van den Heuvel, and M. Johnson. Connectedness of the graph of vertex-colourings. *Discrete Mathematics*, 308(56):913–919, 2008.
7. L. Cereceda, J. van den Heuvel, and M. Johnson. Finding paths between 3-colorings. *Journal of Graph Theory*, 67(1):69–82, 2011.
8. B. Courcelle. The monadic second-order logic of graphs. I. recognizable sets of finite graphs. *Information and Computation*, 85(1):12 – 75, 1990.
9. E. Demaine, M. Hajiaghayi, and K. Kawarabayashi. Algorithmic graph minor theory: secomposition, approximation, and coloring. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 637–646, Oct 2005.
10. E. D. Demaine and M. Hajiaghayi. The bidimensionality theory and its algorithmic applications. *Comput. J.*, 51(3):292–302, 2008.
11. R. Diestel. *Graph Theory*. Springer-Verlag, Electronic Edition, 2005.
12. R. G. Downey and M. R. Fellows. *Parameterized complexity*. Springer-Verlag, 1997.
13. D. Eppstein. Diameter and treewidth in minor-closed graph families. *Algorithmica*, 27(3):275–291, 2000.
14. S. Fiorini, N. Hardy, B. Reed, and A. Vetta. Planar graph bipartization in linear time. *Discrete Appl. Math.*, 156(7):1175–1180, Apr. 2008.
15. J. Flum and M. Grohe. *Parameterized complexity theory*. Springer-Verlag, 2006.
16. P. Gopalan, P. G. Kolaitis, E. N. Maneva, and C. H. Papadimitriou. The connectivity of boolean satisfiability: computational and structural dichotomies. *SIAM J. Comput.*, 38(6):2330–2355, 2009.
17. M. Grohe. Logic, graphs, and algorithms. *Electronic Colloquium on Computational Complexity (ECCC)*, 14(091), 2007.
18. 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.
19. T. Ito, M. Kamiński, and E. D. Demaine. Reconfiguration of list edge-colorings in a graph. *Discrete Applied Mathematics*, 160(15):2199–2207, 2012.
20. M. Kamiński, P. Medvedev, and M. Milanič. Shortest paths between shortest paths. *Theor. Comput. Sci.*, 412(39):5205–5210, 2011.
21. T. Kloks. *Treewidth, Computations and Approximations*, volume 842 of *Lecture Notes in Computer Science*. Springer, 1994.
22. A. E. Mouawad, N. Nishimura, V. Raman, N. Simjour, and A. Suzuki. On the parameterized complexity of reconfiguration problems. In *Proc. of the 8th International Symposium on Parameterized and Exact Computation*, pages 281–294, 2013.
23. R. Niedermeier. *Invitation to fixed-parameter algorithms*. Oxford University Press, 2006.
24. E. L. Post. Recursive unsolvability of a problem of Thue. *The Journal of Symbolic Logic*, 12(1):1–11, 1947.
25. M. Wrochna. Reconfiguration in bounded bandwidth and treedepth, 2014. arXiv:1405.0847.