Reconfiguration over tree decompositions

Amer E. Mouawad\textsuperscript{1*}, Naomi Nishimura\textsuperscript{1*}, Venkatesh Raman\textsuperscript{2}, and Marcin Wrochna\textsuperscript{3}

\textsuperscript{1} David R. Cheriton School of Computer Science
University of Waterloo, Ontario, Canada.
{aabdoumou, nishi}@uwaterloo.ca
\textsuperscript{2} Institute of Mathematical Sciences
Chennai, India.
vraman@imsc.res.in
\textsuperscript{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 $\ell$ 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 $\ell$, 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 \cite{7,17,19}; 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 \cite{5,19,20}, using reductions that construct

\* Research supported by the Natural Science and Engineering Research Council of Canada.
examples where the length \( \ell \) 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 \( \ell \) or on other properties of the problem, such as a bound \( k \) on the size of feasible solutions. These results motivated Mouawad et al. [23] to study reconfiguration under the parameterized complexity framework [13], showing the W[1]-hardness of \textsc{Vertex Cover Reconfiguration} (VC-R), \textsc{Feedback Vertex Set Reconfiguration} (FVS-R), and \textsc{Odd Cycle Transversal Reconfiguration} (OCT-R) parameterized by \( \ell \), and of \textsc{Independent Set Reconfiguration} (IS-R), \textsc{Induced Forest Reconfiguration} (IF-R), and \textsc{Induced Bipartite Subgraph Reconfiguration} (IBS-R) parameterized by \( k + \ell \) [23].

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 [6]. 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 [9], 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 \( \mathcal{O}^*(4 \ell(t+3)^\ell) \) time (the \( \mathcal{O}^* \) notation suppresses factors polynomial in \( n, \ell, \) and \( t \)). In Section 5.4, we extend the algorithms to solve OCT-R and IBS-R in \( \mathcal{O}^*(2\ell t^{\ell}(t+3)^\ell) \) time, as well as FVS-R and IF-R in \( \mathcal{O}^*(\ell t^{\ell\ell}(t+3)^\ell) \) time. We further demonstrate in Section 5.3 that VC-R and IS-R parameterized by \( \ell \) can be solved in \( \mathcal{O}^*(4\ell(3\ell+2)^\ell) \) 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 [12]. 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 \( N_G(S) = \{ v \not\in S \mid uv \in E(G), u \in S \} \) 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 $\ell$, $S_s \subseteq V(G)$, 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($G,S_s,S_t,k,\ell$): For $S_s,S_t \in V(R_{\text{MIN}}(G,k))$, is there a path of length at most $\ell$ between the nodes for $S_s$ and $S_t$ in $R_{\text{MIN}}(G,k)$?
- $Q$-MAX-R($G,S_s,S_t,k,\ell$): For $S_s,S_t \in V(R_{\text{MAX}}(G,k))$, is there a path of length at most $\ell$ 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 $\ell$, as all our algorithmic techniques can be generalized to shorter paths. Throughout, we implicitly consider reconfiguration problems as parameterized problems with $\ell$ as the parameter. The reader is referred to the books of Downey and Fellows [13], Flum and Grohe [16], and Niedermeier [24] for more on parameterized complexity.

In Section 5, we consider problems that can be defined using graph properties, where a graph property $\Pi$ 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 $\Pi$. The property $\Pi$ is hereditary if for any $G \in \Pi$, any induced subgraph of $G$ is also in $\Pi$. For a graph property $\Pi$, $R_{\text{MAX}}(G,k)$ has a node for each $S \subseteq V(G)$ such that $|S| \geq k$ and $G[S]$ has property $\Pi$, 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 $\Pi$. We use $\Pi$-MIN-R and $\Pi$-MAX-R instead of $Q$-MIN-R and $Q$-MAX-R, respectively, to denote reconfiguration problems for $\Pi$; examples include VC-R, FVS-R, and OCT-R for the former and IS-R, IF-R, and IBS-R for the latter, for $\Pi$ defined as the collection of all edgeless graphs, forests, and bipartite graphs, respectively.

Proofs of propositions, lemmas, and theorems marked with a star can be found in the appendix.

**Proposition 2.** Given $\Pi$ and a collection of graphs $\mathcal{C}$, if $\Pi$-MIN-R parameterized by $\ell$ is fixed-parameter tractable on $\mathcal{C}$ then so is $\Pi$-MAX-R.
Proof. Given an instance \((G, S_s, S_t, k, \ell)\) of \(\Pi\text{-MAX-R}\), where \(G \in \mathcal{E}\), we solve the \(\Pi\text{-MIN-R}\) instance \((G, V(G) \setminus S_s, V(G) \setminus S_t, n-k, \ell)\). Note that the parameter \(\ell\) remains unchanged.

It is not hard to see that there exists a path between the nodes corresponding to \(S_s\) and \(S_t\) in \(R_{\max}(G, k)\) 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 \(R_{\min}(G, n-k)\).

We obtain our results by solving \(\Pi\text{-MIN-R}\), which by Proposition 2 implies results for \(\Pi\text{-MAX-R}\). We always assume \(\Pi\) 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 \(\chi\) 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 [22], where a rooted tree decomposition \(\mathcal{T} = (T, \chi)\) with root \(\text{root}\) 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 \(X_i = X_p \times X_q\)).

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 \(G[V_{\text{root}}] = G\).

## 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 \(\Sigma\) is an alphabet and \(E \subseteq \Sigma^2\) a binary relation between symbols, we say that a word over \(\Sigma\) 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 \(H\text{-WORD RECONFIGURATION}\) 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 \((\Sigma, R)\), where \(\Sigma\) 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 vertex covers of size \( k \) (that is, we ask for a reconfiguration sequence of any length) and \( S \) is a saturation sequence between such vertex covers starts by adding a vertex (since \( k \) larger than \( G \) has no vertex cover of size \( n \) giving a bijection between vertex covers of \( G \) and an edge \( v \) \( H \) would be an uncovered edge) and any \( \mid \Sigma \mid \) edge set of \( G \) since \( \Sigma \) contains all vertices of \( \Sigma \) except at most one. Since \( \mid \Sigma \mid = 2|\Sigma| \), 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 \( \mid S \mid = \sum_i (|V_i| - 1) \), \( S \) contains all vertices except exactly one from each set \( V_i \), say \( v_i^a \) 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 \( v_i^a v_{i+1}^b \) 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 \( n = \mid s \mid = \mid t \mid, \ell = 2|\Sigma| \) (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 [23]. 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. \hfill $\Box$

4 A meta-theorem

In contrast to Theorem 5, 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 [18] 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) $\phi$ over a graph $G$ is constructed from atomic formulas $E(x,y)$, $x \in X$, 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 $\phi$ 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 $\phi$ is satisfied in $G$ when $E$ 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 $\phi_{\text{IS}}(X) = \forall x \forall y (x \in X \land y \in X) \rightarrow \neg E(x, y)$.

Theorem 6 (Courcelle [9]). 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 $O(f(tw(G), |\phi|) \cdot n)$ time, for some computable function $f$.

Theorem 7. 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 $(G, S_\ell, S_\ell, k, \ell)$ of $Q$-MIN-R, we build an MSOL-formula $\omega(X_0, X_\ell)$ such that $G \models \omega(S_\ell, S_\ell)$ if and only if the corresponding instance is a yes-instance. Since the size of $\omega$ will be bounded by a function of $\ell + |\phi|$, the statement will follow from Theorem 6.
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 \( \ell \) over \( \{-1, +1\} \) which do not violate the maximum allowed capacity. In other words, given \( S_s \) and \( k \), a sequence \( \sigma \) 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 \( \sigma \). We let \( \omega = \bigvee_{\sigma \in L} \omega_{\sigma} \) and

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

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:

\[
\psi_{-1}(X_{i-1}, X_i) = \exists x \ x \in X_{i-1} \land x \not\in X_i \land \forall y (y \in X_i \leftrightarrow (y \in X_{i-1} \land y \neq x))
\]

\[
\psi_{+1}(X_{i-1}, X_i) = \exists x \ x \not\in X_{i-1} \land x \in X_i \land \forall y (y \in X_i \leftrightarrow (y \in X_{i-1} \lor y = x))
\]

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 \( \omega \) is bounded by an (exponential) function of \( \ell + |\phi| \).

\[\square\]

5 Dynamic programming algorithms

Throughout this section we will consider one fixed instance \((G, S_s, S_t, k, \ell)\) of \( \Pi\text{-Min-R} \) 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 \( \sigma \) 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 \( 2^\ell \) 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 8.** A signature \( \tau \) over a set \( X \subseteq V(G) \) is a sequence of steps \( \tau[1], \ldots, \tau[\ell] \in X \cup \{\text{used, unused}\} \). Steps from \( X \) are called vertex steps.
The total number of signatures over a bag $X$ of at most $t$ vertices is $(t + 3)\ell$. 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 ($\tau[i] = \text{unused}$ 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 $\tau(i, S) \cup X$ denote the set of vertices obtained after executing the first $i$ steps of $\tau$: 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 $\tau[i] \in \{\text{used}, \text{unused}\}$.

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 $S \cap X$ is the set $S_t \cap X$. Additionally, because $\Pi$ 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 9.** A signature $\tau$ 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. $\tau[i] \not\in \tau(i - 1, S_s \cap X)$ whenever $\tau[i] \in X$ and $\sigma[i] = +1$,
3. $\tau(\ell, S_t \cap X) = S_t \cap X$, 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 $\tau$ 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 $R_{\min}(G[X], n)$. 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 10.** A signature $\pi$ over $V_i$ extends a signature $\pi$ 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 11.** Given a signature $\tau$ over $X$ and a vertex $v \not\in X$, the introduce operation, $\text{introduce}(\tau, v)$ returns the following set of signatures over $X \cup \{v\}$:
for every subset $I$ of indices $i$ for which $\tau[i] = \text{unused}$, consider a copy $\tau'$ of $\tau$ 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 $\tau \in \text{introduce}(\tau, v)$ and $|\text{introduce}(\tau, v)| \leq 2^\ell$. All signatures obtained through the introduce operation are valid, because of the explicit check.

**Definition 12.** Given a signature $\tau$ over $X$ and a vertex $v \in X$, the forget operation, returns a new signature $\tau' = \text{forget}(\tau, v)$ over $X \setminus \{v\}$ such that for all $i \leq \ell$, we have $\tau'[i] = \text{used}$ if $\tau[i] = v$ and $\tau'[i] = \tau[i]$ otherwise.

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

**Definition 13.** Given two signatures $\tau_1$ and $\tau_2$ over $X \subseteq V(G)$, we say $\tau_1$ and $\tau_2$ are compatible if for all $i \leq \ell$:

1. $\tau_1[i] = \tau_2[i] = \text{unused}$,
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 $\text{used}$ and the other is equal to $\text{unused}$.

For two compatible signatures $\tau_1$ and $\tau_2$, the join operation returns a new signature $\tau' = \text{join}(\tau_1, \tau_2)$ over $X$ such that for all $i \leq \ell$ we have, respectively:

1. $\tau'[i] = \text{unused}$,
2. $\tau'[i] = v$, and
3. $\tau'[i] = \text{used}$.

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

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 $A_i = \text{introduce}(\tau, v)$, where $\tau$ is the signature with only $\text{unused}$ 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 $A_i = \bigcup_{\tau \in A_j} \text{introduce}(\tau, v)$.

**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 $A_i = \{\text{forget}(\tau, v) \mid \tau \in A_j\}$.

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

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

**Lemma 14 (*)&.** For $i \in V(T)$ and a signature $\tau$ over $X_i$, $\tau \in A_i$ if and only if $\tau$ can be extended to a signature over $V_i$ that is valid.

**Theorem 15 (*)&.** VC-R and IS-R can be solved in $O^*(4^\ell(t + 3)^\ell)$ time on graphs of treewidth $t$. 

9
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 [4, 11, 14], we show a similar result for VC-R and IS-R on planar graphs. The idea is that at most \( \ell \) 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 16 (\(*)\).** VC-R and IS-R are fixed-parameter tractable on planar graphs when parameterized by \( \ell \). Moreover, there exists an algorithm which solves both problems in \( O^*(4^\ell(3\ell + 2)^\ell) \) time.

We note that, by a simple application of the result of Demaine et al. [10], Theorem 16 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 \( \Pi \) 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 \( O^*(3^\ell) \) by additionally maintaining a bipartition of the bag (with the solution deleted) \([15, 16]\). That is, at every bag \( X_i \), we would maintain a list of assignments \( X \rightarrow \{\text{used}, \text{left}, \text{right}\} \) 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 \text{used}, \text{left}, and \text{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 17.** An OCT-signature \( \tau \) over a set \( X \subseteq V(G) \) is a sequence of steps \( \tau[1], \ldots, \tau[\ell] \in X \cup \{\text{used, unused}\} \) together with an entry \( \tau[i, v] \in \{\text{left, right}\} \) for every \( 1 \leq i \leq \ell - 1 \) and \( v \in X \setminus \tau(i, S_s \cap X) \).

There are at most \( (t + 3)^\ell 2^{t(\ell - 1)} \) 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 \( \{v \mid \tau[i, v] = \text{left}\} \) and \( \{v \mid \tau[i, v] = \text{right}\} \) 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 $\{\text{left}, \text{right}\}$ 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 14, with the new definitions, can then be proved again by induction.

**Theorem 18 (\(*\)).** OCT-R and IBS-R can be solved in $O^*(2^{t\ell^4}(t + 3)^t)$ time on graphs of treewidth $t$.

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

**Theorem 19.** FVS-R and IF-R can be solved in $O^*(t^{t\ell^4}(t + 3)^t)$ 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. A partial k-arboretum of graphs with bounded treewidth. *Theor. Comput. Sci.*, 209(1–2):1–45, Dec. 1998.
4. H. L. Bodlaender and A. M. C. A. Koster. Combinatorial optimization on graphs of bounded treewidth. *Comput. J.*, 51(3):255–269, May 2008.
5. P. Bonsma. The complexity of rerouting shortest paths. In *Proc. of Mathematical Foundations of Computer Science*, pages 222–233, 2012.
6. P. Bonsma. Rerouting shortest paths in planar graphs. In FSTTCS 2012, volume 18 of Leibniz International Proceedings in Informatics (LIPIcs), pages 337–349, 2012.
7. L. Cereceda, J. van den Heuvel, and M. Johnson. Connectedness of the graph of vertex-colourings. Discrete Mathematics, 308(56):913–919, 2008.
8. L. Cereceda, J. van den Heuvel, and M. Johnson. Finding paths between 3-colorings. Journal of Graph Theory, 67(1):69–82, 2011.
9. B. Courcelle. The monadic second-order logic of graphs. I. recognizable sets of finite graphs. Information and Computation, 85(1):12 – 75, 1990.
10. 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.
11. E. D. Demaine and M. Hajiaghayi. The bidimensionality theory and its algorithmic applications. Comput. J., 51(3):292–302, 2008.
12. R. Diestel. Graph Theory. Springer-Verlag, Electronic Edition, 2005.
13. R. G. Downey and M. R. Fellows. Parameterized complexity. Springer-Verlag, 1997.
14. D. Eppstein. Diameter and treewidth in minor-closed graph families. Algorithmica, 27(3):275–291, 2000.
15. S. Fiorini, N. Hardy, B. Reed, and A. Vetta. Planar graph bipartization in linear time. Discrete Appl. Math., 156(7):1175–1180, Apr. 2008.
16. J. Flum and M. Grohe. Parameterized complexity theory. Springer-Verlag, 2006.
17. 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.
18. M. Grohe. Logic, graphs, and algorithms. Electronic Colloquium on Computational Complexity (ECCC), 14(091), 2007.
19. 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.
20. T. Ito, M. Kamiński, and E. D. Demaine. Reconfiguration of list edge-colorings in a graph. Discrete Applied Mathematics, 160(15):2199–2207, 2012.
21. M. Kamiński, P. Medvedev, and M. Milanič. Shortest paths between shortest paths. Theor. Comput. Sci., 412(39):5205–5210, 2011.
22. T. Kloks. Treewidth, Computations and Approximations, volume 842 of Lecture Notes in Computer Science. Springer, 1994.
23. 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.
24. R. Niedermeier. Invitation to fixed-parameter algorithms. Oxford University Press, 2006.
25. E. L. Post. Recursive unsolvability of a problem of Thue. The Journal of Symbolic Logic, 12(1):1–11, 1947.
26. W. J. Savitch. Relationships between nondeterministic and deterministic tape complexities. Journal of computer and system sciences, 4(2):177–192, 1970.
27. M. Wrochna. Reconfiguration in bounded bandwidth and treedepth, 2014. arXiv:1405.0847.
Appendix

A  Details omitted from Section 3

Proof of Lemma 3

Proof. We note that Bauer and Otto’s explicit proof for $c$-balanced Thue systems \cite{2} can easily be adapted to give a 2-balanced Thue system. We include a self-contained proof here for completeness.

Since only words of the same length can be reached by application of rules in a balanced Thue system, it suffices to nondeterministically search all words of the same length to solve the problem in nondeterministic polynomial space. By Savitch’s Theorem \cite{26}, this places the problem in PSPACE.

Let $M = (\Sigma, Q, q_0, q_{acc}, q_{rej}, \delta)$ be a deterministic Turing Machine working in space bounded by a polynomial $p(|x|)$, where $p$ is a polynomial function and $x \in \Sigma^*$, which accepts any PSPACE-complete language. (By starting from a fixed PSPACE-complete problem we show the word problem to be hard for a certain fixed Thue system; starting from any language in PSPACE we would only show that the more general word problem, where the system is given as input, is PSPACE-complete). $\Sigma$ is the tape alphabet of $M$, $Q$ is the set of states, $q_0, q_{acc}, q_{rej}$ are the initial, accepting, and rejecting state respectively, and $\delta : Q \times \Sigma \to Q \times \Sigma \times \{L, R\}$ is the transition function of $M$. Let $\$, $\in \Sigma$ denote the left and right end-markers. We assume without loss of generality that the machine clears the tape and moves its head to the left end when reaching the accepting state.

For any input $x \in \Sigma^*$ we encode a configuration of the Turing Machine by a word of length exactly $p(|x|)$ over the alphabet $\Gamma = \Sigma \cup (\Sigma \times Q) \cup \{\\}$. If the tape content is $a_1a_2 \ldots a_n\$ for some $a_1, \ldots, a_n \in \Sigma$, the head’s position is $i \in \{0, 1, \ldots, n + 1\}$ and the machine’s state is $q$, then we define the corresponding word to be the tape content padded with $\$ symbols and with $a_i$ replaced by $(q, a_i)$, that is $a_1 \ldots a_{i-1}(q, a_i)a_{i+1} \ldots a_n\$ $\in \Gamma^{p(|x|)}$. The initial configuration is then encoded as $s_x = (q_0, \$)a_1 \ldots a_n\$ $\in \Gamma^{p(|x|)}$ and the only possible accepting configuration is encoded as $t_x = (q_{acc}, \$)a_1 \ldots a_n\$ $\in \Gamma^{p(|x|)}$. Since $M$ never uses more than $p(|x|)$ space on input $x$, our encoding is well defined for all configurations appearing in the execution of $M$ on $x$. So $M$ accepts input $x$ if and only if from $s_x$ one reaches the configuration $t_x$ by repeatedly applying the transition function. Such an application corresponds exactly to the following (ordered) string rewriting rules, in the encodings:

- $(q, a)c$ , $(p, b)c$ for $q \in Q$, $a, c \in \Sigma$ and $\delta(q, a) = (p, b, \cdot)$,
- $(q, a)c$ , $b(p, c)$ for $q \in Q$, $a, c \in \Sigma$ and $\delta(q, a) = (p, b, R)$,
- $(c(q, a), (p, c)b$ for $q \in Q$, $a, c \in \Sigma$ and $\delta(q, a) = (p, b, L)$.

The transition relation is not symmetric, but since the machine $M$ is deterministic, the configuration digraph (with machine configurations as vertices and the transition function as the adjacency relation) has out-degree one. The configuration $t_x$ (which is a configuration in the accepting state) has a loop, i.e.
a directed edge from $t_x$ to $t_x$. Therefore from any configuration, $t_x$ is reachable by a directed path if and only if it is reachable by any path. This means that $M$ accepts input $x$ if and only if applying the transition rules to $s_x$ leads to $t_x$ if and only if $s_x \leftrightarrow_R t_x$, where $R$ is the symmetric closure of the above rules, i.e., the 2-balanced Thue system over $\Gamma$ with rules:

- $\{(q, a)c, (p, b)c\}$ for $q \in Q$, $a, c \in \Sigma$ and $\delta(q, a) = (p, b, \cdot)$,
- $\{(q, a)c, b(p, c)\}$ for $q \in Q$, $a, c \in \Sigma$ and $\delta(q, a) = (p, b, R)$,
- $\{c(q, a), (p, c)b\}$ for $q \in Q$, $a, c \in \Sigma$ and $\delta(q, a) = (p, b, L)$.

Since the map $x \mapsto (s_x, t_x)$ is computable in logarithmic space, this proves the word problem of $(\Gamma, R)$ to be PSPACE-hard. □

Proof of Lemma 4

Proof. We first need to slightly strengthen Lemma 3 to give a Thue system where only one symbol at a time can be changed. To that aim, it suffices to replace a rule changing two symbols with a sequence of rules using two new intermediary symbols.

Claim 1 There is a 2-balanced Thue system $(\Gamma, R)$ whose word problem is PSPACE-complete and such that for every rule $\{a_1a_2, b_1b_2\} \subseteq R$ either $a_1 = b_1$ or $a_2 = b_2$.

Proof. Let $(\Sigma, R)$ be the 2-balanced Thue system from Lemma 3. Suppose $\{a_1a_2, b_1b_2\}$ is a rule of $R$ in which $a_1 \neq b_1$ and $a_2 \neq b_2$. We construct a 2-balanced Thue system $(\Gamma, S)$ with one fewer such rule, preserving PSPACE-completeness of the word problem. The claim then follows inductively.

Let $\Gamma = \Sigma \cup \{X, Y\}$, where $X$ and $Y$ are new symbols which will be used to replace a rule changing two symbols with a sequence of rules changing only one symbol. Let $S = R \setminus \{\{a_1a_2, b_1b_2\}\} \cup \{\{a_1a_2, Xa_2\}, \{Xa_2, XY\}, \{XY, b_1Y\}, \{b_1Y, b_1b_2\}\}$. We show that for any $s, t \in \Sigma^*$ it holds that $s \leftrightarrow_R t$ if and only if $s \leftrightarrow_S t$, which implies that our construction preserves PSPACE-completeness.

Clearly if $s \leftrightarrow_R t$ then $s \leftrightarrow_S t$, because replacing $a_1a_2$ with $b_1b_2$ can be done in $S$ by replacing $a_1a_2$ with $Xa_2$, then $XY$, then $b_1Y$ and finally $b_1b_2$. Suppose now $s \leftrightarrow_S t$ for some $s, t \in \Sigma^*$. Then there is a sequence $s = u_0, u_1, u_2, \ldots, u_t = t$ of words $u_i \in \Gamma^*$ such that $u_i \leftrightarrow_S u_{i+1}$. Let $\phi: \Gamma^* \rightarrow \Sigma^*$ be defined by replacing all $XY$ substrings of a word with $a_1a_2$, then replacing all remaining $X$ symbols with $a_1$ and all remaining $Y$ symbols with $b_2$. It is easy to check that $\phi(u_i) \leftrightarrow_R \phi(u_{i+1})$ or $\phi(u_i) = \phi(u_{i+1})$. Since $\phi(u_0) = \phi(s) = s$ and $\phi(u_t) = \phi(t) = t$, this implies that $s \leftrightarrow_R t$. □

Let $(\Gamma, S)$ be the 2-balanced Thue system from Lemma 1 (so if $\{a_1a_2, b_1b_2\} \subseteq S$ then $a_1 = b_1$ or $a_2 = b_2$). Let $S = \{S_1, \ldots, S_m\}$.

Let $\$, $\$, $\$, $x_1, \ldots, x_m$ be new symbols, let $\Delta_1 = \{\$, $\$, $x_1, \ldots, x_m\}$, $\Delta_2 = (\Gamma \cup \{\$\}) \times (\Gamma \cup \{\$\})$, and let $\Delta = \Delta_1 \cup \Delta_2$. We will call $\Delta_1$ special symbols and $\Delta_2$ pair symbols. Let $H = (\Delta, E)$, where we define $E \subseteq \Delta^2$ as the relation containing the following pairs:

14
\((a, b), (b, c)\) for any \(a, b, c \in \Gamma\),
\((\$, a)\) for any \(a \in \Gamma\),
\((a, \_), \_\) for any \(a, b, c \in \Gamma\),
\((a, 1), x_i)\)
\((b, 1), x_i\)
\((x_i, (a_2, \_))\)
\((x_i, (b_2, \_))\) for any \(\_ \in \Gamma\) and \(i \in \{1, \ldots, m\}\) such that \(S_i = \{a_1a_2, b_1b_2\}\).

Let \((s, t) \in \Gamma^* \times \Gamma^*\) be an instance of the word problem for \(S\), without loss of generality \(|s| = |t| = n\). Define \(\psi : \Gamma^n \to \Delta^{n+3}\) as

\[
\psi(a_1a_2 \ldots a_n) = \$(\_ , a_1)(a_1, a_2)(a_2, a_3) \ldots (a_{n-1}, a_n)(a_n, \_). 
\]

It is easy to see that if \(s \leftrightarrow \_ t\), then \(\psi(s)\) can be transformed into \(\psi(t)\), e.g., applying the rule \(S_i = \{a_1a_2, b_1a_2\}\) corresponds to replacing \((\_, a_1)(a_1, a_2)(a_2, \_))\) by \((\_, a_1)x_i(a_2, \_))\). Notice that the special symbol \$ must precede a pair symbol \((\_, \_))\) for some \(_ \in \Gamma\) and any such pair symbol must be preceded by \$. Since only one symbol at a time can be changed, it follow inductively that for each \(j \in \{0, \ldots, l\}\) the first two symbols of \(u_j\) must be \$(\_, \_))\) for some \(\_ \in \Gamma\) and \$ appears nowhere else. A similar argument applies to the last two symbols, \((\_, \_))\) for some \(\_ \in \Gamma\).

Since \(u_{j-1}\) and \(u_j\) differ at one position, there are non-empty words \(v, w \in \Delta^*\) and symbols \(a, b \in \Delta\), \(a \neq b\) such that \(u_{j-1} = vaw\) and \(u_j = vbw\). If \(a\) or \(b\) is a special symbol then both the last symbol of \(v\) and the first symbol of \(w\) are pair symbols, so \(\phi(u_{j-1}) = \phi(u_j)\). Otherwise, let \(a = (a_1, a_2), b = (b_1, b_2).\) Assume without loss of generality that \(a_1 \neq b_1\) and \(a_2 \neq b_2\) (the case \(a_1 = b_1, a_2 \neq b_2\) is analogous and the case \(a_1 \neq b_1, a_2 = b_2\) can be split by showing that \(\phi(u_{j-1}) \leftrightarrow \_ \phi(u')\) and \(\phi(u') \leftrightarrow \_ \phi(u_j)\) for \(u' = v(b_1, a_2)w\), which can easily be checked to be an \(H\)-word). If the last symbol of \(v\) is a pair symbol \((c, d), then \(d = a_1\) and \(d = b_1\), contradicting our assumption. If the last symbol of \(v\) is \$, then \(a_1 = b_1 = \_\). Finally if the last symbol of \(v\) is \(x_i\) for some \(i \in \{1, \ldots, m\}\), then \(S_i\) must equal \(\{ca_1, c'b_1\}\) for some \(c, c' \in \Gamma\). Since \(a_1 \neq b_1\), we have \(c = c'\) and the second-to-last symbol of \(v\) must be a pair \((\_, \_))\). Thus \(\phi(v(b_1, a_2)w)\) is obtained from \(\phi(v(a_1, a_2)w)\) by replacing the symbol \(a_1\) at position \(|v|\), which is preceded by a \(c\), by the symbol \(b_1\), that is, \(\phi(v(b_1, a_2)w) \leftrightarrow \_ \phi(v(a_1, a_2)w)\).
B  Details omitted from Section 5.2

Proof of Lemma 14

Proof. We first prove a few statements about signature validity. Note that all signatures in the algorithm are obtained through join, forget or introduce operations, which preserve validity and thus for each \( i \in V(T) \), the table \( A_i \) contains only valid signatures over \( X_i \).

**Lemma 20.** If a signature \( \tau \) over \( X \) is obtained from a valid signature by replacing all vertex steps not in \( X \) by used or unused steps, then \( \tau \) is valid as well.

Proof. Let \( \tau \) be obtained from a valid signature \( \tau' \) over \( X' \) by replacing all vertex steps in \( X' \setminus X \) by used or unused steps. First note that \( \tau(i,S_s \cap X) = \tau'(i,S_s \cap X') \cap X \). The first three conditions of Definition 9 follow immediately. As \( \Pi \) is hereditary, \( G[X' \setminus S] \in \Pi \) implies \( G[X \setminus (S \cap X)] \in \Pi \), hence the fourth condition also follows.

**Lemma 21.** Let \( G \) be a graph \( S, X_1, X_2 \) be subsets of \( V(G) \) such that every edge of \( G[X_1 \cup X_2] \) is contained in \( G[X_1] \) or \( G[X_2] \). If \( S \cap X_1 \) is a vertex cover of \( G[X_1] \), \( S \cap X_2 \) is a vertex cover of \( G[X_2] \) then \( S \) is a vertex cover of \( G[X_1 \cup X_2] \).

Proof. Let \( uv \) be an edge of \( G[X_1 \cup X_2] \). Then it is an edge of \( G[X_i] \) for some \( i \in \{1,2\} \). Hence it one of \( u, v \) must be a member of \( S \cap X_i \). Thus every edge of \( G[X_1 \cup X_2] \) has an endpoint in \( S \).

**Corollary 22.** Let \( \tau, \tau_1, \tau_2 \) be signatures over \( X, X_1, X_2 \) respectively, such that \( X = X_1 \cup X_2 \) and every edge of \( G[X] \) is contained in \( G[X_1] \) or \( G[X_2] \). Assume furthermore that for all \( i \leq \ell \):

\[
\begin{align*}
\tau[i] &= \tau_1[i] \text{ whenever } \tau[i] \in X_1 \text{ or } \tau_1[i] \in X_1 \\
\tau[i] &= \tau_2[i] \text{ whenever } \tau[i] \in X_2 \text{ or } \tau_2[i] \in X_2.
\end{align*}
\]

If \( \tau_1 \) and \( \tau_2 \) are valid, then so is \( \tau \).

Proof. The assumption means that \( \tau \) and \( \tau_1 \) agree over all changes within \( X_1 \), that is, \( \tau(i,S_s \cap X_1) = \tau_1(i,S_s \cap X_1) \cap X_1 \) (and similarly for \( \tau_2 \)). The first two conditions of Definition 9 for \( \tau \) follow immediately: if \( \tau[i] \in X \) then \( \tau[i] \in X_1 \) or \( \tau[i] \in X_2 \), so the statement is equivalent to the first two conditions for \( \tau_1 \) or for \( \tau_2 \). To show the third condition for \( \tau \), observe that \( \tau(i,S_s \cap X) = (\tau(i,S_s \cap X_1) \cap X_1) \cup (\tau(i,S_s \cap X_2) \cap X_2) = \tau_1(i,S_s \cap X_1) \cup \tau_2(i,S_s \cap X_2) = (S_1 \cap X_1) \cup (S_1 \cap X_2) = S_1 \cap X \). For the last condition, it suffices to use Lemma 21 for \( S = \tau(i,S_s \cap X) \).

We now prove Lemma 14: For \( i \in V(T) \) and a signature \( \tau \) over \( X_i \), \( \tau \in A_i \) if and only if \( \tau \) can be extended to a signature over \( V_i \) that is valid. We prove the statement by induction over the tree \( T \), that is, we prove the statement to be true at \( i \in V(T) \) assuming we have already proved it for all other nodes in the subtree of \( T \) rooted at \( i \). Depending on whether \( i \) is a leaf, forget, introduce or join node, we have the following cases.
Leaf nodes. Let \( v \) be the only vertex of \( X_i \), that is, \( V_i = X_i = \{ v \} \). Since \( V_i = X_i \), a signature \( \tau \) over \( X_i \) can be extended to a signature valid over \( V_i \) if and only if \( \tau \) is valid and has no \textit{used} steps. That is, if and only if \( \tau \) has only \textit{unused} and \( v \) steps and is valid (over \( X_i \)), which happens if and only if \( \tau \in A_i \).

Forget nodes. Let \( j \) be the child of \( i \), thus \( X_i = X_j \setminus \{ v \} \) for some \( v \in X_j \) and \( V_i = V_j \).

For one direction, suppose \( \tau \in A_i \) over \( X_i \). Then there is a \( \tau_j \) in \( A_j \) over \( X_j \) such that \( \tau = \text{forget}(\tau_j, v) \). By inductive assumption, \( \tau_j \) has an extension \( \pi \) valid over \( V_j = V_i \). Since \( \tau_j \) is be obtained from \( \tau \) by replacing some \textit{used} steps with \textit{v} steps, \( \pi \) is also an extension of \( \tau \). Thus \( \tau \) has an extension valid over \( V_i \).

For the other direction, suppose \( \tau \) has an extension \( \pi \) valid over \( V_i \). Then \( \pi \) is obtained from \( \tau \) by replacing some \textit{used} steps with vertex steps from \( V_i \setminus X_i \). Since \( V_i \setminus X_i = (V_j \setminus X_j) \cup \{ v \} \), we can consider the signature \( \tau_j \) over \( X_j \cup \{ v \} \) obtained by only using the replacements with \textit{v} steps. This signature \( \tau_j \) can be extended to \( \pi \) by using the remaining replacements, so by inductive assumption \( \tau_j \in A_j \). Furthermore, \( \text{forget}(\tau_j, v) = \tau \). Thus \( \tau \in A_i \).

| \( \tau \) | \textit{unused} | \textit{used} | \textit{used} | \( X_j \) |
| --- | --- | --- | --- | --- |
| \( \tau_j \) | \textit{unused} | \textit{used} | \textit{v} | \( X_j \) |
| \( \pi \) | \textit{unused} | \( V_i \setminus X_i \) | \textit{v} | \( X_j \) |

Introduce nodes. Let \( j \) be the child of \( i \), thus \( X_i = X_j \cup \{ v \} \) for some \( v \in X_i \) and \( V_i = V_j \cup \{ v \} \).

For one direction, suppose \( \tau \in A_i \) is a signature over \( X_i \). Then there is a \( \tau_j \in A_j \) such that \( \tau \) can be obtained from \( \tau_j \) by replacing some \textit{unused} steps with \textit{v} steps. By inductive assumption \( \tau_j \) has an extension \( \pi_j \) over \( V_j \) that is valid. As \( \pi_j \) can be obtained from \( \tau_j \) by replacing \textit{used} steps with vertex steps from \( V_j \setminus X_j \) and \( \tau \) has \textit{used} steps at the same positions, we can use the same replacements to obtain an extension \( \pi \) over \( V_j \cup \{ v \} \) of \( \tau \). \( \pi \) agrees with \( \pi_j \) over \( V_j \) and with \( \tau \) over \( X_i \), it is thus valid over \( V_i \) by Corollary 22. Therefore \( \tau \) has an extension over \( V_i \) that is valid.

For the other direction, suppose \( \tau \) has an extension \( \pi \) over \( V_i \) that is valid. Let \( \pi_j \) be the signature over \( V_j = V_i \setminus \{ v \} \) obtained by replacing all \textit{v} steps of \( \pi \) with \textit{unused} steps. By Lemma 20, \( \pi_j \) is valid. Let \( \tau_j \) be the signature over \( X_j = X_i \setminus \{ v \} \) obtained by replacing all \textit{v} steps of \( \tau \) with \textit{unused} steps. Then \( \pi_j \) is an extension of \( \tau_j \), thus \( \tau_j \in A_j \) by inductive assumption. Since \( \pi \) is valid, so is \( \tau \) (Lemma 20), thus \( \tau \in \text{introduce}(\tau_j, v) \) and \( \tau \in A_i \).

| \( \tau \) | \textit{unused} | \textit{used} | \textit{v} | \( X_j \) |
| --- | --- | --- | --- | --- |
| \( \tau_j \) | \textit{unused} | \textit{used} | \textit{unused} | \( X_j \) |
| \( \pi_j \) | \textit{unused} | \( V_j \setminus X_j \) | \textit{unused} | \( X_j \) |
| \( \pi \) | \textit{unused} | \( V_j \setminus X_j \) | \textit{v} | \( X_j \) |

Join nodes. Let \( j, h \) be the children of \( i \), thus \( V_i = V_j \cup V_h \) and we will write \( X \) for \( X_i = X_j = X_h \).

For one direction suppose \( \tau \in A_i \) valid over \( X \). Then there are two compatible signatures \( \tau_j \in A_j, \tau_h \in A_h \) such that \( \tau = \text{join}(\tau_j, \tau_h) \). By inductive assumption,
they have valid extensions, \( \pi_j \) over \( V_j \) and \( \pi_h \) over \( V_h \), respectively. Let \( I_i, I_j, I_h \) be the sets of indices of \textbf{used} steps in \( \tau, \tau_j, \tau_h \), respectively. By Definition 13, \( I_i \) is the sum of disjoint sets \( I_j, I_h \). Since \( \pi_j \) is obtained from \( \tau_j \) by replacing steps at indices \( I_j \) with vertex steps from \( V_j \setminus X \) and similarly for \( \pi_h \), we can define a signature \( \pi \) obtained from \( \tau \) over \( X \cup (V_j \setminus X) \cup (V_h \setminus X) = V_i \) by using both sets of replacements. \( \pi \) is an extension of \( \tau \). Moreover, \( \pi \) agrees with \( \pi_j \) over \( V_j \) and with \( \pi_h \) over \( V_h \), so by Corollary 22, \( \pi \) is valid over \( V_j \cup V_h = V_i \). Therefore \( \tau \) has an extension over \( V_i \) that is valid.

For the other direction, suppose \( \tau \) has an extension \( \pi \) over \( V_i \) that is valid. Let \( I_j \) be the set of indices of vertex steps from \( V_j \setminus X \) in \( \pi \) and define \( I_h \) accordingly. Let \( \tau_j, \pi_j \) be obtained from \( \tau, \pi \) by replacing all steps at indices \( I_h \) by \textbf{unused} steps. Since \( V_j \cap V_h = X \), \( \pi_j \) is an extension of \( \tau_j \) over \( V_j \). By Lemma 20 \( \pi_j \) is valid, thus by inductive assumption \( \tau_j \in A_j \). Define \( \tau_h, \pi_h \) accordingly and observe that \( \tau_h \in A_h \). It is easy to see that \( \tau \) has \textbf{used} steps exactly at the indices \( I_j \cup I_h \) and \( \tau_j, \tau_h \) have \textbf{used} steps exactly at the disjoint sets of indices \( I_j, I_h \), respectively. This implies \( \tau_j, \tau_h \) are compatible and \( \tau = \text{join}(\tau_j, \tau_h) \), so \( \tau \in A_i \).

**Proof of Theorem 15**

**Proof.** Recall that we say \( G \models \omega_\sigma(S_s, S_t) \) if there is a reconfiguration sequence (of vertex covers of \( G \)) of length exactly \( \ell \) from \( S_s \) to \( S_t \), such that the \( i^{th} \) step is a vertex removal if \( \sigma[i] = -1 \) and a vertex addition if \( \sigma[i] = +1 \). The following lemma states the correctness of the acceptance condition of our algorithm.

**Lemma 23.** \( G \models \omega_\sigma(S_s, S_t) \) if and only if \( A_{\text{root}} \) contains a signature \( \tau \) over \( X_{\text{root}} \) such that no step of \( \tau \) is \textbf{unused}.

**Proof.** From Lemma 14, we know that \( A_{\text{root}} \) contains a signature \( \tau \) over \( X_{\text{root}} \) such that no step of \( \tau \) is \textbf{unused} if and only if there is a signature \( \pi \) over \( V_{\text{root}} = V \) that is valid and such that no step of \( \pi \) is \textbf{unused}. This means that \( \pi \) contains only vertex steps and by definition of validity, the corresponding sequence \( \pi(0, S_s), \ldots, \pi(\ell, S_t) \) is a reconfiguration sequence of length exactly \( \ell \) from \( S_s \) to \( S_t \) such that the \( i^{th} \) step is a vertex removal if \( \sigma[i] = -1 \) and a vertex addition if \( \sigma[i] = +1 \).

It remains to prove the bound on the running time of our algorithm. The number of nodes in \( T \) is in \( \mathcal{O}(n) \). Checking the compatibility and validity of
signatures can be accomplished in time polynomial in $\ell, t, n$. For each node $i \in V(T)$ the table $A_i$ contains at most $(t + 3)^\ell$ signatures. Updating tables at the leaf nodes requires $O^*(2^\ell)$ time, since we check the validity of $2^\ell$ signatures obtained from one introduce operation. In the worst case, updating the table of an introduce node requires $O^*(2^\ell(t + 3)^\ell)$ time, i.e. applying the introduce operation on each signature in a table of size $(t + 3)^\ell$. For forget nodes, the time spent is polynomial in the maximum size of a table, that is $O^*((t + 3)^\ell)$. Finally, updating the table of a join node can be implemented in $O^*(2^\ell(t + 3)^\ell)$ time by checking for each of the $(t + 3)^\ell$ possible signatures all possible ways to split used steps among the two children. The algorithm needs to be run for every $\sigma \in \{-1, +1\}^\ell$ that doesn’t violate the maximum allowed capacity, giving in total the claimed $O^*(4^\ell(t + 3)^\ell)$ time bound.

Given an instance $(G, S_s, S_t, k, \ell)$ of IS-R, we can solve the corresponding VC-R instance $(G, V(G) \setminus S_s, V(G) \setminus S_t, n - k, \ell)$ in $O^*(4^\ell(t + 3)^\ell)$ time on graphs of treewidth $t$. Combining this fact with Proposition 2 yields the result for IS-R. \hfill $\square$

C Details omitted from Section 5.3

Proof of Theorem 16

Proof. Given a plane embedding of a planar graph $G$, the vertices of $G$ are divided into layers $\{L_1, \ldots, L_r\}$ as follows: Vertices incident to the exterior face are in layer $L_0$. For $i \geq 0$, we let $G'$ be the graph obtained by deleting all vertices in $L_0 \cup \ldots \cup L_i$ from $G$. All the vertices that are incident to the exterior face in $G'$ are in layer $L_{i+1}$ in $G$. $L_r$ is thus the last non-empty layer. A planar graph that has an embedding where the vertices are in $r$ layers is called $r$-outerplanar.

The following result is due to Bodlaender [3].

Lemma 24 (Bodlaender [3]). The treewidth of an $r$-outerplanar graph $G$ is at most $3r - 1$. Moreover, a tree decomposition of width at most $3r - 1$ can be constructed in time polynomial in $|V(G)|$.

From Lemma 24, we have the following corollary.

Corollary 25 ([3, 4]). For a planar graph $G$, we let $E$ be an arbitrary plane embedding of $G$ and $\{L_1, \ldots, L_r\}$ be the collection of layers corresponding to $E$. Then for any $i, r \geq 1$, the treewidth of the subgraph $G[L_{i+1} \cup \ldots \cup L_{i+\ell}]$ is at most $3\ell - 1$.

We now summarize the main ideas behind how we use the shifting technique. Note that every vertex in $S_s \Delta S_t$ must be touched at least once in any reconfiguration sequence $\alpha$ from $S_s$ to $S_t$. In other words, $S_s \Delta S_t \subseteq V(\alpha)$. Moreover, we know that $|V(\alpha)|$ is at most $\ell$, as otherwise the corresponding VC-R instance is a no-instance. For an arbitrary plane embedding of a planar graph $G$ and every fixed $j \in \{0, \ldots, \ell\}$, we let $G_j$ be the graph obtained by deleting all vertices in $L_{i(\ell+1)+j}$, for all $i \in \{0, 1, \ldots, \lfloor n/(\ell + 1)\rfloor\}$. Note that $tw(G_j) \leq 3\ell - 1$. 

19
**Proposition 26.** If there exists a reconfiguration sequence $\alpha$ of length exactly $\ell$ between two vertex covers $S_s$ and $S_t$ of a planar graph $G$, then for some fixed $j \in \{0, \ldots, \ell\}$ we have $V(\alpha) \subseteq V(G_j)$.

We still need a few gadgets before we can apply Theorem 15 on each graph $G_j$ and guarantee correctness. In particular, we need to handle deleted vertices and “border” vertices correctly, i.e. vertices incident to the exterior face in $G_j$.

We solve at most $\lfloor n/(\ell + 1) \rfloor + 1$ instances of the VC-R problem as follows:

1. Find an arbitrary plane embedding of $G$.
2. For every fixed $j \in \{0, \ldots, \ell\}$:
   3. Let $G^*_j = G_j$.
   4. Let $D^*_j$ denote the set of vertices deleted from $G$ to obtain $G^*_j$.
   5. If $\{S_s, \Delta S_s\} \cap D^*_j \neq \emptyset$:
      a. Ignore this instance (continue from line 2).
   6. Partition $D^*_j$ into $A^*_j = D^*_j \cap \{S_s \cap S_t\}$ and $B^*_j = D^*_j \setminus A^*_j$.
   7. Let $S^*_{s,j} = S_s \cap V(G^*_j)$ and $S^*_{t,j} = S_t \cap V(G^*_j)$.
   8. If \( v \in S^*_{s,j} \cup B^*_j \neq \emptyset \): 
      a. Ignore this instance (continue from line 2).
   9. For every vertex $v \in A^*_j$:
      a. Add an $(\ell + 1)$-star centered at $u$ to $G^*_j$.
      b. Add $u$ to $S^*_{s,j}$ and $S^*_{t,j}$.
   10. For every vertex in $\{v \in S^*_{s,j} \cap S^*_{t,j} \cup B^*_j \neq \emptyset\}$:
      a. Add $\ell + 1$ degree-one neighbors to $v$ in $G^*_j$.
   11. Solve instance $(G^*_j, S^*_{s,j}, S^*_{t,j}, k, \ell)$.

On lines 5 and 6, we make sure that no vertices from the symmetric difference of $S_s$ and $S_t$ lie in the deleted layers of $G$, as otherwise $G^*_j$ can be ignored, by Proposition 26. Hence, we know that $D^*_j$ can only include vertices common to both $S_s$ and $S_t$ (vertices in $S_s \cap S_t$) and we can partition $D^*_j$ into two sets accordingly (line 7). In the remaining steps, we add gadgets to account for the capacity used by vertices in $A^*_j$ and the fact that the neighbors of any vertex in $B^*_j$ must remain untouched. In other words, we assume that there exists a reconfiguration sequence $\alpha$ from $S^*_{s,j}$ to $S^*_{t,j}$ in $R_{\min}(G^*_j, k)$. Then $\alpha$ is a reconfiguration sequence from $S_s$ to $S_t$ in $R_{\min}(G, k)$ only if:

1. $|S^*_{s,j}| + \text{capacity}(\alpha) \leq k - |A^*_j|$, where $\text{capacity}(\alpha) = \max_{1 \leq i \leq \ell} (\sum_{i=1}^{\ell} \text{sign}(\alpha, i))$ and $\text{sign}(\alpha, i)$ is -1 when the $i^{th}$ step of $\alpha$ is a deletion, +1 when it is an addition; and
2. no vertex deletion in $\alpha$ leaves an edge uncovered in $G$.

To guarantee property (1), we add an $(\ell + 1)$-star to $G^*_j$ for every vertex in $A^*_j$, then add the center of the star into both $S^*_{s,j}$ and $S^*_{t,j}$ (lines 11, 12, and 13). Therefore, for every value of $j$ we have $|S_j| = |S^*_{s,j}|$, $|T| = |S^*_{t,j}|$, and $|S^*_{s,j}| + \text{capacity}(\alpha) \leq k - |A^*_j|$. For property (2), we add $\ell + 1$ degree-one neighbors to every vertex in $\{v \in S^*_{s,j} \cap S^*_{t,j} \cup B^*_j \neq \emptyset\}$ (lines 14 and 15). Those vertices, as well as the centers of the stars, will have to remain untouched in $\alpha$, as otherwise deleting any such vertex would require more than $\ell$ additions.
Since adding degree-one vertices and \((\ell+1\))-stars to a graph does not increase its treewidth, we have \(tw(G_j^*) \leq 3\ell - 1\) for all \(j\) (Corollary 25). Hence, for each graph \(G_j^*\) we can now apply Theorem 15 and solve the VC-R instance \((G_j^*, S_{s,j}^*, S_{t,j}^*, k, \ell)\) in \(O^*(4^\ell(3\ell + 1)^\ell)\) time. We prove in Lemma 27 that our original instance on planar \(G\) is a yes-instance if and only if \((G_j^*, S_{s,j}^*, S_{t,j}^*, k, \ell)\) is a yes-instance for some fixed \(j \in \{0, 1, \ldots, \lfloor n/(\ell+1) \rfloor\}\).

**Lemma 27.** \((G, S_s, S_t, k, \ell)\) is a yes-instance of VC-R if and only if \((G_j^*, S_{s,j}^*, S_{t,j}^*, k, \ell)\) is a yes-instance for some fixed \(j \in \{0, 1, \ldots, \lfloor n/(\ell+1) \rfloor\}\).

**Proof.** For \((G, S_s, S_t, k, \ell)\) a yes-instance of VC-R, there exists a reconfiguration sequence \(\alpha\) of length exactly \(\ell\) from \(S_s\) to \(S_t\). Then by Corollary 26, we know that for some fixed \(j \in \{0, 1, \ldots, \lfloor n/(\ell+1) \rfloor\}\) we have \(V(\alpha) \subseteq V(G_j^*)\) and \(V(\alpha) \cap N_G(B_j^*) = \emptyset\), as otherwise \(V(\alpha) \cap B_j^* \neq \emptyset\). By our construction of \(G_j^*\), the maximum capacity constraint is never violated. Therefore, \(\alpha\) is also a reconfiguration sequence from \(S_{s,j}^*\) to \(S_{t,j}^*\).

For the converse, suppose that \((G_j^*, S_{s,j}^*, S_{t,j}^*, k, \ell)\) is a yes-instance for some fixed \(j \in \{0, 1, \ldots, \lfloor n/(\ell+1) \rfloor\}\) and let \(\alpha\) denote the corresponding reconfiguration sequence from \(S_{s,j}^*\) to \(S_{t,j}^*\). Since the maximum capacity constraint cannot be violated, we only need to make sure that (i) no reconfiguration step in \(\alpha\) leaves an uncovered edge in \(G\) and that (ii) none of the degree-one gadget vertices are in \(V(\alpha)\). For (i), it is not hard to see that any such vertex must be touched an even number of times and we can delete those reconfiguration steps to obtain a shorter reconfiguration sequence. Moreover, any reconfiguration sequence of length \(\ell-x\), where \(x\) is even, can be transformed into a reconfiguration sequence of length \(\ell\) by a simple application of the last reconfiguration step and its reversal \(\frac{\ell}{2}\) times. For (ii), assume that \(\alpha\) leaves an uncovered edge in \(G\). By our construction of \(G_j^*\), such an edge must have one endpoint in \(B_j^*\). But since we added \(\ell+1\) degree-one neighbors to every vertex in the neighborhood of \(B_j^*\), this is not possible.

Theorem 16 then follows by combining Proposition 2, Lemma 24, Lemma 27, Theorem 15, and the fact that \(tw(G_j^*) \leq 3\ell - 1\), for all \(j \in \{0, 1, \ldots, \lfloor n/(\ell+1) \rfloor\}\).

**D Details omitted from Section 5.4**

**Proof of Theorem 18**

**Proof.** The proof of correctness proceeds very similarly as for VC-R, we only need to argue that the strengthened last condition for validity (which uses the additional information about bipartitions in an essential way) is now strong enough to carry through the main inductive proof.

**Lemma 28.** If an OCT-signature \(\tau\) over \(X\) is obtained from a valid OCT-signature by replacing all vertex steps not in \(X\) by used or unused steps, then \(\tau\) is valid as well.
Proof. Let \( \tau \) be obtained from a valid OCT-signature \( \tau' \) over \( X' \) by replacing all vertex steps in \( X' \setminus X \) by \textit{used} or \textit{unused} steps. First note that \( \tau(i, S_s \cap X) = \tau'(i, S_s \cap X') \cap X \). The first three conditions of Definition 9 follow immediately. Moreover, if \( G[X \setminus S] \) has a bipartition \( L, R \), then \( L \cap X, R \cap X \) is a bipartition of \( G[X \setminus (S \cap X)] \), hence the fourth condition also follows. \( \square \)

Lemma 29. Let \( G \) be a graph \( S, X_1, X_2 \) be subsets of \( V(G) \) such that every edge of \( G[X_1 \cup X_2] \) is contained in \( G[X_1] \) or \( G[X_2] \). Let \( L, R \) be a partition of \( X_1 \cup X_2 \). If \( L \cap X_1, R \cap X_1 \) is a bipartition of \( G[X_1 \setminus S] \) and \( L \cap X_2, R \cap X_2 \) is a bipartition of \( G[X_2 \setminus S] \), then \( L, R \) is a bipartition of \( G[(X_1 \cup X_2) \setminus S] \).

Proof. Let \( uv \) be an edge of \( G[(X_1 \cup X_2) \setminus S] \). Then it is contained in \( G[X_i] \) for some \( i \in \{1, 2\} \). It has no endpoint in \( S \cap X_i \), hence it is an edge of \( G[X_i \setminus S] \). Thus one endpoint is in \( L \cap X_i \) and the other in \( R \cap X_i \). In particular every edge of \( G[(X_1 \cup X_2) \setminus S] \) has one endpoint in \( L \) and the other in \( R \). \( \square \)

Corollary 30. Let \( \tau, \tau_1, \tau_2 \) be a OCT-signatures over \( X, X_1, X_2 \) respectively, such that \( X = X_1 \cup X_2 \) and every edge of \( G[X] \) is contained in \( G[X_1] \) or \( G[X_2] \).

Assume furthermore that for all \( i \leq t \):

\[
\tau[i] = \tau_1[i] \quad \text{whenever } \tau[i] \in X_1 \text{ or } \tau_1[i] \in X_1, \\
\tau[i] = \tau_2[i] \quad \text{whenever } \tau[i] \in X_2 \text{ or } \tau_2[i] \in X_2, \\
\tau[i, v] = \tau_1[i, v] \quad \text{whenever } v \in X_1 \text{ and } \tau[i, v] \text{ is defined and} \\
\tau[i, v] = \tau_2[i, v] \quad \text{whenever } v \in X_2 \text{ and } \tau[i, v] \text{ is defined.}
\]

If \( \tau_1 \) and \( \tau_2 \) are valid, then so is \( \tau \).

Proof. The assumption implies that \( \tau \) and \( \tau_1 \) agree over all changes within \( X_1 \), that is, \( \tau(i, S_s \cap X_1) = \tau(i, S_s \cap X) \cap X_1 \) (and similarly for \( \tau_2 \)). The first three conditions of validity for \( \tau \) follow as for VC-R. For the last condition, it suffices to use Lemma 29 for \( S = \tau(i, S_s \cap X) \cap X_1 \), \( L = \{v \mid \tau[i, v] = \text{left}\} \), \( R = \{v \mid \tau[i, v] = \text{right}\} \). \( \square \)

The following lemma is proved by induction exactly as for VC-R, only with Lemma 28 and Corollary 30 used when validity needs to be argued.

Lemma 31. For \( i \in V(T) \) and an OCT-signature \( \tau \) over \( X_i \), \( \tau \in A_i \) if and only if \( \tau \) can be extended to an OCT-signature over \( V_i \) that is valid.

The accepting condition is unchanged and its correctness follows from Lemma 31 the same way. It only remains to consider the running time. The number of possible OCT-signatures is \((t + 3)^{2^{t(t-1)}}\) (instead of the \((t + 3)^t\) for VC-R). In the join operation, we required the new \( \tau[i, v] \) entries to be equal and thus the running time is again \( 2^t \) times the number of possible OCT-signatures. In the forget operation the algorithm only does a polynomial number of calculations for each of the OCT-signatures. In the introduce operation, for each of the OCT-signatures we consider in the worst case \( 2^t \) possible subsets of \textit{unused} steps and \( 2^t \) possible assignments of \textit{left} or \textit{right} to new \( \tau[i, v] \) entries. The total running time is thus \( O^*(4^t(t + 3)^22^t) \).

Combining the same complementing technique we used for VC-R and IS-R with Proposition 2, the result for IBS-R follows.  \( \square \)