Vertex Cover Reconfiguration and Beyond

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

[1] David R. Cheriton School of Computer Science
University of Waterloo, Ontario, Canada.
{aabdomou, nishi}@uwaterloo.ca
[2] The Institute of Mathematical Sciences
Chennai, India.
vraman@imsc.res.in

Abstract. In the Vertex Cover Reconfiguration (VCR) problem, given graph $G = (V, E)$, positive integers k and ℓ, and two vertex covers S and T of G of size at most k, we determine whether S can be transformed into T by a sequence of at most ℓ vertex additions or removals such that each operation results in a vertex cover of size at most k. Motivated by recent results establishing the **W**[**1**]-hardness of VCR when parameterized by ℓ, we delineate the complexity of the problem restricted to various graph classes. In particular, we show that VCR remains **W**[**1**]-hard on bipartite graphs, is **NP**-hard but fixed-parameter tractable on graphs of bounded degree, and is solvable in time polynomial in $|V(G)|$ on even-hole-free graphs and cactus graphs. We prove **W**[**1**]-hardness and fixed-parameter tractability via two new problems of independent interest.

1 Introduction

Under the reconfiguration framework, we consider structural and algorithmic questions related to the solution space of a search problem $\mathcal{Q}$. Given an instance $\mathcal{I}$, an optional range $[r_l, r_u]$ bounding a numerically quantifiable property Ψ of feasible solutions for $\mathcal{Q}$, and a symmetric adjacency relation (usually polynomially-testable) $\mathcal{A}$ on the set of feasible solutions, we can construct a *reconfiguration graph* $R_{\mathcal{Q}}(\mathcal{I}, r_l, r_u)$ for each instance $\mathcal{I}$ of $\mathcal{Q}$. The nodes of $R_{\mathcal{Q}}(\mathcal{I}, r_l, r_u)$ correspond to the feasible solutions of $\mathcal{Q}$ having $r_l \leq \Psi \leq r_u$, with an edge between each pair of nodes corresponding to solutions adjacent under $\mathcal{A}$ (viewed as a *reconfiguration step* transforming one solution into the other). One can ask if there exists a walk (*reconfiguration sequence*) in $R_{\mathcal{Q}}(\mathcal{I}, r_l, r_u)$ between feasible solutions S and T of $\mathcal{I}$, or for the shortest such walk.

These types of reconfiguration questions have received considerable attention in recent literature [9, 11] and are interesting for a variety of reasons. From an

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

algorithmic standpoint, reconfiguration problems model dynamic situations in which we seek to transform a solution into a more desirable one, maintaining feasibility during the process. Moreover, the study of reconfiguration yields insights into the structure of the solution space of the underlying problem, crucial for the design of efficient algorithms [3, 9]. Reconfiguration problems have so far been studied mainly under classical complexity assumptions, with most work devoted to determining the existence of a reconfiguration sequence between two given solutions. For most **NP**-complete problems, this question has been shown to be **PSPACE**-complete [11], while for some problems in **P**, the reconfiguration question could be either in **P** [11] or **PSPACE**-complete [2]. As **PSPACE**-completeness implies that the length of reconfiguration sequences can be exponential in the size of the input graph, it is natural to ask whether tractability is possible when the running time depends on the length of the sequence or on other properties of the problem. These results motivated Mouawad et al. [13] to study reconfiguration under the *parameterized complexity* framework [5].

Overview of our results For the Vertex Cover Reconfiguration (VCR) problem, the node set of $R_{\mathrm{VC}}(G, 0, k)$ consists of all vertex covers of size at most k of an n-vertex graph G and two nodes are adjacent in $R_{\mathrm{VC}}(G, 0, k)$ if the vertex cover corresponding to one can be obtained from the other by the addition or removal of a single vertex. VCR is known to be **PSPACE**-hard even when restricted to planar graphs of maximum degree three [10]. Recently [13], the problem was shown to be $\mathbf{W[1]}$-hard when parameterized by ℓ. Hence, for the general case, one cannot hope for an algorithm solving the problem in $\mathcal{O}(f(\ell)(nk)^{\mathcal{O}(1)})$ time, for some computable function f. However, as noted by Mouawad et al. [13], there is a close relation between the fixed-parameter tractability of the problem parameterized by ℓ and the size of the symmetric difference of the two vertex covers in question. In particular, when the size of the symmetric difference is greater than ℓ, we have a trivial no-instance. When the size is equal to ℓ, the problem is solvable by a simple $\mathcal{O}(\ell!)$ time enumeration algorithm. In fact, it is easy to see that even when the size of the symmetric difference is $\ell - c$, for any constant c, we can solve the problem in $\mathcal{O}(\ell^\ell \binom{n}{c})$ time. In some sense, these observations imply that the problem becomes "harder" as the number of "choices" we have to make "outside" of the symmetric difference increases.

In this work, we embark on a systematic investigation of the (parameterized) complexity of the problem to better understand the relationship between the (fixed-parameter) tractability of the problem and the size and structure of the symmetric difference. In Section 3, we show, via a new problem of independent interest, that VCR parameterized by ℓ remains $\mathbf{W[1]}$-hard when restricted to bipartite graphs. The main motivation behind considering bipartite graphs is due to another observation relating the structure of the input graph to the size of the symmetric difference. That is, when the size of the symmetric difference is equal to n, the input graph is bipartite (Observation 2) and ℓ must be greater than or equal to n. Thus, even if $\ell = n$, the enumeration algorithm mentioned above would run in time exponential in n. Can we do any better if $\ell \ll n$ and the input graph is bipartite? Our hardness result answers this question in the negative unless $\mathbf{FPT} = \mathbf{W[1]}$. The result also answers a question left open by Mouawad et al. [13] by providing the first example of a search problem in $\mathbf{P}$ whose reconfiguration version is $\mathbf{W[1]}$-hard parameterized by ℓ.

Interestingly, excluding odd cycles does not seem to make the VCR problem any easier, whereas excluding (induced or non-induced) even cycles puts the problem in $\mathbf{P}$. We prove this result in Section 4. Although at first glance this positive result does not seem to be related to the symmetric difference, we show that we can in fact obtain polynomial-time algorithms whenever the symmetric difference has some "nice" properties. We show that for even-hole-free graphs the symmetric difference is indeed a forest. The structure is slightly more complex for cactus graphs. Moreover, in both cases, the number of vertices we need to consider outside of the symmetric difference is bounded by a constant. We note that a similar polynomial-time algorithm for even-hole-free graphs was also recently, and independently, obtained by Kamiński et al. for solving several variants of the Independent Set Reconfiguration problem [12].

In Section 5, we start by introducing the notion of nice reconfiguration sequences and show that any reconfiguration sequence can be converted into a nice one. A nice reconfiguration sequence can be split into smaller "pieces" where the added/removed vertices in the first and last piece induce independent sets in the input graph G and the added/removed vertices in all other pieces induce a biclique in G. For graphs of degree at most d, each biclique has at most d vertices on each side. Given this rather intriguing structure, we believe that nice reconfiguration sequences deserve a more careful study. Using the notion of nice reconfiguration sequences, we show in two steps that VCR is $\mathbf{NP}$-hard on 4-regular graphs. In the first step, we prove a general hardness result showing that VCR is at least as hard as the compression variant of the Vertex Cover (VC) problem. Then, we construct a 4-regular gadget W_k on $6k^2$ vertices with the following property: There exist two minimum vertex-disjoint vertex covers S and T of W_k, each of size $3k^2$, such that there exists a path of length $6k^2$ between the nodes corresponding to S and T in $R_{\text{VC}}(W_k, 0, 3k^2 + g(k))$ but no such path exists in $R_{\text{VC}}(W_k, 0, 3k^2 + g(k) - 1)$, for some computable function g and $k - 2 \leq g(k) \leq k + 3$. The existence of graphs with properties similar to W_k has played an important role in determining the complexity of other reconfiguration problems [3, 14]. For instance, the existence of a 3-regular version of W_k, combined with the fact that reconfiguration is at least as hard as compression, would immediately imply that VCR is $\mathbf{NP}$-hard on 3-regular graphs.

Finally, we show that even though $\mathbf{NP}$-hard on 4-regular graphs, VCR can be solved in $\mathcal{O}(f(\ell, d)(nk)^{\mathcal{O}(1)})$ time on graphs of degree at most d, for some computable function f. This result answers another open question [13], presenting the first fixed-parameter algorithm for VCR parameterized by ℓ, in this case for graphs of bounded degree. The algorithm is rather technical, as it involves reductions to three intermediary problems, uses a structural decomposition of the input graph, and exploits the properties of nice reconfiguration sequences.

However, at a very high level, this result relates to the symmetric difference as follows: In any yes-instance of the VCR problem on graphs of degree at most d, one can easily bound the size of the symmetric difference and the set of vertices “not too far away” from it, i.e. the distance is some function of ℓ and d. However, to guarantee that the capacity constraint k, i.e. the maximum size of a vertex cover, is never violated, it may be necessary to add/remove vertices which are ”far” from (maybe not even connected to) the symmetric difference. Hence, the main technical challenge is to show that finding such vertices can be accomplished “efficiently”.

We believe that the techniques used in both our hardness proofs and positive results can be extended to cover a host of graph deletion problems defined in terms of hereditary graph properties [13]. It also remains to be seen whether our **FPT** result can be extended to a larger class of sparse graphs similar to the work of Fellows et al. on local search [7].

2 Preliminaries

For general graph theoretic definitions, we refer the reader to the book of Diestel [4]. Unless otherwise stated, we assume that each graph G is a simple, undirected graph with vertex set $V(G)$ and edge set $E(G)$, where $|V(G)| = n$ and $|E(G)| = m$. The *open neighborhood* of a vertex v is denoted by $N_G(v) = \{u \mid (u, v) \in E(G)\}$ and the *closed neighborhood* by $N_G[v] = N_G(v) \cup \{v\}$. For a set of vertices $S \subseteq V(G)$, we define $N_G(S) = \{v \notin S \mid (u, v) \in E(G), u \in S\}$ and $N_G[S] = N_G(S) \cup S$. The subgraph of G induced by S is denoted by $G[S]$, where $G[S]$ has vertex set S and edge set $\{(u, v) \in E(G) \mid u, v \in S\}$.

Vertices s and t (vertex sets A and B) are *separated* if there is no edge (s, t) (no edge (a, b) for $a \in A$, $b \in B$). The *distance* between two vertices s and t of G, $dist_G(s, t)$, is the length of a shortest path in G from s to t. For $r \geq 0$, the *r-neighborhood* of a vertex $v \in V(G)$ is defined as $N^r_G[v] = \{u \mid dist_G(u, v) \leq r\}$. We write $B(v, r) = N^r_G[v]$ and call it a *ball of radius r around v*; for $A \subseteq V(G)$, $B(A, r) = \bigcup_{v \in A} N^r_G[v]$. Section 5 makes use of the following:

Observation 1 *For any graph G of degree at most d, $v \in V(G)$, and $A \subseteq V(G)$, $|B(v, r)| \leq d^{r+1}$ and $|B(A, r)| \leq |A|d^{r+1}$.*

To avoid confusion, we refer to *nodes* in reconfiguration graphs, as distinguished from *vertices* in the input graph. We denote an instance of the VCR problem by (G, S, T, k, ℓ), where G is the input graph, S and T are the *source* and *target* vertex covers respectively, k is the *maximum allowed capacity*, and ℓ is an upper bound on the length of the reconfiguration sequence we seek in $R_{\text{VC}}(G, 0, k)$. By a slight abuse of notation, we use upper case letters to refer to both a node in the reconfiguration graph as well as the corresponding vertex cover. For any node $S \in V(R_{\text{VC}}(G, 0, k))$, the quantity $k - |S|$ corresponds to the *available capacity* at S. We partition $V(G)$ into the sets $C_{ST} = S \cap T$ (vertices common to S and T), $S_R = S \setminus C_{ST}$ (vertices to be removed from S in the course of reconfiguration), $T_A = T \setminus C_{ST}$ (vertices to be added to form T), and

$O_{ST} = V(G) \setminus (S \cup T) = V(G) \setminus (C_{ST} \cup S_R \cup T_A)$ (all other vertices). A vertex is *touched* in the course of a reconfiguration sequence from S to T if v is either added or removed at least once.

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

Observation 2 (*) *For a graph G and two vertex covers S and T of G, $G[S_R \cup T_A]$ is bipartite, and there is no edge (u, v) for $u \in S_R \cup T_A$ and $v \in O_{ST}$.*

Observation 3 *For a graph G and vertex covers S and T of G, in any reconfiguration sequence of length at most ℓ from S to T a vertex can be touched at most $\ell - |S_R \cup T_A| + 1$ times, each vertex in $S_R \cup T_A$ being touched an odd number of times and each other vertex being touched an even number of times.*

Throughout this work, we implicitly consider VCR as a parameterized problem with ℓ as the parameter. The reader is referred to the book of Downey and Fellows [5] for more on parameterized complexity. We sometimes use the modified big-Oh notation $\mathcal{O}^*$ that suppresses all polynomially bounded factors.

3 Bipartite graphs

For a graph $G = (V, E)$, a *crown* is a pair (W, H) satisfying the following properties: (i) $W \neq \emptyset$ is an independent set of G, (ii) $N_G(W) = H$, and (iii) there exists a matching in $G[W \cup H]$ which saturates H [1]. Crown structures have played a central role in the development of kernelization algorithms for the VC problem [1]. We define a *(k, d)-constrained crown* as a crown (W, H) such that $|H| \leq k$ and $|W| - |H| \geq d \geq 0$. Given a bipartite graph $G = (A \cup B, E)$ and two positive integers k and d, the (k, d)-Bipartite Constrained Crown ((k, d)-BCC) problem asks whether G has a (k, d)-constrained crown (W, H) such that $W \subseteq A$ and $H \subseteq B$.

Lemma 1. *(k, d)-BCC parameterized by $k + d$ is $\mathbf{W[1]}$-hard even when the graph, $G = (A \cup B, E)$, is C_4-free and vertices in A have degree at most two.*

Proof. We give an **FPT** reduction from k-Clique, known to be $\mathbf{W[1]}$-hard, to $(k, \binom{k}{2})$-Bipartite Constrained Crown. For (G, k) an instance of k-Clique, we let $V(G) = \{v_1, \ldots, v_n\}$ and $E(G) = \{e_1, \ldots, e_m\}$.

We first form a bipartite graph $G' = ((X \cup Z) \cup Y, E_1 \cup E_2)$, where vertex sets X and Y contain one vertex for each vertex in $V(G)$ and Z contains one vertex for each edge in $E(G)$. More formally, we set $X = \{x_1, \ldots, x_n\}$, $Y = \{y_1, \ldots, y_n\}$, and $Z = \{z_1, \ldots, z_m\}$. The edges in E_1 join each pair of vertices x_i and y_i for $1 \leq i \leq n$ and the edges in E_2 join each vertex z in Z to the two vertices y_i and y_j corresponding to the endpoints of the edge in $E(G)$ to which z corresponds. Since each edge either joins vertices in X and Y or vertices in Y and Z, it is not difficult to see that the vertex sets $X \cup Z$ and Y form a bipartition.

By our construction, G' is C_4-free; vertices in X have degree 1, and since there are no double edges in G, i.e. two edges between the same pair of vertices, no pair of vertices in Y can have more than one common neighbour in Z.

For $(G', k, \binom{k}{2})$ an instance of $(k, \binom{k}{2})$-BCC, $A = X \cup Z$, and $B = Y$, we claim that G has a clique of size k if and only if G' has a $(k, \binom{k}{2})$-constrained crown (W, H) such that $W \subseteq A$ and $H \subseteq B$.

If G has a clique K of size k, we set $H = \{y_i \mid v_i \in V(K)\}$, namely the vertices in Y corresponding to the vertices in the clique. To form W, we choose $\{x_i \mid v_i \in V(K)\} \cup \{z_i \mid e_i \in E(K)\}$, that is, the vertices in X corresponding to the vertices in the clique and the vertices in Z corresponding to the edges in the clique. Clearly H is a subset of size k of B and W is a subset of size $k + \binom{k}{2}$ of A; this implies that $|W| - |H| \geq d = \binom{k}{2}$, as required. To see why $N_{G'}(W) = H$, it suffices to note that every vertex $x_i \in W$ is connected to exactly one vertex $y_i \in H$ and every degree-two vertex $z_i \in W$ corresponds to an edge in K whose endpoints $\{v_i, v_j\}$ must have corresponding vertices in H. Moreover, due to E_1 there is a matching between the vertices of H and the vertices of W in X, and hence a matching in $G'[W \cup H]$ which saturates H.

Assuming that G' has a $(k, \binom{k}{2})$-constrained crown (W, H) such that $W \subseteq X \cup Z$ and $H \subseteq Y$, it suffices to show that $|H|$ must be equal to k, $|W \cap Z|$ must be equal to $\binom{k}{2}$, and hence $|W \cap X|$ must be equal to k; from this we can conclude the vertices in $\{v_i \mid y_i \in H\}$ form a clique of size k in G as $|W \cap Z| = \binom{k}{2}$, requiring that edges exist between each pair of vertices in the set $\{v_i \mid y_i \in H\}$. Moreover, since $|W \cap X| = k$ and $N_{G'}(W) = H$, a matching that saturates H can be easily found by picking all edges (x_i, y_i) for $y_i \in H$.

To prove the sizes of H and W, we first observe that since $|H| \leq k$, $N_{G'}(W) = H$, and each vertex in Y has exactly one neighbour in X, we know that $|W \cap X| \leq |H| \leq k$. Moreover, since $|W| = |W \cap X| + |W \cap Z|$ and $|W| - |H| \geq \binom{k}{2}$, we know that $|W \cap Z| = |W| - |W \cap X| \geq \binom{k}{2} + |H| - |W \cap X| \geq \binom{k}{2}$. If $|W \cap Z| = \binom{k}{2}$ our proof is complete since, by our construction of G', H is a set of at most k vertices in the original graph G and the subgraph induced by those vertices in G has $\binom{k}{2}$ edges. Hence, $|H|$ must be equal to k. If instead $|W \cap Z| > \binom{k}{2}$, since each vertex of Z has degree two, the number of neighbours of $W \cap Z$ in Y is greater than k, violating the assumptions that $N_{G'}(W) = H$ and $|H| \leq k$. □

Combining Lemma 1 with an **FPT** reduction from (k, d)-BCC to VCR yields the main theorem of this section:

Theorem 1 (*). *VCR parameterized by ℓ is* $\mathbf{W[1]}$*-hard on bipartite graphs.*

4 Even-hole-free and cactus graphs

A *cactus graph* is a connected graph in which each edge is in at most one cycle. A graph G is *even-hole-free* if no induced subgraph of G is a cycle on an even number of vertices. Examples of even-hole-free graphs include trees, interval graphs, and chordal graphs.

We present a characterization of instances of the VCR problem solvable in time polynomial in n, and then apply this characterization to trees, even-hole-free graphs, and cactus graphs. In all cases, we find reconfiguration sequences of shortest possible length and therefore ignore the parameter ℓ. Reconfiguration sequences are represented as ordered sequences of nodes in $R_{\mathrm{VC}}(G, 0, k)$.

Definition 1. *Given two vertex covers A and B of G, a reconfiguration sequence β from A to some vertex cover A' is a* c-bounded prefix *of a reconfiguration sequence α from A to B, denoted $A \xleftrightarrow{c,\,B} A'$, if and only if all of the following conditions hold: (1) $|A'| \leq |A|$, (2) for each node A'' in β, $|A''| \leq |A| + c$, (3) for each node A'' in β, A'' is obtained from its predecessor by either the removal or the addition of a single vertex in the symmetric difference of the predecessor and B, and (4) no vertex is touched more than once in the course of β. Moreover, $A \xleftrightarrow{c,\,B} A'$ implies $A \xleftrightarrow{d,\,B} A'$ for all $d > c$.*

Lemma 2 (*). *Given two vertex covers S and T of G and two positive integers k and c such that $|S|, |T| \leq k$, a reconfiguration sequence α of length $|S_R| + |T_A| = |S \Delta T|$ from S to T exists if: (1) $|S| \leq k - c$, (2) $|T| \leq k - c$, and (3) for any two vertex covers A and B of G such that $|A| \leq k - c$ and $|B| \leq k - c$, either $A \xleftrightarrow{c,\,B} A'$ or $B \xleftrightarrow{c,\,A} B'$, where A' and B' are vertex covers of G. Moreover, if c-bounded prefixes can be found in time polynomial in n, then so can α.*

The proof of Theorem 2 shows that Lemma 2 applies if G is a tree and S and T are of size at most $k - 1$, as we can always find 1-bounded prefixes $S \xleftrightarrow{1,\,T} S'$ or $T \xleftrightarrow{1,\,S} T'$ in time polynomial in n. Further refinements are required when at least one of S and T is of size greater than $k - 1$.

Theorem 2 (*). VCR *on trees can be solved in time polynomial in n.*

The proof of Theorem 2 uses G being a tree only to establish the fact that $G[S_R \cup T_A]$ is a forest. This fact holds for any even-hole-free graph, since a graph that is bipartite (Observation 2) and has no induced even cycles is a forest, hence:

Corollary 1. VCR *on even-hole-free graphs can be solved in time polynomial in n.*

To extend Corollary 1 to all cactus graphs (which are not necessarily even-hole-free), we show in Lemmas 3 and 4 that the third condition of Lemma 2 is satisfied for cactus graphs with $c = 2$ and that 2-bounded prefixes can be found in time polynomial in n.

Lemma 3 (*). *Given two vertex covers S and T of G, there exists a vertex cover S' (or T') of G such that $S \xleftrightarrow{2,\,T} S'$ (or $T \xleftrightarrow{2,\,S} T'$) if one of the following conditions holds: (1) $G[S_R \cup T_A]$ has a vertex $v \in S_R$ ($v \in T_A$) such that $|N_{G[S_R \cup T_A]}(v)| \leq 1$, or (2) there exists a cycle Y in $G[S_R \cup T_A]$ such that all vertices in $Y \cap S_R$ ($Y \cap T_A$) have degree exactly two in $G[S_R \cup T_A]$. Moreover, both conditions can be checked in time polynomial in n and when one of them is true the corresponding 2-bounded prefix can be found in time polynomial in n.*

Lemma 4 (*). *If G is a cactus graph and S and T are two vertex covers of G, then there exists a vertex cover S' (or T') of G such that $S \xleftrightarrow{2,T} S'$ (or $T \xleftrightarrow{2,S} T'$). Moreover, finding such 2-bounded prefixes can be accomplished in time polynomial in n.*

Theorem 3 (*). VCR *on cactus graphs can be solved in time polynomial in n.*

5 Graphs of bounded degree

Nice edit sequences We represent a partially specified reconfiguration sequence in terms of *markers* $\mathcal{E} = \{\varnothing, a, r\} \cup \mathcal{E}_a \cup \mathcal{E}_r$, where $\mathcal{E}_a = \{a_1, \ldots, a_n\}$ and $\mathcal{E}_r = \{r_1, \ldots, r_n\}$. An *edit sequence* α is an ordered sequence of elements of $\mathcal{E}$, where the markers a_i, a, r_j, r, and $\varnothing$ represent the addition of vertex v_i, an unspecified addition, the removal of vertex v_j, an unspecified removal, and a blank placeholder, respectively. We refer to $\mathcal{E}_a \cup \{a\}$ and $\mathcal{E}_r \cup \{r\}$ as the sets of *addition markers* and *removal markers*, respectively. An edit sequence α is *unlabeled* if it contains no markers in $\mathcal{E}_a \cup \mathcal{E}_r$, *partly labeled* if it contains at least one element from each of the sets $\{a, r\}$ and $\mathcal{E}_a \cup \mathcal{E}_r$, and *labeled* if it contains no markers in $\{a, r\}$. We say α is *partial* if it contains at least one $\varnothing$ and is *full* otherwise.

Observation 4 *The total number of possible full (partial) unlabeled edit sequences of length at most ℓ is $\sum_{i=1}^{\ell} 2^i < 2^{\ell+1}$ ($\sum_{i=1}^{\ell} 3^i < 3^{\ell+1}$).*

The *length* of α, $|\alpha|$, is the number of markers in α. We use $\alpha[p] \in \mathcal{E}$, $1 \leq p \leq |\alpha|$ to denote the marker at position p in α, and refer to it as a *blank marker* if $\alpha[p] = \varnothing$. By extension, $\alpha[p_1, p_2]$, $1 \leq p_1 \leq p_2 \leq |\alpha|$, denotes the edit sequence of length $p_2 - p_1 + 1$ formed from $\alpha[p_1]$ through $\alpha[p_2]$; such a sequence is a *segment* of α. Two segments β and β' are *consecutive* if $\beta = \alpha[p_1, p_2]$ and $\beta' = \alpha[p_2 + 1, p_3]$ for some $p_1 \leq p_2 \leq p_3$; β' (β) is the *successor* (*predecessor*) of β (β'). A segment β of α is an *add-remove segment* if β contains addition markers followed by removal markers, and a *d-add-remove segment*, $d > 0$, if it is an add-remove segment with i addition and j removal markers, $1 \leq i \leq d$ and $1 \leq j \leq d$. A *piece* is a group of zero or more consecutive segments.

Definition 2. *Given a positive integer $d > 0$, an edit sequence α is d-*well-formed *if it is subdivided into three consecutive pieces such that: (1) The* starting piece *consists of zero or more removal markers, (2) the* central piece *consists of zero or more d-add-remove segments, and (3) the* ending piece *consists of zero or more addition markers.*

We can form the length $|\beta| + |\gamma|$ *concatenation* $concat(\beta, \gamma)$ of two edit sequences β and γ in the obvious way, and *cut* the marker at position p by forming $concat(\beta[1, p-1], \beta[p+1, |\alpha|])$. The edit sequence $clean(\beta)$ is formed by cutting all blank markers. Given a partial edit sequence β and a full edit sequence γ, the *merging* operation consists of replacing the pth blank marker in β with the

pth marker in γ. We say a full edit sequence γ is a *filling edit sequence* of partial edit sequence β if $merge(\beta, \gamma)$ produces a full edit sequence. Given $t \geq 2$ full labeled edit sequences α_1, …, α_t, the *mixing* of those sequences, $mix(\alpha_1, \ldots, \alpha_t)$, produces the set of all full labeled edit sequences of length $|\alpha_1| + \ldots + |\alpha_t|$. Each $\alpha \in mix(\alpha_1, \ldots, \alpha_t)$ consists of all markers in each α_i, $1 \leq i \leq t$, such that the respective orderings of markers from each α_i is maintained, i.e. if we cut from α the markers of all sequences except α_1, we obtain α_1.

To relate edit and reconfiguration sequences, for graph G and edit sequence α, we use $V(\alpha)$ to denote the set of vertices touched in α, i.e. $V(\alpha) = \{v_i \mid a_i \in \alpha \vee r_i \in \alpha\}$. For α full and labeled, $V(S, \alpha)$ denotes the set of vertices obtained after executing all reconfiguration steps in α on G starting from some vertex cover S of G. If each set $V(S, \alpha[1,p])$, $1 \leq p \leq |\alpha|$, is a vertex cover of G, then α is *valid* (and *invalid* otherwise). Even if $|S| \leq k$, α is not necessarily a walk in $R_{\text{VC}}(G, 0, k)$, as α might violate the maximum allowed capacity constraint k. We say α is *tight* if it is valid and $\max_{1 \leq p \leq |\alpha|}(|V(S, \alpha[1,p])|) \leq k$. A partial labeled edit sequence α is valid or tight if $clean(\alpha)$ is valid or tight.

Observation 5 *Given a graph G and two vertex covers S and T of G, an edit sequence α is a reconfiguration sequence from S to T if and only if α is a tight edit sequence from S to T.*

Given a graph G, a vertex cover S of G, a full unlabeled edit sequence α, and an ordered sequence $L = \{l_1, \ldots, l_{|\alpha|}\}$ of (not necessarily distinct) labels between 1 and n, the *label* operation, denoted by $label(\alpha, L)$, returns a full labeled edit sequence α'; each marker in α' is copied from α and assigned the corresponding label from L. A full unlabeled edit sequence α can be *applied* to G and S if there exists an L such that $label(\alpha, L)$ is valid starting from S.

Definition 3. *For $t \geq 2$, graph G, and vertex cover S of G, valid labeled edit sequences α_1, …, α_t are* compatible *if each $\alpha \in mix(\alpha_1, \ldots, \alpha_t)$ is a valid edit sequence starting from S, and* incompatible *otherwise.*

Definition 4. *For a graph G of degree at most d and a vertex cover S of G, a valid edit sequence α starting from S is a* nice edit sequence *if it is valid, d-well-formed, and satisfies the following invariants:*

- **Connectivity invariant:** *$G[V(\beta_i)]$ is connected for all i, where β_i denotes the i^{th} d-add-remove segment in the central piece of α.*
- **Early removal invariant:** *For $1 \leq p_1 < p_2 < p_3 \leq |\alpha|$, if $\alpha[p_1] \in \mathcal{E}_a$, $\alpha[p_2] \in \mathcal{E}_a$, and $\alpha[p_3] \in \mathcal{E}_r$, then $V(\alpha[p_3])$ and $V(\alpha[p_1+1, p_3-1])$ are not separated.*

Intuitively, the early removal invariant states that every removal marker in a nice edit sequence must occur "as early as possible". In other words, a vertex is removed right after its neighbors (and possibly itself) are added.

Lemma 5 (*). *Given a graph G of degree at most d and two vertex covers S and T of G, it is possible to transform any valid edit sequence α from S to T into a nice edit sequence α' in $\mathcal{O}(n^4|\alpha|^4 2^d)$ time such that $|V(S, \alpha'[1,p])| \leq |V(S, \alpha[1,p])|$ for all $1 \leq p \leq |\alpha|$. In other words, if α is tight then so is α'.*

NP-hardness on 4-regular graphs We prove our result by demonstrating a reduction from VERTEX COVER COMPRESSION (VCC) to VCR where the input graph is restricted to be 4-regular; given a graph G and a vertex cover of size k, VCC asks whether G has a vertex cover of size $k-1$.

Theorem 4 (*). VCR *is at least as hard as* VCC.

Theorem 4 relies on a reduction involving the disjoint union of an instance of VCC and a biclique $K_{k,k}$; the instance of VCR can be reconfigured only if compression is possible. Using this idea, we show that VCR remains **NP**-hard for 4-regular graphs by constructing a 4-regular gadget W_k which will replace the $K_{k,k}$ biclique. Theorem 5 then follows from the facts that VC is **NP**-hard on 4-regular graphs [8] and any algorithm which solves the VCC problem can be used to solve VC.

Theorem 5 (*). VCR *is* **NP**-*hard on* 4-*regular graphs.*

FPT algorithm for graphs of bounded degree We make use of three different problems. In the ANNOTATED VCR (AVCR) problem, the vertex set of the input graph is partitioned into sets X, W, and R such that X and R are separated, $S_R \cup T_A \subseteq X$, and we seek a reconfiguration sequence in which no vertex in W is touched. In the VERTEX COVER WALK (VCW) problem, given a graph G, a vertex cover S of G, and a full unlabeled edit sequence σ of length $\ell \geq 1$, the goal is to determine whether we can apply σ to G and S. In the parameterized setting, VCW is at least as hard as the problem of determining, given a graph G, a vertex cover S of G, and integer $\ell \geq 1$, whether G has a vertex cover S' such that $|S'| < |S|$ and $|S' \Delta S| \leq \ell$ (VERTEX COVER LOCAL SEARCH (VCLS)) [7], known to be **W[1]**-hard on graphs of bounded degeneracy and **FPT** on graphs of bounded degree [7].

Lemma 6 (*). *When parameterized by ℓ,* VCW *is at least as hard as* VCLS.

Finally, we also make use of ℓ-MULTICOLORED INDEPENDENT SET (ℓ-MIS), the problem of determining, for a graph G, a positive integer ℓ, and a (not necessarily proper) vertex-coloring $c : V(G) \to \{c_1, \ldots, c_\ell\}$, whether G has an independent set of size ℓ including exactly one vertex of each color. Using a reduction from the **W[1]**-hard ℓ-MULTICOLORED CLIQUE problem [6] in which we complement all edges in the input graph, ℓ-MIS is **W[1]**-hard in general graphs. For $c(v)$ the color assigned to $v \in V(G)$, we say v belongs to *color class* $c(v)$, and let $V_i(G)$ denote the set of vertices assigned colored c_i in G, i.e. $V_i(G) = \{v \in V(G) \mid c(v) = c_i\}$.

Lemma 7 (*). *The* ℓ-MIS *problem parameterized by ℓ can be solved in (***FPT***) $\mathcal{O}^*((d\ell)^{2\ell})$ time if for every vertex $v \in V(G)$ such that $c(v) = c_i$, $|N_G(v) \cap V_j(G)| \leq d$, for some fixed constant d, $i \neq j$, and $1 \leq i, j \leq \ell$.*

Our **FPT** algorithm for VCR relies on a combination of enumeration and reductions to the aforementioned problems, starting with a reduction to AVCR:

Lemma 8 (*). *For any instance of* VCR*, there exists a set of 2ℓ instances $\{\mathcal{I}_1, \ldots, \mathcal{I}_{2\ell}\}$ of* AVCR *such that the original instance is a yes-instance for* VCR *if and only if at least one $\mathcal{I}_x$ is a yes-instance for* AVCR*, $1 \leq x \leq 2\ell$ and for each X_x, $S_R \cup T_A \subseteq X_x$.*

The algorithm implicit in the proof of Lemma 8 generates 2ℓ instances of AVCR such that the original instance is a yes-instance of VCR if and only if a generated instance is a yes-instance of AVCR. In each instance, there is a subset W of C_{ST} separating a superset X of $S_R \cup T_A$ from a vertex set R such that no vertex in W is touched during reconfiguration. We use enumeration to generate all partial labeled edit sequences that touch only vertices in X; if any produces a tight sequence that transforms S to T, we have a yes-instance. Otherwise, we consider sequences which are valid and transform S to T but exceed the capacity constraint. By finding an appropriate labeled filling sequence γ' that touches vertices in R, we can free up capacity so that $merge(\beta, \gamma')$ is tight.

We can find such a γ' trivially if there is a sufficiently large independent set in the vertices of $S \cap R$ with no neighbours in O_{ST}, as our reconfiguration sequence will consist of removing the vertices in the independent set to free up capacity, applying β, and then adding back the vertices in the independent set. Otherwise, we reduce the problem of finding γ' to an instance of VCW on $G[R]$ for each suitable unlabeled edit sequence γ of length the number of blanks in β.

Lemma 9 (*). *If* VCW *is solvable in $\mathcal{O}^*(f(d,\ell))$ time, for some computable function f, on a graph G of degree at most d, then* VCR *is solvable in $\mathcal{O}^*(2\ell 3^{\ell+1}(\ell d^{2\ell+1})^{2\ell+2}(d+\ell)^{\ell} 2^{\ell} f(d,\ell))$ time on G.*

To try all possible ways of generating γ', we start by enumerating all d-well-formed full unlabeled edit sequences γ of the appropriate length, and for each try all possible choices for the starting piece; by Lemma 5 this is sufficient. Given γ and a starting piece, we create t instances of VCW, where instance $\mathcal{J}_y$ corresponds to the graph induced by the labeled central piece having y connected components. To solve instance $\mathcal{J}_y$, we consider all ways of assigning the d-add-remove segments to connected components. This allows us to create a sequence for each component, where γ_h touches only vertices in component h, and all vertices touched by γ'_h can be found in a ball of radius $|\gamma'_h|$.

We can reduce each such subproblem to an instance of y-MIS satisfying Lemma 7, where a color c_h corresponds to component h. We denote the corresponding auxiliary graph by G_A. In G_A, we create vertices for each labeled full edit sequence λ, where $G[V(\lambda)]$ is connected and λ can be derived by adding labels to γ_h. There is an edge between two vertices in G_A if they have different colors and the sets of vertices associated with their edit sequences are not separated. Thus, a solution to y-MIS indicates that there are y full labeled edit sequences with separated vertex sets, as required to complete the central piece of γ. As the ending piece of γ' will be determined by the starting and central pieces, this completes the algorithm.

Lemma 10 (*). *Each instance of the* VCW *problem generated by our algorithm for* AVCR *can be solved in $\mathcal{O}^*(f(d,\ell))$ time, for some computable function f, on graphs of degree at most d.*

Combining Lemmas 9 and 10 yields the main theorem of this section:

Theorem 6. *For every fixed constant d,* VCR *parameterized by ℓ is fixed-parameter tractable for graphs of degree at most d.*

References

1. F. N. Abu-Khzam, R. L. Collins, M. R. Fellows, M. A. Langston, W. H. Suters, and C. T. Symons. Kernelization algorithms for the vertex cover problem: Theory and experiments. In *Proc. of the Sixth Workshop on Algorithm Engineering and Experiments*, pages 62–69, 2004.
2. P. Bonsma. The complexity of rerouting shortest paths. In *Proc. of Mathematical Foundations of Computer Science*, pages 222–233, 2012.
3. L. Cereceda, J. van den Heuvel, and M. Johnson. Connectedness of the graph of vertex-colourings. *Discrete Mathematics*, 308(56):913–919, 2008.
4. R. Diestel. *Graph Theory.* Springer-Verlag, Electronic Edition, 2005.
5. R. G. Downey and M. R. Fellows. *Parameterized complexity.* Springer-Verlag, New York, 1997.
6. M. R. Fellows, D. Hermelin, F. Rosamond, and S. Vialette. On the parameterized complexity of multiple-interval graph problems. *Theor. Comput. Sci.*, 410(1):53–61, 2009.
7. M. R. Fellows, F. A. Rosamond, F. V. Fomin, D. Lokshtanov, S. Saurabh, and Y. Villanger. Local search: is brute-force avoidable? In *Proc. of the 21st International Joint Conference on Artifical Intelligence*, pages 486–491, 2009.
8. M. R. Garey, D. S. Johnson, and L. J. Stockmeyer. Some simplified NP-complete graph problems. *Theor. Comput. Sci.*, 1(3):237–267, 1976.
9. 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.
10. R. A. Hearn and E. D. Demaine. PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theor. Comput. Sci.*, 343(1-2):72–96, 2005.
11. 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.
12. M. Kamiński, P. Medvedev, and M. Milanič. Complexity of independent set reconfigurability problems. *Theor. Comput. Sci.*, 439:9–15, June 2012.
13. 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.
14. A. Suzuki, A. E. Mouawad, and N. Nishimura. Reconfiguration of dominating sets. In *Computing and Combinatorics*, volume 8591 of *Lecture Notes in Computer Science*, pages 405–416. Springer International Publishing, 2014.