On the Parameterized Complexity of Reconfiguration Problems

Amer E. Mouawad[1⋆], Naomi Nishimura[1⋆], Venkatesh Raman[2], Narges Simjour[1⋆], and Akira Suzuki[3⋆⋆]

[1] David R. Cheriton School of Computer Science
University of Waterloo, Waterloo, Ontario, Canada.
{aabdomou, nishi, nsimjour}@uwaterloo.ca
[2] The Institute of Mathematical Sciences
Chennai, India. vraman@imsc.res.in
[3] Graduate School of Information Sciences, Tohoku University
Aoba-yama 6-6-05, Aoba-ku, Sendai, 980-8579, Japan.
a.suzuki@ecei.tohoku.ac.jp

Abstract. We present the first results on the parameterized complexity of reconfiguration problems, where a reconfiguration version of an optimization problem Q takes as input two feasible solutions S and T and determines if there is a sequence of *reconfiguration steps* that can be applied to transform S into T such that each step results in a feasible solution to Q. For most of the results in this paper, S and T are subsets of vertices of a given graph and a reconfiguration step adds or deletes a vertex. Our study is motivated by recent results establishing that for most NP-hard problems, the classical complexity of reconfiguration is PSPACE-complete.

We address the question for several important graph properties under two natural parameterizations: k, the size of the solutions, and ℓ, the length of the sequence of steps. Our first general result is an algorithmic paradigm, the *reconfiguration kernel*, used to obtain fixed-parameter algorithms for the reconfiguration versions of Vertex Cover and, more generally, Bounded Hitting Set and Feedback Vertex Set, all parameterized by k. In contrast, we show that reconfiguring Unbounded Hitting Set is $W[2]$-hard when parameterized by $k+\ell$. We also demonstrate the $W[1]$-hardness of the reconfiguration versions of a large class of maximization problems parameterized by $k+\ell$, and of their corresponding deletion problems parameterized by ℓ; in doing so, we show that there exist problems in FPT when parameterized by k, but whose reconfiguration versions are $W[1]$-hard when parameterized by $k+\ell$.

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

⋆⋆ Research supported by JSPS Grant-in-Aid for Scientific Research, Grant Number 24.3660.

1 Introduction

The reconfiguration version of an optimization problem asks whether it is possible to transform a source feasible solution S into a target feasible solution T by a (possibly minimum-length) sequence of *reconfiguration steps* such that every intermediate solution is also feasible. Reconfiguration problems model dynamic situations in which we seek to transform a solution into a more desirable one, maintaining feasibility during the process. The study of reconfiguration yields insights into the structure of the solution space of the underlying optimization problem, crucial for the design of efficient algorithms.

Motivated by these facts, there has been a lot of recent interest in studying the complexity of reconfiguration problems. Problems for which reconfiguration has been studied include Vertex Colouring [1–5], List Edge-Colouring [6], Independent Set [7, 8], Set Cover, Matching, Matroid Bases [8], Satisfiability [9], Shortest Path [10, 11], and Dominating Set [12, 13]. Most work has been limited to the problem of determining the existence of a reconfiguration sequence between two given solutions; for most NP-complete problems, this problem has been shown to be PSPACE-complete.

As there are typically exponentially many feasible solutions, the length of a reconfiguration sequence can be exponential in the size of the input instance. It is thus natural to ask whether reconfiguration problems become tractable if we allow the running time to depend on the length of the sequence. In this work, we explore reconfiguration in the framework of parameterized complexity [14] under two natural parameterizations: k, a bound on the size of feasible solutions, and ℓ, the length of the reconfiguration sequence. One of our key results is that for most problems, the reconfiguration versions remain intractable in the parameterized framework when we parameterize by ℓ. It is important to note that when k is not bounded, the reconfiguration problems we study become easy.

We present fixed-parameter algorithms for problems parameterized by k by modifying known parameterized algorithms for the problems. The paradigms of bounded search tree and kernelization typically work by exploring minimal solutions. However, a reconfiguration sequence may necessarily include non-minimal solutions. Any kernel that removes solutions (non-minimal or otherwise) may render finding a reconfiguration sequence impossible, as the missing solutions might appear in every reconfiguration sequence. To handle these difficulties, we introduce a general approach for parameterized reconfiguration problems. We use a *reconfiguration kernel*, showing how to adapt Bodlaender's cubic kernel [15] for Feedback Vertex Set, and a special kernel by Damaschke and Molokov [16] for Bounded Hitting Set (where the cardinality of each input set is bounded) to obtain polynomial reconfiguration kernels, with respect to k. These results can be considered as interesting applications of kernelization, and a general approach for other similar reconfiguration problems.

As a counterpart to our result for Bounded Hitting Set, we show that reconfiguring Unbounded Hitting Set or Dominating Set is $W[2]$-hard parameterized by $k+\ell$ (Section 4). Finally, we show a general result on reconfiguration problems of hereditary properties and their 'parametric duals', implying the $W[1]$-hardness of reconfiguring Independent Set, Induced Forest, and Bipartite Subgraph parameterized by $k+\ell$ and Vertex Cover, Feedback Vertex Set, and Odd Cycle Transversal parameterized by ℓ.

2 Preliminaries

Unless otherwise stated, we assume that each input graph G is a simple, undirected graph on n vertices with vertex set $V(G)$ and edge set $E(G)$. To avoid confusion, we refer to *nodes* in reconfiguration graphs (defined below), as distinguished from *vertices* in the input graph. We use the modified big-Oh notation O^* that suppresses all polynomially bounded factors.

Our definitions are based on optimization problems, each consisting of a polynomial-time recognizable set of valid instances, a set of feasible solutions for each instance, and an objective function assigning a nonnegative rational value to each feasible solution.

Definition 1. *The* reconfiguration graph $R_Q(I, \mathrm{adj}, k)$*, consists of a node for each feasible solution to instance I of optimization problem Q, where the size of each solution is at least k for Q a maximization problem (of size at most k for Q a minimization problem, respectively), for positive integer k, and an edge between each pair of nodes corresponding to solutions in the binary adjacency relation* adj *on feasible solutions.*

We define the following *reconfiguration problems*, where S and T are feasible solutions for I: Q Reconfiguration determines if there is a path from S to T in $R_Q(I, \mathrm{adj}, k)$; the *search variant* returns a *reconfiguration sequence*, the sequence of feasible solutions associated with such a path; and the *shortest path variant* returns the reconfiguration sequence associated with a path of minimum length.

Using the framework developed by Downey and Fellows [14], a *parameterized reconfiguration problem* includes in the input a positive integer ℓ (an upper bound on the length of the reconfiguration sequence) and a parameter p (typically k or ℓ). For a parameterized problem Q with inputs of the form (x, p), $|x| = n$ and p a positive integer, Q is *fixed-parameter tractable* (or in *FPT*) if it can be decided in $f(p)n^c$ time, where f is an arbitrary function and c is a constant independent of both n and p. Q has a *kernel* of size $f(p)$ if there is an algorithm A that transforms the input (x, p) to (x', p') such that A runs in polynomial time (with respect to $|x|$ and p) and (x, p) is a yes-instance if and only if (x', p') is a yes-instance, $p' \leq g(p)$, and $|x'| \leq f(p)$. Each problem in *FPT* has a kernel, possibly of exponential (or worse) size. The main hierarchy of parameterized complexity classes is $FPT \subseteq W[1] \subseteq W[2] \subseteq \ldots \subseteq XP$, where W-hardness, shown using *FPT reductions*, is the analogue of NP-hardness in classical complexity. The reader is referred to [17, 18] for more on parameterized complexity.

We introduce the notion of a *reconfiguration kernel*; it follows from the definition that a reconfiguration problem that has such a kernel is in *FPT*.

Definition 2. *A* reconfiguration kernel *of an instance* $(x, p) = (Q, \text{adj}, S, T, k, \ell, p)$ *of a parameterized reconfiguration problem is a set of* $h(p)$ *instances, for an arbitrary function* h*, such that for* $1 \leq i \leq h(p)$*:*

- *for each instance in the set,* $(x_i, p_i) = (Q, \text{adj}, S_i, T_i, k_i, \ell_i, p_i)$*, the values of* S_i*,* T_i*,* k_i*,* ℓ_i*, and* p_i *can all be computed in polynomial time,*
- *the size of each* x_i *is bounded by* $j(p)$*, for an arbitrary function* j*, and*
- (x, p) *is a yes-instance if and only if at least one* (x_i, p_i) *is a yes-instance.*

Most problems we consider can be defined using graph properties, where a *graph property* π is a collection of graphs, and is *non-trivial* if it is non-empty and does not contain all graphs. A graph property is *polynomially decidable* if for any graph G, it can be decided in polynomial time whether G is in π. For a subset $V' \subseteq V$, $G[V']$ is the *subgraph of* G *induced on* V', with vertex set V' and edge set $\{\{u, v\} \in E \mid u, v \in V'\}$. The property π is *hereditary* if for any $G \in \pi$, any induced subgraph of G is also in π. It is well-known [19] that every hereditary property π has a forbidden set $\mathcal{F}_\pi$, in that a graph has property π if and only if it does not contain any graph in $\mathcal{F}_\pi$ as an induced subgraph.

For a graph property π, we define two reconfiguration graphs, where solutions are sets of vertices and two solutions are adjacent if they differ by the addition or deletion of a vertex. The *subset reconfiguration graph of* G *with respect to* π, $R^\pi_{\text{SUB}}(G, k)$, has a node for each $S \subseteq V(G)$ such that $|S| \geq k$ and $G[S]$ has property π, and the *deletion reconfiguration graph of* G *with respect to* π, $R^\pi_{\text{DEL}}(G, k)$, has a node for each $S \subseteq V(G)$ such that $|S| \leq k$ and $G[V(G) \setminus S]$ has property π. We can obtain $R^\pi_{\text{DEL}}(G, |V(G)| - k)$ by replacing the set corresponding to each node in $R^\pi_{\text{SUB}}(G, k)$ by its (setwise) complement.

Definition 3. *For any graph property* π*, graph* G*, positive integer* k*,* $S \subseteq V(G)$*, and* $T \subseteq V(G)$*, we define the following decision problems:* π-DELETION(G, k)*: Is there* $V' \subseteq V(G)$ *such that* $|V'| \leq k$ *and* $G[V(G) \setminus V'] \in \pi$*?*
π-SUBSET(G, k)*: Is there* $V' \subseteq V(G)$ *such that* $|V'| \geq k$ *and* $G[V'] \in \pi$*?*
π-DEL-RECONF(G, S, T, k, ℓ)*: For* $S, T \in V(R^\pi_{\text{DEL}}(G, k))$*, is there a path of length at most* ℓ *between* S *and* T *in* $R^\pi_{\text{DEL}}(G, k)$*?*
π-SUB-RECONF(G, S, T, k, ℓ)*: For* $S, T \in V(R^\pi_{\text{SUB}}(G, k))$*, is there a path of length at most* ℓ *between* S *and* T *in* $R^\pi_{\text{SUB}}(G, k)$*?*

We say that π-DELETION(G, k) and π-SUBSET(G, k) are *parametric duals* of each other. We refer to π-DEL-RECONF(G, S, T, k, ℓ) and π-SUB-RECONF(G, S, T, k, ℓ) as π*-reconfiguration problems.*

Due to the page limitation, some proofs (marked with an asterisk) have been omitted and can be found in the full version of the paper [20].

3 Fixed-Parameter Tractability Results

For an instance (G, S, T, k, ℓ) of a π-reconfiguration problem, we partition $V(G)$ into the sets $C = S \cap T$, $S_D = S \setminus C$, $T_A = T \setminus C$, and $O = V(G) \setminus (S \cup T) =$

$V(G) \setminus (C \cup S_D \cup T_A)$ (all other vertices). Furthermore, we can partition C into two sets C_F and $C_M = C \setminus C_F$, where a vertex is in C_F if and only if it is in every feasible solution of size bounded by k. In any reconfiguration sequence, each vertex in S_D must be deleted and each vertex in T_A must be added. We say that a vertex v is *touched* if v is either added or deleted in at least one reconfiguration step. The following fact is a consequence of the definitions above, the fact that π is hereditary, and the observations that $G[S_D]$ and $G[O]$ are both subgraphs of $G[V(G) \setminus T]$, and $G[T_A]$ and $G[O]$ are both subgraphs of $G[V(G) \setminus S]$.

Fact 1 *For an instance π-DEL-RECONF(G, S, T, k, ℓ) of a reconfiguration problem for hereditary property π, $G[O]$, $G[S_D]$, and $G[T_A]$ all have property π.*

In the next section, we show that for most hereditary properties, reconfiguration problems are hard when parameterized by ℓ. Here, we prove the parameterized tractability of reconfiguration for certain superset-closed k-subset problems when parameterized by k, where a *k-subset problem* is a parameterized problem Q whose solutions for an instance (I, k) are all subsets of size at most k of a domain set, and is *superset-closed* if any superset of a solution of Q is also a solution of Q.

Theorem 4. *If a k-subset problem Q is superset-closed and has an FPT algorithm to enumerate all its minimal solutions, the number of which is bounded by a function of k, then Q Reconfiguration parameterized by k is in FPT, as well as the search and shortest path variants.*

Proof. By enumerating all minimal solutions of Q, we compute the set M of all elements v of the domain set such that v is in a minimal solution to Q. For (I, S, T, k, ℓ) an instance of Q Reconfiguration, we show that there exists a reconfiguration sequence from S to T if and only if there exists a reconfiguration sequence from $S \cap M$ to $T \cap M$ that uses only subsets of M.

Each set U in the reconfiguration sequence from S to T is a solution, hence contains at least one minimal solution in $U \cap M$; $U \cap M$ is a superset of the minimal solution and hence also a solution. Moreover, since any two consecutive solutions U and U' in the sequence differ by a single element, $U \cap M$ and $U' \cap M$ differ by at most a single element. By replacing each subsequence of identical sets by a single set, we obtain a reconfiguration sequence from $S \cap M$ to $T \cap M$ that uses only subsets of M.

The reconfiguration sequence from $S \cap M$ to $T \cap M$ using only subsets of M can be extended to a reconfiguration sequence from S to T by transforming S to $S \cap M$ in $|S \setminus M|$ steps and transforming $T \cap M$ to T in $|T \setminus M|$ steps. In this sequence, each vertex in $C \setminus M$ is removed from S to form $S \setminus M$ and then readded to form T from $T \setminus M$. For each vertex $v \in C \setminus M$, we can choose instead to add v to each solution in the sequence, thereby decreasing ℓ by two (the steps needed to remove and then readd v) at the cost of increasing by one the capacity used in the sequence from $S \cap M$ to $T \cap M$. This choice can be made independently for each of these $\mathcal{E} = |C \setminus M|$ vertices.

Consequently, (I, S, T, k, ℓ) is a yes-instance for Q Reconfiguration if and only if one of the $\mathcal{E}+1$ reduced instances $(I, S\cap M, T\cap M, k-e, \ell-2(\mathcal{E}-e))$, for $0 \leq e \leq \mathcal{E}$ and $\mathcal{E} = |C \backslash M|$, is a yes-instance for Q' Reconfiguration: we define Q' as a k-subset problem whose solutions for an instance (I, k) are solutions of instance (I, k) of Q that are contained in M. To show that Q' Reconfiguration is in *FPT*, we observe that the number of nodes in the reconfiguration graph for Q' is bounded by a function of k: each solution of Q' is a subset of M, yielding at most $2^{|M|}$ nodes, and $|M|$ is bounded by a function of k. □

Corollary 5. Bounded Hitting Set Reconfiguration*,* Feedback Vertex Set in Tournaments Reconfiguration*, and* Minimum Weight SAT in Bounded CNF Formulas Reconfiguation *parameterized by k are in FPT.*

For Bounded Hitting Set, the proof of Theorem 4 can be strengthened to develop a polynomial reconfiguration kernel. In fact, we use the ideas in Theorem 4 to adapt a special kernel that retains all minimal k-hitting sets in the reduced instances [16].

Theorem 6. Bounded Hitting Set Reconfiguration *parameterized by k has a polynomial reconfiguration kernel.*

Proof. We let (G, S, T, k, ℓ) be an instance of Bounded Hitting Set Reconfiguration: G is a family of sets of vertices of size at most r and each of S and T is a hitting set of size at most k, that is, a set of vertices intersecting each set in G. We form a reconfiguration kernel using the reduction algorithm $\mathcal{A}$ of Damaschke and Molokov [16]: $G' = \mathcal{A}(G)$ contains all minimal hitting set solutions of size at most k, and is of size at most $(r-1)k^r + k$.

$V(G')$ includes all minimal k-hitting sets, and the k-hitting sets for G' are actually those k-hitting sets for G that are completely included in $V(G')$. Therefore, as in the proof of Theorem 4, (G, S, T, k, ℓ) is a yes-instance for Bounded Hitting Set Reconfiguration if and only if one of the $\mathcal{E}+1$ reduced instances $(G', S \cap V(G'), T \cap V(G'), k-e, \ell - 2(\mathcal{E}-e))$, for $0 \leq e \leq \mathcal{E}$, is a yes-instance for Bounded Hitting Set Reconfiguration.

Notice that unlike in the proof of Theorem 4, here the set containing all minimal solutions can be computed in polynomial time, whereas Theorem 4 guarantees only a fixed-parameter tractable procedure. □

Bounded Hitting Set generalizes any deletion problem for a hereditary property with a finite forbidden set:

Corollary 7. *If π is a hereditary graph property with a finite forbidden set, then* π-del-reconf(G, S, T, k, ℓ) *parameterized by k has a polynomial reconfiguration kernel.*

Corollary 7 does not apply to Feedback Vertex Set, for which the associated hereditary graph property is the collection of all forests; the forbidden set is the set of all cycles and hence is not finite. Indeed, Theorem 4 does not apply to Feedback Vertex Set either, since the number of minimal solutions exceeds $f(k)$ if the input graph includes a cycle of length $f(k)+1$, for any function f. While it may be possible to adapt the compact enumeration of minimal feedback vertex sets [21] for reconfiguration, we develop a reconfiguration kernel for feedback vertex set by modifying a specific kernel for the problem.

We are given an undirected graph and two feedback vertex sets S and T of size at most k. We make use of Bodlaender's cubic kernel for Feedback Vertex Set [15], modifying reduction rules (shown in italics in the rules below) to allow the reconfiguration sequence to use non-minimal solutions, and to take into account the roles of C, S_D, T_A, and O. In some cases we remove vertices from O only, as others may be needed in a reconfiguration sequence.

The reduction may introduce multiple edges, forming a multigraph. Bodlaender specifies that a double edge between vertices u and v consists of two edges with u and v as endpoints. Since we preserve certain degree-two vertices, we extend the notion by saying that there is a *double edge* between u and v if either there are two edges with u and v as endpoints, one edge between u and v and one path from u to v in which each internal vertex is of degree two, or two paths (necessarily sharing only u and v) from u to v in which each internal vertex is of degree two. Following Bodlaender, we define two sets of vertices, a feedback vertex set A of size at most $2k$ and the set B containing each vertex with a double edge to at least one vertex in A. A *piece* is a connected component of $G[V \setminus (A \cup B)]$, the *border* of a piece with vertex set X is the set of vertices in $A \cup B$ adjacent to any vertex in X, and a vertex v in the border *governs* a piece if there is a double edge between v and each other vertex in the border. We introduce $\mathcal{E}$ to denote how much capacity we can "free up" for use in the reduced instance by removing vertices and then readding them.

Bodlaender's algorithm makes use of a repeated initialization phase in which an approximate solution A is found and B is initialized; for our purposes, we set $A = C \cup S_D \cup T_A$ in the first round and thereafter remove vertices as dictated by the application of reduction rules. Although not strictly necessary, we preserve this idea in order to be able to apply Bodlaender's counting arguments. In the following rules, v, w, and x are vertices.

Rule 1 If v has degree 0, remove v from G. *If v is in $S_D \cup T_A$, subtract 1 from ℓ. If v is in C, increment $\mathcal{E}$ by 1.*

Rule 2 If v has degree 1, remove v and its incident edge from G. *If v is in $S_D \cup T_A$, subtract 1 from ℓ. If v is in C, increment $\mathcal{E}$ by 1.*

Rule 3 If there are three or more edges $\{v, w\}$, remove all but two.

Rule 4 If v has degree 2 *and v is in O*, remove v and its incident edges from G and add an edge between its neighbours w and x; add w (respectively, x) to B if a double edge is formed, w (respectively, x) is not in $A \cup B$, and x (respectively, w) is in A.

Rule 5 If v has a self-loop, remove v and all incident edges and decrease k by 1, then restart the initialization phase.

Rule 6 If there are at least $k+2$ vertex-disjoint paths between $v \in A$ and any w and there is no double edge between v and w, add two edges between v and w, and if $w \notin A \cup B$, add w to B.

Rule 7 If for $v \in A$ there exist at least $k + 1$ cycles such that each pair of cycles has exactly $\{v\}$ as the intersection, remove v and all incident edges and decrease k by 1, then restart the initialization phase.

Rule 8 If v has at least $k + 1$ neighbours with double edges, remove v and all incident edges and decrease k by 1, then restart the initialization phase.

Rule 9 If $v \in A \cup B$ governs a piece with vertex set X and has exactly one neighbour w in X, then remove the edge $\{v, w\}$.

Rule 10 If $v \in A \cup B$ governs a piece with vertex set X and has at least two neighbours in X, then remove v and all incident edges and decrease k by 1, then restart the initialization phase. *Replaced by the following rule: If a piece with vertex set X has a border set Y such that there is a double edge between each pair of vertices in Y, remove X.*

Lemma 8. *The instance (G, S, T, k, ℓ) is a yes-instance if and only if one of the $\mathcal{E} + 1$ reduced instances $(G', S', T', k - e, \ell - 2(\mathcal{E} - e))$, for $0 \leq e \leq \mathcal{E}$, is a yes-instance.*

Proof. We show that no modification of a reduction rule removes possible reconfiguration sequences. This is trivially true for Rules 3 and 6.

The vertices removed by Rules 1, 2, and 4 play different roles in converting a reconfiguration sequence for a reduced instance to a reconfiguration sequence for the original instance. As there is no cycle that can be destroyed only by a vertex removed from O by Rule 1, 2, or 4, none of these vertices are needed. To account for the required removal (addition) of each such vertex in S_D (T_A), we remove all d such vertices and decrease ℓ by d. We can choose to leave a $v \in C_M$ in each solution in the sequence (with no impact on ℓ) or to remove and then readd v to free up extra capacity, at a cost of incrementing ℓ by two; in the reduced instance we thus remove v and either decrement k or subtract two from ℓ. Since this choice can be made for each of these vertices, $\mathcal{E}$ in total, we try to solve any of $\mathcal{E} + 1$ versions $(G', S', T', k - e, \ell - 2(\mathcal{E} - e))$ for $0 \leq e \leq \mathcal{E}$.

For each of Rules 5, 7, and 8, we show that the removed vertex v is in C_F; since the cycles formed by v must be handled by each solution in the sequence, the instance can be reduced by removing v and decrementing k. For Rule 5, $v \in C_F$ since every feedback arc set must contain v. For Rules 7 and 8, $v \in C_F$, since any feedback vertex set not containing v would have to contain at least $k + 1$ vertices, one for each cycle.

For Rule 9, Bodlaender's Lemma 8 shows that the removed edge has no impact on feedback vertex sets.

For Rule 10, we first assume that Rule 9 has been exhaustively applied, and thus each vertex in the border has two edges to X. By Fact 1 for π the set of acyclic graphs, there cannot be a cycle in $G[O \cup \{v\}]$ for any $v \in S_D \cup T_A \cup O$, and hence each member of the border is in C. Lemma 9 in Bodlaender's paper shows that there is a minimum size feedback vertex set containing v: even if all the neighbours of v in the border are included in a feedback vertex set, at least one more vertex is required to break the cycle formed by v and X. There is no gain in capacity possible by replacing v in the reconfiguration sequence, and hence this particular piece is of no value in finding a solution. □

We first present the key points and lemmas in Bodlaender's counting argument and then show that, with minor modifications, the same argument goes through for our modified reduction rules and altered definition of *double edge*. In Bodlaender's proof, the size of the reduced instance is bounded by bounding the sizes of A and B (Lemma 10), bounding the number of pieces (Lemma 12), and bounding the size of each piece. Crucial to the proof of Lemma 12 is Lemma 11, as the counting associates each piece with a pair of vertices in its border that are not connected by a double edge and then counts the number of pieces associated with each different type of pair. We use Lemma 9 in the discussion below.

Lemma 9. *[15] Suppose $v \in A \cup B$ governs a piece with vertex set X. Suppose there are at least two edges with one endpoints v and one endpoint in X. Then there is a minimum size feedback vertex set in G that contains v.*

Lemma 10. *[15] In a reduced instance, there are at most $2k$ vertices in A and at most $2k^2$ vertices in B.*

Lemma 11. *[15] Suppose none of the Rules 1–10 can be applied to G. Suppose $Y \subseteq V$ is the border of a piece in G. Then there are two disjoint vertices $v, w \in Y$ such that $\{v, w\}$ is not a double edge.*

Lemma 12. *[15] Suppose we have a reduced instance. There are at most $8k^3 + 9k^2 + k$ pieces.*

Lemma 13. *Each reduced instance has $O(k^3)$ vertices and $O(k^3)$ edges, and can be obtained in polynomial time.*

Proof. Our modifications to Rules 1–3 and 5–9 do not have an impact on the size of the kernel. Although our Rule 4 preserves some vertices in A of degree two, due to the initialization of A to be $C \cup S_D \cup T_A$, and hence of size at most $2k$, the bound on B and hence Lemma 10 follows from Rule 8. In essence, our extended definition of double edges handles the degree-two vertices that in Bodlaender's constructions would have been replaced by an edge.

To claim the result of Lemma 12, it suffices to show that Lemma 11 holds for our modified rules. Bodlaender shows that if there is a piece such that each pair of vertices in the border set is connected by a double edge, Rule 10 along with Rule 9 can be applied repeatedly to remove vertices from the border of the piece and thereafter Rules 2 and 1 to remove the piece entirely.

To justify Rule 10, Bodlaender shows in Lemma 9 that if $v \in A \cup B$ governs a piece with vertex set X and there are at least two edges between v and X, then there is a minimum size feedback vertex set in G that contains v. For our purposes, however, since there may be non-minimum size feedback vertex sets used in the reconfiguration sequence, we wish to retain v rather than removing it. Our modification to Rule 10 allows us to retain v, handling all the removals from the piece without changing the border, and thus establishing Lemma 11, as needed to prove Lemma 12. In counting the sizes of pieces, our modifications result in extra degree-two vertices. Rule 4 removes all degree-two vertices in O, and hence the number of extra vertices is at most $2k$, having no effect on the asymptotic count. □

Theorem 14. Feedback Vertex Set Reconfiguration *and the search variant parameterized by k are in* FPT.

Proof. Since the number of reduced instances is $\mathcal{E} + 1 \leq |C| + 1 \leq k + 1$, as a consequence of Lemmas 8 and 13, we have a reconfiguration kernel, proving the first result. For the search version, we observe that we can generate the reconfiguration graph of the reduced yes-instance and use it to extract a reconfiguration sequence. We demonstrate that we can form a reconfiguration sequence for (G, S, T, k, ℓ) from the reconfiguration sequence σ for the reduced yes-instance $(G', S', t', k - e, \ell - 2(\mathcal{E} - e))$. We choose an arbitrary partition of the vertices removed from G by Rules 1 and 2 into two sets, K (the ones to keep) of size e and M (the ones to modify) of size $\mathcal{E} - e$. We can modify σ into a sequence σ' in which all vertices in K are added to each set; clearly no set will have size greater than k. Our reconfiguration sequence then consists of $\mathcal{E} - e$ steps each deleting an element of M, the sequence σ', and $\mathcal{E} - e$ steps each adding an element of M, for a length of at most $(\mathcal{E} - e) + (\ell - (\mathcal{E} - e)) + (\mathcal{E} - e) \leq \ell$, as needed. □

4 Hardness Results

The reductions presented in this section make use of the forbidden set characterization of heredity properties. A π-*critical graph* H is a (minimal) graph in the forbidden set $\mathcal{F}_\pi$ that has at least two vertices; we use the fact that $H \notin \pi$, but the deletion of any vertex from H results in a graph in π. For convenience, we will refer to two of the vertices in a π-critical graph as *terminals* and the rest as *internal vertices.* We construct graphs from multiple copies of H. For a positive integer c, we let H_c^* be the ("star") graph obtained from each of c copies H_i of H by identifying an arbitrary terminal v_i, $1 \leq i \leq c$, from each H_i; in H_c^* vertices v_1 through v_c are replaced with a vertex w, the *gluing vertex of* v_1 *to* v_c, to form a graph with vertex set $\cup_{1 \leq i \leq c}(V(H_i) \setminus \{v_i\}) \cup \{w\}$ and edge set $\cup_{1 \leq i \leq c}\{\{u, v\} \in E(H_i) \mid v_i \notin \{u, v\}\} \cup \cup_{1 \leq i \leq c}\{\{u, w\} \mid \{u, v_i\} \in E(H_i)\}$. A terminal is *non-identified* if it is not used in forming a gluing vertex. In Figure 1, H is a K_3 with terminals marked black and gray; H_4^* is formed by identifying all the gray terminals to form w.

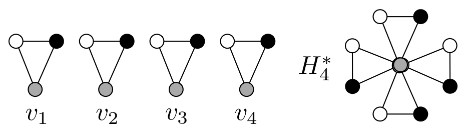

Fig. 1. An example H_c^*

Theorem 15. *Let π be any hereditary property satisfying the following:*

– *For any two graphs G_1 and G_2 in π, their disjoint union is in π.*

– *There exists an $H \in \mathcal{F}_\pi$ such that if H^*_c is the graph obtained from identifying a terminal from each of c copies of H, then $R = H^*_c[V(H^*_c) \setminus \{u_1 \dots u_c\}]$ is in π, where $u_1 \dots u_c$ are the non-identified terminals in the c copies of H.*

Then each of the following is at least as hard as π-SUBSET(G, k):

1. *π-DEL-RECONF(G, S, T, k, ℓ) parameterized by ℓ, and*
2. *π-SUB-RECONF(G, S, T, k, ℓ) parameterized by $k + \ell$.*

Proof. Given an instance of π-SUBSET(G, k) and a π-critical graph H satisfying the hypothesis of the lemma, we form an instance of π-DEL-RECONF$(G', S, T, |V(G)| + k, 4k)$, with G', S, and T defined below. The graph G' is the disjoint union of G and a graph W formed from k^2 copies of H, where $H_{i,j}$ has terminals $\ell_{i,j}$ and $r_{i,j}$. We let a_i, $1 \leq i \leq k$, be the gluing vertex of $\ell_{i,1}$ through $\ell_{i,k}$, and let b_j, $1 \leq j \leq k$, be the gluing vertex of $r_{1,j}$ through $r_{k,j}$, so that there is a copy of H joining each a_i and b_j. We let $A = \{a_i \mid 1 \leq i \leq k\}$, $B = \{b_j \mid 1 \leq j \leq k\}$, $S = V(G) \cup A$, and $T = V(G) \cup B$. Clearly $|V(G')| = |V(G)| + 2k + k^2(|V(H)| - 2)$ and $|S| = |T| = |V(G)| + k$. Moreover, each of $V(G') \setminus S$ and $V(G') \setminus T$ induce a graph in π, as each consists of k disjoint copies of H^*_k with one of the terminals removed from each H in H^*_k.

Suppose the instance of π-DEL-RECONF$(G', S, T, |V(G)| + k, 4k)$ is a yes-instance. As there is a copy of H joining each vertex of A to each vertex of B, before deleting $a \in A$ from S the reconfiguration sequence must add all of B to ensure that the complement of each intermediate set induces a graph in π. Otherwise, the complement will contain at least one copy of H as a subgraph and is therefore not in π. The capacity bound of $|V(G)| + k$ implies that the reconfiguration sequence must have deleted from S a subset $S' \subseteq V(G)$ of size at least k such that $V(G') \setminus (S \setminus S') = S' \cup B$ induces a subgraph in π. Thus, $G[S'] \in \pi$, and hence π-SUBSET(G, k) is a yes-instance.

Conversely if the instance of π-SUBSET(G, k) is a yes-instance, then there exists $V' \subseteq V(G)$ such that $|V'| = k$ and $G[V'] \in \pi$. We form a reconfiguration sequence between S and T by first deleting all vertices in V' from S to yield a set of size $|V(G)|$. $G'[V(G') \setminus (S \setminus V')]$ consists of the union of $G'[V'(G) \setminus S]$ and $G'[V'] = G[V']$, both of which are in π. Next we add one by one all vertices of B, then delete one by one all vertices of A and then add back one by one each vertex in the set V' resulting in a reconfiguration sequence of length $k + k + k + k = 4k$. It is clear that in every step, the complement of the set induces a graph in π.

Thus we have showed that π-SUBSET(G, k) is a yes-instance if and only if there is a path of length at most $4k$ between S and T in $R^{\pi}_{\text{DEL}}(G', |V(G)| + k)$. Since $|V(G')| - (|V(G)| + k) = k + k^2(|V(H)| - 2))$, this implies that π-SUBSET(G, k) is a yes-instance if and only if there is a path of length at most $4k$ between $V(G') \setminus S$ and $V(G') \setminus T$ in $R^{\pi}_{\text{SUB}}(G', k + k^2(|V(H)| - 2))$. Therefore, π-SUB-RECONF(G, S, T, k, ℓ) parameterized by $k + \ell$ is at least as hard as π-SUBSET(G, k), proving the second part. □

It is easy to see that for π the collection of all edgeless graphs, or all forests, or all bipartite graphs, the hypothesis of Theorem 15 is satisfied. Since the π-SUBSET(G, k) problem is $W[1]$-hard for these properties [22], it follows that:

Corollary 16. * Vertex Cover Reconfiguration, Feedback Vertex Set Reconfiguration, *and* Odd Cycle Transversal Reconfiguration *parameterized by ℓ are all $W[1]$-hard and* Independent Set Reconfiguration, Forest Reconfiguration, *and* Bipartite Subgraph Reconfiguration *parameterized by $k + \ell$ are all $W[1]$-hard.*

We obtain further results for properties not covered by Theorem 15. Lemma 17 handles the collection of all cliques, which does not satisfy the first condition of the theorem and the collection of all *cluster graphs* (disjoint unions of cliques), which satisfies the first condition but not the second. Moreover, as π-subset(G, k) is in *FPT* for π the collection of all cluster graphs [22], Theorem 15 provides no lower bounds.

Lemma 17. * Clique Reconfiguration *and* Cluster Subgraph Reconfiguration *parameterized by $k + \ell$ are $W[1]$-hard.*

As neither Dominating Set nor its parametric dual is a hereditary graph property, Theorem 15 is inapplicable; we instead use a construction specific to the problem in Lemma 18.

Lemma 18. * Dominating Set Reconfiguration *and* (Unbounded) Hitting Set Reconfiguration *parameterized by $k + \ell$ are $W[2]$-hard.*

5 Conclusions and Directions for Further Work

Our results constitute the first study of the parameterized complexity of reconfiguration problems. We give a general paradigm, the reconfiguration kernel, for proving fixed-parameter tractability, and provide hardness reductions that apply to problems associated with hereditary graph properties. Our result on cluster graphs (Lemma 17) demonstrates the existence of a problem that is fixed-parameter tractable [22], but whose reconfiguration version is W-hard when parameterized by k. It remains open whether there exists an NP-hard problem for which the reconfiguration version is in *FPT* if parameterized by ℓ.

Our *FPT* algorithms for reconfiguration of Bounded Hitting Set and Feedback Vertex Set have running times of $O^*(2^{O(k \lg k)})$. Further work is needed to determine whether the running times can be improved to $O^*(2^{O(k)})$, or whether these bounds are tight under the *Exponential Time Hypothesis.*

We observe connections to another well-studied paradigm, local search [23], where the aim is to find an *improved solution* at distance ℓ of a given solution S. Not surprisingly, as in local search, the problems we study turn out to be hard even in the parameterized setting when parameterized by ℓ. Other natural directions to pursue (as in the study of local search) are the parameterized complexity of reconfiguration problems in special classes of graphs and of non-graph reconfiguration problems, as well as other parameterizations.

Acknowledgements. The second author wishes to thank Marcin Kamiński for suggesting the examination of reconfiguration in the parameterized setting.

References

1. Bonamy, M., Bousquet, N.: Recoloring bounded treewidth graphs. In: Proc. of Latin-American Algorithms Graphs and Optimization Symposium. (2013)
2. Bonsma, P.S., Cereceda, L.: Finding paths between graph colourings: PSPACE-completeness and superpolynomial distances. Theor. Comput. Sci. **410**(50) (2009) 5215–5226
3. Cereceda, L., van den Heuvel, J., Johnson, M.: Connectedness of the graph of vertex-colourings. Discrete Mathematics **308**(56) (2008) 913–919
4. Cereceda, L., van den Heuvel, J., Johnson, M.: Mixing 3-colourings in bipartite graphs. European Journal of Combinatorics **30**(7) (2009) 1593–1606
5. Cereceda, L., van den Heuvel, J., Johnson, M.: Finding paths between 3-colorings. Journal of Graph Theory **67**(1) (2011) 69–82
6. Ito, T., Kamiński, M., Demaine, E.D.: Reconfiguration of list edge-colorings in a graph. Discrete Applied Mathematics **160**(15) (2012) 2199–2207
7. Hearn, R.A., Demaine, E.D.: PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. Theor. Comput. Sci. **343**(1-2) (2005) 72–96
8. Ito, T., Demaine, E.D., Harvey, N.J.A., Papadimitriou, C.H., Sideri, M., Uehara, R., Uno, Y.: On the complexity of reconfiguration problems. Theor. Comput. Sci. **412**(12-14) (2011) 1054–1065
9. Gopalan, P., Kolaitis, P.G., Maneva, E.N., Papadimitriou, C.H.: The connectivity of boolean satisfiability: computational and structural dichotomies. SIAM J. Comput. **38**(6) (2009) 2330–2355
10. Bonsma, P.: The complexity of rerouting shortest paths. In: Proc. of Mathematical Foundations of Computer Science. (2012) 222–233
11. Kamiński, M., Medvedev, P., Milanič, M.: Shortest paths between shortest paths. Theor. Comput. Sci. **412**(39) (2011) 5205–5210
12. Haas, R., Seyffarth, K.: The k-Dominating Graph (2012) arXiv:1209.5138.
13. Suzuki, A., Mouawad, A.E., Nishimura, N.: Reconfiguration of dominating sets. submitted
14. Downey, R.G., Fellows, M.R.: Parameterized complexity. Springer-Verlag, New York (1997)
15. Bodlaender, H.L.: A cubic kernel for feedback vertex set. In: Proc. of the 24th Annual Conference on Theoretical Aspects of Computer Science. (2007) 320–331
16. Damaschke, P., Molokov, L.: The union of minimal hitting sets: Parameterized combinatorial bounds and counting. Journal of Discrete Algorithms **7**(4) (2009) 391–401
17. Flum, J., Grohe, M.: Parameterized complexity theory. Springer-Verlag, Berlin (2006)
18. Niedermeier, R.: Invitation to fixed-parameter algorithms. Oxford University Press (2006)
19. Lewis, J.M., Yannakakis, M.: The node-deletion problem for hereditary properties is NP-complete. Journal of Computer and System Sciences **20**(2) (1980) 219–230
20. Mouawad, A.E., Nishimura, N., Raman, V., Simjour, N., Suzuki, A.: On the parameterized complexity of reconfiguration problems (2013) arXiv:1308.2409.
21. Guo, J., Gramm, J., Hüffner, F., Niedermeier, R., Wernicke, S.: Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization. Journal of Computer and System Sciences **72**(8) (2006) 1386–1396

22. Khot, S., Raman, V.: Parameterized complexity of finding subgraphs with hereditary properties. Theor. Comput. Sci. **289**(2) (2002) 997–1008
23. Fellows, M.R., Rosamond, F.A., Fomin, F.V., Lokshtanov, D., Saurabh, S., Villanger, Y.: Local search: is brute-force avoidable? In: Proc. of the 21st International Joint Conference on Artifical Intelligence. (2009) 486–491