Distributed Vertex Cover Reconfiguration

Keren Censor-Hillel
Department of Computer Science, Technion, Haifa, Israel

Yannic Maus
Institute of Software Technology, TU Graz, Austria

Shahar Romem-Peled
Department of Computer Science, Technion, Haifa, Israel

Tigran Tonoyan
Department of Computer Science, Technion, Haifa, Israel

Abstract

Reconfiguration schedules, i.e., sequences that gradually transform one solution of a problem to another while always maintaining feasibility, have been extensively studied. Most research has dealt with the decision problem of whether a reconfiguration schedule exists, and the complexity of finding one. A prime example is the reconfiguration of vertex covers. We initiate the study of *batched vertex cover reconfiguration*, which allows to reconfigure multiple vertices concurrently while requiring that any adversarial reconfiguration order within a *batch* maintains feasibility. The latter provides robustness, e.g., if the simultaneous reconfiguration of a batch cannot be guaranteed. The quality of a schedule is measured by the number of batches until all nodes are reconfigured, and its *cost*, i.e., the maximum size of an intermediate vertex cover.

To set a baseline for batch reconfiguration, we show that for graphs belonging to one of the classes $\{\mathsf{cycles}, \mathsf{trees}, \mathsf{forests}, \mathsf{chordal}, \mathsf{cactus}, \mathsf{even\text{-}hole\text{-}free}, \mathsf{claw\text{-}free}\}$, there are schedules that use $O(\varepsilon^{-1})$ batches and incur only a $1+\varepsilon$ multiplicative increase in cost over the best sequential schedules. Our main contribution is to compute such batch schedules in a distributed setting $O(\varepsilon^{-1}\log^* n)$ rounds, which we also show to be tight. Further, we show that once we step out of these graph classes we face a very different situation. There are graph classes on which no efficient distributed algorithm can obtain the best (or almost best) existing schedule. Moreover, there are classes of bounded degree graphs which do not admit any reconfiguration schedules without incurring a large multiplicative increase in the cost at all.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms; Mathematics of computing → Approximation algorithms; Theory of computation → Distributed algorithms

Keywords and phrases reconfiguration, vertex cover, network decomposition

Digital Object Identifier 10.4230/LIPIcs.ITCS.2022.36

Related Version *Full Version*: `https://arxiv.org/abs/2109.06601`

Funding This project has been supported by the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no. 755839.

1 Introduction

Consider a huge network of computers connected via communication links, in which each communication link needs to be monitored at all times by at least one of its endpoints (computers). If the network is abstracted as a graph with each node representing a computer and each edge representing a communication link, the set of all monitoring computers is a *vertex cover*. A usual constraint is to have small vertex covers (consisting of few computers, in our case), and the problem of finding minimal vertex cover is one of the classic problems of algorithmic graph theory and has been extensively studied in various graph classes and computational models. In the setting we are interested in, the system might decide at some point in time to switch to another vertex cover, say, to evenly distribute the load of monitoring over the nodes. To ensure correct system performance, it is natural to require each communication link be monitored at all times, even during the switching process, and at the same time to save resources by keeping the vertex cover size small at all times. This naturally leads us to the vertex cover reconfiguration problem.

More generally, reconfiguration problems ask the following type of questions: *Given two solutions to a problem, is it always possible to gradually move from one solution to the other by changing* ***one element*** *at a time, while always maintaining a feasible solution?*

Reconfiguration problems thus explore reachability in a graph over the solutions, and as such they have been extensively studied for various problems. Notable examples are colorings [8, 24, 3], matchings [23], independent sets and vertex covers [27, 33]. In the vertex cover reconfiguration problem one needs to find a schedule that moves from a given first vertex cover to a given second vertex cover by changing the membership of one vertex at a time, and while ensuring that each intermediate set is a valid vertex cover (**feasibility**). Traditionally, the emphasis has been on the size of the intermediate solutions – the problem is trivial with no size bound – while the elementary steps consist of adding or removing a single vertex.

Distributed reconfiguration and its motivation. In this work, we initiate the study of *distributed vertex cover reconfiguration.* In contrast to the previously discussed setup, in huge networks, the states of vertices might have to change concurrently, for various reasons; e.g., to obtain short schedules, or simply if there is no single entity controlling the whole network, and communication between far-apart computers is too costly to agree on a global reconfiguration schedule. Motivated by this, the goal of this work is to exploit parallelism for reconfiguration schedules. For vertex cover reconfiguration, this leads us to allow changing the membership in the intermediate vertex cover of more than one vertex in each step. We formally capture this setting by introducing the concept of batch reconfiguration.

In *batch reconfiguration*, one is allowed to change a *batch* of an **unbounded number of elements** in a single reconfiguration step, as opposed to the previous *sequential reconfiguration*, which changes a single vertex at a time. However, such a solution is not practically robust, in the following sense. Suppose that implementing the change for a vertex is not an immediate operation and could rather take a bit of time. Then, changing several vertices concurrently may result in a sequence of changes in these vertices, with an unpredictable order. As a result, although we aim at reconfiguring all vertices at once in one swipe, in reality what could happen is that we get an arbitrary sequence of changes, which can easily violate feasibility in an adversarial execution of a batch.

In light of the above, in addition to feasibility, we require a **robustness** condition for batch reconfiguration schedules. The goal of a robustness condition is to guarantee that no matter in which order the elements of a batch are eventually executed, feasibility is never violated. We require that *the set of vertices that are reconfigured within a batch is always an independent set.* This promises that each edge is always covered, also within any internal ordering of a batch.

The tradeoff between the number of batches and the solution size. When computing a schedule between two covers, typically denoted by α and β, batching brings the advantage of short schedules, but comes with a proportional overhead in solution sizes. We show via a pigeonhole argument that for some instances of vertex covers α and β, every reconfiguration schedule with t batches necessarily creates an intermediate solution of size at least $(1 +$

$1/t) \cdot \max\{|\alpha|, |\beta|\}$. We desire to get as close as possible to this optimal length-vs.-size tradeoff. We evaluate the quality of a batch reconfiguration schedule for vertex cover by two measures: the length of the schedule, i.e., the number of batches, and the maximum size of an intermediate vertex cover in the schedule, which, by the robustness condition, includes all possible intermediate vertex covers that can occur in any internal ordering of each batch. In view of the natural barrier of $\max\{|\alpha|, |\beta|\}$ on the worst-case intermediate vertex cover size, as well as the necessary overhead given by batch schedules, we say a reconfiguration schedule is an (η, c)-*approximation* if the size of the worst-case intermediate cover is at most $\eta \cdot \max\{|\alpha|, |\beta|\} + c$ (see Section 2 for formal definitions). Here, η is between 1 and 2, and is normally related to the number of batches, while c is often a constant (e.g., the trivial 2-batch schedule that adds all nodes in $\beta \setminus \alpha$, then removes all nodes in $\alpha \setminus \beta$, is a $(2, 0)$-approximation for a schedule from α to β).

1.1 Our Contribution

As we initiate the study of distributed algorithms for vertex cover reconfiguration, our first contribution is an examination of the baseline for our distributed algorithms. We provide several (tight) existential results on batch schedules on various graph classes. Then we devise distributed algorithms that nearly match our existential results and efficiently compute schedules of almost the same quality. Further, we show that once we step out of these graph classes we face a very different situation. There are graph classes on which no efficient distributed algorithm can obtain the best (or almost best) existing schedule. Moreover, there are classes of bounded degree graphs which do not admit any reconfiguration schedules without incurring a large multiplicative increase in the cost at all. We next discuss our existential (centralized) results and then discuss our distributed results.

Existential results on batch reconfiguration schedules. Our first technical contribution is a black-box compression scheme that mechanically transforms a sequential schedule into a batched one of desired length, with a proportional and unavoidable overhead. Thus, we derive batch schedules from known sequential schedules. Our result holds for *monotone* schedules (that never touch a vertex twice), which is aligned with prior work (see Section 1.2).

▶ **Theorem 1** (Schedule Compression). *Let $G = (V, E)$ be a graph with two vertex covers α, β, and let $\mathcal{S}$ be a monotone sequential schedule from α to β that is an (η, c)-approximation, for a real $\eta \geq 1$ and an integer $c \geq 0$. For every $\varepsilon \in (0, 1)$, $\mathcal{S}$ can be transformed into a monotone $(2\lceil 1/\varepsilon \rceil)$-batch schedule $\mathcal{S}'$ that is an $(\eta + \varepsilon, c + 1)$-approximation.*

Based on known separator theorems (e.g., [32]), we observe that a number of graph classes possessing small separators admit $(1, O(\sqrt{n}))$-approximation schedules, where n is the number of vertices. It is also known that cactus graphs have $(1, 2)$-approximation schedules [25] (see Theorem 8). We provide a simple and unified proof of these results. Moreover, the same results apply for graphs G for which $G[\alpha \oplus \beta]$ belongs to the corresponding graph class. The property "$G[\alpha \oplus \beta]$ is a cactus graph" holds for all graphs G in the following graph classes: {cycles, trees, forests, chordal, cactus, even-hole-free, claw-free}. Combined with our batching scheme, we get the following theorem.

▶ **Theorem 2.** *Let $\varepsilon > 0$. Let $G = (V, E)$ be a graph with a pair α, β of vertex covers. If G belongs to one of the graph classes* {cactus, chordal, even-hole-free, claw-free}*, there is a monotone $(2\lceil 1/\varepsilon \rceil)$-batch $(1 + \varepsilon, 3)$-approximation schedule from α to β. If G is planar, there is a monotone $(2\lceil 2/\varepsilon \rceil)$-batch $(1 + \varepsilon, O(\varepsilon^{-1}))$-approximation schedule from α to β.*

We show in Section 3 that the tradeoff between the number of batches and the approximation overhead in Theorem 1 is (almost) best possible. This implies that the batch schedules from Theorem 2 are nearly optimal in terms of the length vs. approximation tradeoff.

A natural guess would be that the $(1+\varepsilon)$-type approximations á la Theorem 2 extend to bounded degree or bounded arboricity graphs[1] (the latter contains planar graphs). In the full version, we show that that is not the case: for any $d \geq 4$, there are infinitely many d-regular Ramanujan graphs G ([34]) with two vertex covers α, β, for which no schedule can be better than a $(2 - O(1/\sqrt{d}))$-approximation. The proof exploits the *expansion* properties of such graphs. On the positive side, we give a $(2 - \Omega(1/\lambda))$-approximation for any graph of arboricity λ, by a simple monotone *greedy schedule* that reconfigures α-nodes in an increasing degree order, and a similar approximation can even be achieved with a *4-batch* schedule.

▶ **Theorem 3.** *For every $d \geq 4$, there is an infinite class of d-regular graphs G_i, $i \geq 1$, with vertex covers α_i, β_i, that do not admit $\left(2 - 4/(\sqrt{d+1}+1), 0\right)$-approximation schedules. Any graph G of arboricity λ with vertex covers α, β has a sequential $(2 - 1/(2\lambda), 1)$-approximation and a* 4-batch *$(2 - 1/(2\lambda+1), 1)$-approximation schedules.*

Note that the batching in Theorem 3 is better than if we combined the sequential result with Theorem 1: in order to give any non-trivial approximation, the latter would take at least 4λ batches instead of the 4 taken by Theorem 3.

Besides motivation through distributed reconfiguration, we believe that batch reconfiguration is a meaningful concept in its own right and deserves separate study, as it adds to the notions of parallelism that can be introduced into solving a problem.

Distributed computation of reconfiguration schedules. A natural setting where batch reconfiguration may appear is when nodes themselves, as distributed autonomous computers, execute a reconfiguration schedule by exchanging information with others. This immediately raises the question of *computing* schedules in such distributed settings.

We focus on the LOCAL model of distributed computing, where nodes synchronously send messages to their neighbors in the underlying network graph (which also serves as the problem instance) and give efficient algorithms for computing batch schedules. In the distributed reconfiguration problem, each node receives as input whether it is in α or β (or both) and outputs its own part of the schedule (i.e. at which rounds a vertex leaves the cover or joins it).

At the core of most of our algorithmic results is a graph decomposition that we call a *small separator decomposition* and which might be of independent interest. Roughly speaking (and hiding many technical details), it is a decomposition into small diameter clusters and a small separator set S (e.g., an ε fraction of $|\alpha| + |\beta|$, for a small $\varepsilon > 0$), such that there are no *inter-cluster edges.* At a very high level, we show that if there exists a good approximation schedule for each cluster, we can compute a batch schedule for the whole graph with only a small approximation overhead and in a number of rounds that is linear in the maximum cluster diameter.

▶ **Theorem** (Informal version of Theorem 17). *Let $G = (V, E)$ be a graph with two vertex covers α and β, and a small separator decomposition with a separator set S and clusters $(C_i)_{i=1}^{k}$ of diameter d. If each subgraph $G[C_i]$ admits a monotone ℓ-batch (η, c)-approximation schedule, then a $(2\ell+2)$-batch $(\eta, |S| + k \cdot c)$-approximation schedule for α and β can be computed in $O(d)$ rounds in the* LOCAL *model.*

[1] The *arboricity* of a graph is the minimum number of forests that are needed to cover its edge set.

As can be seen in the theorem above, it is crucial to have a graph decomposition both with a small size separator, as well as few clusters. Combining known network decompositions [1, 30, 39] and tools from distributed combinatorial optimization [17] together with our new machinery, we get a $\text{poly}\log(n)$-round algorithm to compute a small separator decomposition with logarithmic-diameter clusters in general graphs. While the resulting decomposition has a small separator set, we use additional postprocessing to also reduce the number of clusters (while keeping the bounds on cluster diameters and the separator size). This allows us to distributively implement batch-scheduling algorithms known for any *hereditary* graph class. For instance, in $\text{poly}\log(n)$ rounds, we can construct an $O(1/\varepsilon)$-batch $(1+\varepsilon, O(\varepsilon^{-1}))$-approximation schedules for planar graphs. See the full version for details.

For other special graph classes, we obtain much stronger results, that is, we can compute small separator decompositions *super-fast* (i.e., in $O(\log^* n)$-rounds). This holds in particular for *cactus* graphs. More concretely (see Lemma 21), there is a super-fast distributed algorithm that for any connected graph G and vertex covers α, β such that $\alpha \oplus \beta$ induces a cactus graph, computes a small separator decomposition with few clusters.

For such graphs G, we have $(1, 2)$-approximation schedules [25] (see also Theorem 8). Together with Theorem 17 and Lemma 21, this implies the following theorem.

▶ **Theorem 4** (Cactus Reconfiguration). *For each $\varepsilon > 0$, there is a* LOCAL *algorithm that for any connected n-node graph G and vertex covers α and β such that $G[\alpha \oplus \beta]$ is a cactus graph, computes an $O(1/\varepsilon)$-batch $(1+\varepsilon, 3)$-approximation schedule in $O(\log^*(n)/\varepsilon)$ rounds.*

We also show that *the runtime of Theorem 4 is asymptotically tight.* Concretely, in the full version, we use an argument based on Ramsey Theory to show that for every $\varepsilon \in (0, 1)$, there is no algorithm with runtime $c \cdot \log^*(n)$, with a small enough constant $c > 0$, that constructs $(1/\varepsilon)$-batch $(2-\varepsilon)$-approximation schedules on the class of *cycle graphs.*

So far, we have seen that we can obtain $\approx (1+\varepsilon)$-approximation for planar graphs, in $\text{poly}\log n$ rounds, and for cactus graphs, in $O(\log^* n)$ rounds. The next theorem provides super-fast algorithms for batch scheduling for general graphs, at the cost of an increased schedule length and approximation (which, by Theorem 3, cannot be improved by much).

▶ **Theorem 5.** *Let G be a graph with vertex covers α and β, such that $G[\alpha \oplus \beta]$ has arboricity at most λ, with λ known to all nodes. For every $\varepsilon > 0$, there exists a $O(\log^*(n)/\varepsilon)$-round* LOCAL *algorithm that computes an $O\left(\lambda/\varepsilon^2\right)$-batch $(2 - 1/(2\lambda) + \varepsilon, 1)$-approximation schedule from α to β.*

The algorithm from Theorem 5 is a non-trivial adaptation of the greedy algorithm from Theorem 3 to the distributed setting. We briefly discuss the underlying ideas in Section 4.5.

Theorem 17 suggests that if every cluster in the decomposition admits a good reconfiguration schedule, one can compute a good schedule for the whole graph efficiently. We show that this condition is necessary, in the following strict sense: there are graphs for which there exist good schedules, and they can be quickly decomposed, but just because some clusters do not have good schedules, no *sub-linear* distributed algorithm can compute a good schedule.

▶ **Theorem** (Informal version). *For each $\varepsilon \in (0, 1)$, there is an infinite family $\mathcal{G}$ of graphs such that every graph in $\mathcal{G}$ admits a $(1+\varepsilon)$-approximation schedule, but no $o(n)$-round deterministic distributed algorithm can compute a $(2-\varepsilon)$-approximation schedule.*

The formal version of the previous statement appears in the full version.

1.2 Related Work

Reconfiguration problems have long been studied under various guises (e.g., in the form of puzzles [19]), but a more systematic study has appeared rather recently, in [22]. The high level picture of the area is that for NP-complete source problems, the reconfiguration variants are usually PSPACE-complete [22, 5, 19], although there are exceptions to this trend [26]. In this paper, we are only concerned with vertex cover/independent set reconfiguration; for a wider view on the subject, we refer the reader to surveys [37, 20].

Three models have been considered for vertex cover reconfiguration. The first is the Token Addition and Removal (TAR) model, which is the one we adopt in this paper, where we can move from an intermediate solution S_1 to another one, S_2, if they differ by a single vertex. In the other two models, Token Sliding (TS) and Token Jumping (TJ), two intermediate solutions S_1 and S_2 are adjacent if S_2 is obtained from S_1 by swapping a vertex v in S_1 with another one, u, which is arbitrary in TJ, but must be a neighbor of v in TS.

The trend of hardness persists in vertex cover reconfiguration too. Here, the decision problem is: given two vertex covers of size at most k, is there a reconfiguration schedule transforming one into the other via TAR, and without ever having a vertex cover of size more than $k + 1$. The problem is PSPACE-complete even in usually tractable graph classes such as perfect graphs [27], graphs of bounded pathwidth or bandwidth [40], and planar graphs of degree at most 3 [19]. It is also NP-complete for bipartite graphs [33]. We refer the reader to [37, Figure 5] for a graphic depiction of the complexity landscape of the problem.

On the positive side, the vertex cover reconfiguration schedule existence problem is known to be polynomially solvable for trees and cactus graphs [36, 6, 11, 21, 27]. More relevant to our work, it is known that there are approximation-style schedules in cactus graphs, such that in all intermediate solutions, the size of the vertex cover is bounded by the larger of the two initial ones, plus 2 [36, 27]. While seemingly quite specialized, these results give schedules with similar guarantees for wider classes of graphs, such as *even-hole-free graphs* (that is, graphs without an induced even cycle), which include *chordal graphs* and *cographs*. In the same spirit, it is shown in [10] that there are schedules in general graphs that increase the maximum vertex cover by at most the *pathwidth* of the graph. All these schedules are *monotone*, i.e., every vertex changes its status at most once. This is the case with all schedules in our paper as well. In fact, as it is shown in [33], when the two vertex covers in a reconfiguration instance are *disjoint*, there always exists a monotone schedule.

Finally, we note that parallelism in reconfiguration has been considered in settings other than vertex cover reconfiguration, e.g., [28]. Distributed reconfiguration has also been studied for colorings [4] and independent sets [7]. However, the techniques used in those papers are tailored for reconfiguration of colorings and independent sets, and are unfitting for our problem. Therefore, we had to develop new and different methods in order to approach distributed reconfiguration.

Roadmap. In Section 2 we provide the basic definitions. In Section 3 we prove Theorem 1 (batch-compression) and present existential results on batch and non-batch reconfiguration schedules. In Section 4 we formally introduce the concept of small separator decompositions, prove Theorem 17, and we present all core steps to prove Theorem 4. All remaining results and missing proofs appear in the full version.

2 Problem Statement, Definitions, and Notation

A *vertex cover* in a graph $G = (V, E)$ is a subset $S \subseteq V$ such that every edge has at least one of its end-vertices in S. An *independent set* is a subset $S \subseteq V$ of vertices such that every edge has at most one of its end-vertices in S. Note that if S is a vertex cover then $V \setminus S$ is an independent set, and vice versa. We use the notation $[x] = \{0, 1, \ldots, x\}$, for an integer $x \geq 0$. For sets A, B, we let $A \oplus B = (A \setminus B) \cup (B \setminus A)$ denote their *symmetric difference.*

▶ **Definition 6** (Vertex Cover Reconfiguration Schedule). *Given a graph $G = (V, E)$ and two vertex covers $\alpha, \beta \subseteq V$ (not necessarily minimal), a* reconfiguration schedule $\mathcal{S}$ *from α to β of length ℓ and* cost s *is a sequence $(V_i)_{i \in [\ell]}$ of vertex covers of G such that*

1. *$V_0 = \alpha$ and $V_\ell = \beta$,*
2. *$\forall i \in [\ell - 1], \quad |V_i \cup V_{i+1}| \leq s$,*
3. *$\forall i \in [\ell - 1], \quad V_i \oplus V_{i+1}$ is an independent set of G.*

The sets $\mathcal{E}_i := V_i \oplus V_{i+1}$, which we call *batches*, contain the vertices that are added to or removed from the current vertex cover in order to obtain the next one. Thus, we also refer to a length-ℓ schedule as an *ℓ-batch schedule.* A reconfiguration schedule is *sequential* if $|\mathcal{E}_i| = 1$, for all $i \in [\ell - 1]$, and a *batch* schedule otherwise. A schedule $\mathcal{S}$ from α to β is *monotone* if every node $v \in V$ is changed at most once in the schedule. Note that a monotone schedule $\mathcal{S}$ only changes nodes in $\alpha \oplus \beta$, and therefore its length (if there are no empty batches) is upper bounded by $|\alpha \oplus \beta|$. In particular, a monotone schedule does not change nodes in $\alpha \cap \beta$.

The first property of Definition 6 ensures that $\mathcal{S}$ is a reconfiguration schedule from α to β. The second and third properties together ensure robustness, in the sense that not only $(V_i)_{i \in [\ell]}$ are vertex covers of size at most s, but any reconfiguration sequence between V_i and V_{i+1} yields vertex covers with the claimed size bound. Note that when in each reconfiguration step we only add or remove nodes, property 2 reduces to having $|V_i| \leq s$ for all $i \in [\ell]$.

For $\eta, c \in \mathbb{R}_+$, a reconfiguration schedule from a vertex cover α to a vertex cover β is an *(η, c)-approximation* if its cost is at most $\eta \max\{|\alpha|, |\beta|\} + c$. Note that here we do not compare to the cost of an optimal reconfiguration schedule but rather to $M = \max\{|\alpha|, |\beta|\}$; the cost of an optimal schedule is always at least M, but often it is larger than M.

▶ **Observation 7** (Reverse Schedule). *Let $\mathcal{S} = (V_i)_{i \in [\ell]}$ be a vertex cover reconfiguration schedule of cost s from α to β in graph G. Then the* reverse schedule *$\mathcal{S}' = (V'_i)_{i \in [\ell]}$, where $V'_i = V_{\ell - i}$, is a vertex cover reconfiguration schedule of cost s and length ℓ from β to α.*

A graph is *even-hole-free* if it contains no induced cycle with an even number of vertices. A graph is *chordal* if it contains no induced cycle with 4 or more vertices (in particular, it is even-hole-free). A graph is *claw-free* if it contains no induced $K_{1,3}$ sub-graph. A graph is a *cactus* graph if each of its edges belongs to at most 1 cycle. Alternatively, cactus graphs are characterized by a forbidden minor, the diamond graph, which is obtained by removing an edge from K_4; thus, they form a minor-closed family.

The LOCAL model of distributed computing [31, 38]. A connected graph is abstracted as an n-node network $G = (V, E)$. Communications happen in synchronous rounds. Per round, each node can send one (unbounded size) message to each of its neighbors. Further, each vertex has a unique ID from a space of size poly n. We say that the nodes of graph G distributively compute a reconfiguration schedule $\mathcal{S} = (V_i)_{i \in [\ell]}$ if each node $v \in V$ knows its membership in each V_i. We emphasize that the length of a schedule, i.e., the number of batches, and the distributed time to compute the schedule are separate measures.

Parameters and notation. Let $G = (V, E)$ be a graph with two vertex covers α and β. For a subset $U \subseteq V$, we let $G[U]$ denote the graph induced by U. We use $n = |V|$ for the number of vertices, Δ for the maximum degree, and λ for the arboricity of G. We use the notation $M = \max\{|\alpha|, |\beta|\}$, $m = \min\{|\alpha|, |\beta|\}$, $D = \alpha \oplus \beta$, and $X = \alpha \cap \beta$. Note that $G[D]$ is *bipartite*, $D \cup X = \alpha \cup \beta$, and that nodes in $V \setminus (D \cup X)$ form an *independent set*.

3 Sequential and Batch Reconfiguration Schedules

Our first result of this section is based on the observation that any k (monotone) sequential reconfiguration steps can be replaced by two batches: the first adds all β-nodes that appear in this sequence, and the second removes all α-nodes that appear in it. In the following theorem, we use a similar (but more careful) manipulation of schedules to *compress* a long sequential schedule into a short batch schedule, while bounding the overhead in approximation. Its tradeoff is asymptotically tight (cf. Theorem 10).

▶ **Theorem 1** (Schedule Compression). *Let $G = (V, E)$ be a graph with two vertex covers α, β, and let $\mathcal{S}$ be a monotone sequential schedule from α to β that is an (η, c)-approximation, for a real $\eta \geq 1$ and an integer $c \geq 0$. For every $\varepsilon \in (0, 1)$, $\mathcal{S}$ can be transformed into a monotone $(2\lceil 1/\varepsilon \rceil)$-batch schedule $\mathcal{S}'$ that is an $(\eta + \varepsilon, c + 1)$-approximation.*

Proof. If $|\beta \setminus \alpha| = 0$, the claim immediately follows by removing all vertices in $|\alpha \setminus \beta|$ in a single batch. Otherwise, note that the schedule $\mathcal{S}$ only reconfigures vertices in $\alpha \oplus \beta$ and only adds/removes each vertex once. Let $\mathcal{S}_\alpha = (u_i)_{i=1}^{|\alpha\setminus\beta|}$ and $\mathcal{S}_\beta = (v_i)_{i=1}^{|\beta\setminus\alpha|}$ be schedule $\mathcal{S}$ restricted to vertices of α and β, respectively, without changing their order (formally, $\mathcal{S}_\alpha$ and $\mathcal{S}_\beta$ consist of the vertices in the singleton batches of $\mathcal{S}$). Define three positive integers

$$s = \lceil \varepsilon |\beta \setminus \alpha| \rceil, \; r = \lfloor \eta M \rfloor + c - |\alpha| + s, \text{and } \ell = \lceil (\max\{|\beta \setminus \alpha| - r, 0\})/s \rceil.$$

The **schedule** $\mathcal{S}'$ consists of $2\ell + 2$ batches $\mathcal{E}_0, \mathcal{E}_1, \ldots, \mathcal{E}_{2\ell+1}$, where in the first batch we add the first r vertices from $\beta \setminus \alpha$, according to $\mathcal{S}_\beta$, then in every subsequent pair of batches we remove s vertices from $\alpha \setminus \beta$, then add s vertices from $\beta \setminus \alpha$, according to $\mathcal{S}_\alpha$ and $\mathcal{S}_\beta$. In the final batch $\mathcal{E}_{2\ell+1}$, we remove the remaining vertices, if any, from $\alpha \setminus \beta$. We introduce notation to define the schedule formally. Given $i < j$, denote by $[u_i, u_j]$ and $[v_i, v_j]$ the sets $\{u_i, u_{i+1}, \ldots, u_j\}$ and $\{v_i, v_{i+1}, \ldots, v_j\}$, respectively, with the convention $[u_i, u_j] = \emptyset$ if $i > |\alpha \setminus \beta|$ (resp. $[v_i, v_j] = \emptyset$ if $i > |\beta \setminus \alpha|$), and $[u_i, u_j] = [u_i, u_{|\alpha\setminus\beta|}]$ if $j > |\alpha \setminus \beta|$ (resp. $[u_i, u_j] = [u_i, u_{|\beta\setminus\alpha|}]$ if $j > |\beta \setminus \alpha|$). Define $\mathcal{E}_0 = [v_1, v_r]$, $\mathcal{E}_{2i-1} = \left[u_{(i-1)s+1}, u_{is}\right]$, $\mathcal{E}_{2i} = \left[v_{r+(i-1)s+1}, v_{r+is}\right]$, for $i = 1, 2, \ldots, \ell$, and $\mathcal{E}_{2\ell+1} = \left[u_{\ell s+1}, u_{|\alpha\setminus\beta|}\right]$.

The schedule $\mathcal{S}'$ is monotone, as each node appears in $\mathcal{S}'$ the same amount of times it appears in $\mathcal{S}$. We next show that we add all vertices of $\beta \setminus \alpha$ in $\mathcal{S}'$. In batch $\mathcal{E}_0$ we add r vertices of $\beta \setminus \alpha$, and if $\ell > 1$ (which doesn't happen only if $r \geq |\beta \setminus \alpha|$), in batches $\mathcal{E}_2, \mathcal{E}_4 \ldots, \mathcal{E}_{2\ell}$ we add s vertices per batch. In total, these are $r + \ell \cdot s \geq |\beta \setminus \alpha|$ vertices.

With $r \geq s$ and by the definition of ℓ, we upper-bound the number of batches of $\mathcal{S}'$ by either $2 \leq 2\lceil 1/\varepsilon \rceil$ (if $\ell = 1$) or by $2\ell + 2 = 2\left\lceil \frac{|\beta\setminus\alpha| - r}{s} \right\rceil + 2 \overset{r \geq s}{\leq} 2\left\lceil \frac{|\beta\setminus\alpha|}{s} \right\rceil \overset{s = \lceil \varepsilon|\beta\setminus\alpha| \rceil}{\leq} 2\left\lceil \frac{1}{\varepsilon} \right\rceil$.

Approximation. After the first batch, the vertex cover is of size $|\alpha| + r = \lfloor \eta M \rfloor + c + s \leq (\eta + \varepsilon)M + c + 1$. Afterwards, as long as we remove s α-nodes and then add s β-nodes in each batch, the size of the vertex cover never goes above $(\eta + \varepsilon)M + c + 1$. If we add fewer than s β-nodes but remove s α-nodes, the size of the vertex cover decreases. If we, in some batch, remove less than s α-nodes, then we have removed all vertices in $\alpha \setminus \beta$, and the output is a subset of β, that is, we trivially satisfy the approximation guarantee.

Validity. Lastly, we show that all edges are covered after each reconfiguration step and that each batch is an independent set. Assume, for contradiction, that i is an index such that, in schedule $\mathcal{S}'$, node $u_i \in \mathcal{S}_\alpha$ is removed and it has a neighbor $v_j \in \mathcal{S}_\beta$ that has not yet been added to the vertex cover in $\mathcal{S}'$ (in particular, v_j could be in the same batch with u_i). This implies that $j \geq r + i - (s-1)$, and that in $\mathcal{S}$, right after the addition of the j-th β-node, the i-th α-node has not been removed. Thus, j nodes have been added, and at most $i-1$ nodes have been removed, i.e., the size of the vertex cover is

$$|\alpha| + j - (i-1) \geq |\alpha| + r + i - s + 1 - i + 1 = |\alpha| + r - s + 2 = \lfloor \eta M \rfloor + c + 2 > \eta M + c\ .$$

This is a contradiction to the assumption that $\mathcal{S}$ is an (η, c)-approximation schedule. Hence, for a node $v \in \mathcal{E}_{i'} \cap \alpha$, all of its neighbors must be in $\mathcal{E}_{j'}$'s with $j' < i'$, implying that every set in $\mathcal{S}'$ is indeed a vertex cover. This, in particular, shows that for every $u \in \mathcal{E}_{i'} \cap \alpha$ and $v \in \mathcal{E}_{i'} \cap \beta$, it holds that $(u, v) \notin E$. Together with the fact that $\mathcal{E}_{i'} \cap \alpha \setminus \beta$ and $\mathcal{E}_{i'} \cap \beta \setminus \alpha$ are independent sets, this shows that every $\mathcal{E}_{i'}$ is an independent set, for all $i' \in [\ell]$. ◀

To apply Theorem 1, we need sequential schedules. The following theorem summarizes some results for planar and cactus graphs that are either known or follow from known results [25, 10]. The result for planar graphs further extends to other graph classes that have small separators, e.g., *fixed minor-free* graphs.

▶ **Theorem 8** (Sequential Schedules). *Let G be a graph with vertex covers α, β. If G is a cactus graph, it admits a monotone $(1, 2)$-approximation schedule. If G is a planar graph, it admits a monotone $(1, O(\sqrt{M}))$-approximation schedule, where $M = \max\{|\alpha|, |\beta|\}$.*

We give a very simple unified proof of the results in Theorem 8 to Section 3.1. Here, we continue with implications of the theorem. It applies not only to the mentioned classes, but also to graphs G for which $G[\alpha \oplus \beta]$ belongs to those classes.

▶ **Observation 9.** *Let $G = (V, E)$ be either a chordal graph, an even-hole-free graph or a claw-free graph with vertex covers α and β; then $G[\alpha \oplus \beta]$ is a cactus graph.*

Proof. Let $D = \alpha \oplus \beta$. Note that $\alpha \setminus \beta$ and $\beta \setminus \alpha$ are independent sets (as a complement of a vertex cover), and so, $G[D]$ is bipartite and does not contain odd cycles.

If G is even-hole free (which includes chordal graphs), then $G[D]$ contains neither even nor odd (as mentioned above) cycles, and therefore it is a forest.

Next, assume that G is a claw-free graph. If $G[D]$ has a node v with at least 3 neighbors u_1, u_2, u_3, then there is no edge between u_1, u_2, u_3 (no odd cycles), which contradicts to G being claw-free. Therefore, every node in D has a degree at most 2, and this shows that each connected component is either a path or a cycle. ◀

We thus get the following batch scheduling theorem.

▶ **Theorem 2.** *Let $\varepsilon > 0$. Let $G = (V, E)$ be a graph with a pair α, β of vertex covers. If G belongs to one of the graph classes* {cactus, chordal, even-hole-free, claw-free}, *there is a monotone $(2\lceil 1/\varepsilon \rceil)$-batch $(1 + \varepsilon, 3)$-approximation schedule from α to β. If G is planar, there is a monotone $(2\lceil 2/\varepsilon \rceil)$-batch $(1 + \varepsilon, O(\varepsilon^{-1}))$-approximation schedule from α to β.*

Proof. By Observation 9, for every graph in a class mentioned in the first claim, $G[\alpha \oplus \beta]$ is a cactus graph, which by Theorem 8, has a monotone $(1, 2)$-approximation schedule, and by Theorem 1, has a $(2\lceil 1/\varepsilon \rceil)$-batch $(1 + \varepsilon, 3)$-approximation schedule. For the second claim, let $M = \max\{|\alpha|, |\beta|\}$ and $\delta = \varepsilon/2$. By Theorem 8, there is a monotone schedule with a worst-case vertex cover of size $M + O(\sqrt{M}) = (1+\delta)M + \sqrt{M}(O(1) - \delta\sqrt{M}/2)$. If $M \geq c/\delta^2$, for a large enough constant c, then the additive term is negative. Otherwise, it is $O(1/\delta)$. Thus, we have a sequential $(1+\delta, O(1/\delta))$-approximation schedule. Theorem 1 with $\varepsilon = \delta$ then gives a $(2\lceil 2/\varepsilon \rceil)$-batch $(1+\varepsilon, O(1/\varepsilon))$-approximation schedule, as claimed. ◀

Theorems 1 and 2 are nearly optimal, in the sense that a general compression theorem cannot prove a better trade-off between approximation overhead and schedule length. This holds even for path graphs.

▶ **Theorem 10** (Batch Lower Bound). *For any $1 \leq t < n$, there are two vertex covers α, β of a $2n$-vertex path graph such that every $(t+1)$-batch reconfiguration schedule from α to β is no better than a $\left(1 + \frac{1}{t}, 0\right)$-approximation.*

Proof. Consider a path graph on $2n$ vertices, represented as a bipartite graph $G = (A \cup B, E)$ with $|A| = |B| = n$. Observe that both A and B are minimal vertex covers in G. Consider any $(t+1)$-batch reconfiguration schedule $\mathcal{S}$ with batches $\mathcal{E}_0, \mathcal{E}_1, \ldots, \mathcal{E}_t$ from A to B. It is easy to translate $\mathcal{S}$ into a refined schedule with the same approximation ratio and at most $2t$ batches, $\mathcal{E}'_0, \mathcal{E}'_1, \ldots, \mathcal{E}'_{2t}$, such that for each i, $\mathcal{E}'_i \subseteq A$ or $\mathcal{E}'_i \subseteq B$. Let $\mathcal{E}'_{i_1}, \ldots, \mathcal{E}'_{i_t}$ be the batches that are contained in B. Since $\bigcup_{j=1}^{t} \mathcal{E}'_{i_j} = B$, there is an index j, such that $|\mathcal{E}'_{i_j}| \geq |B|/t = n/t$. Note that before adding $\mathcal{E}'_{i_j}$ to the current vertex cover, the vertex cover had size at least n (the size of a minimum vertex cover), hence after adding it, we have a vertex cover of size at least $n(1+1/t)$, which proves the claim. ◀

3.1 Separable Graphs (Proof of Theorem 8)

We construct reconfiguration schedules for classes with small separators with an additive approximation that depends on the size of the separators. The same results also follow from the bound of [10] in terms of the *pathwidth* (via a known connection between separators and pathwidth, see e.g., [2, Theorems 20, 21]). For the particular class of cactus graphs tighter results can be obtained: such graphs have $(1,2)$-approximation schedules [25]. We give a simple and unified proof for both results.

The centerpiece of our proof is the following lemma about graphs with certain "well-behaved" separation of vertices. It will also be used for distributed computation of schedules (cf. Lemma 16).

▶ **Lemma 11.** *Let $G = (V_1 \dot\cup V_2, E)$ be a graph with vertex covers α, β, such that there is no edge between $V_1 \cap \alpha$ and V_2. Let $\alpha_i = V_i \cap \alpha$, $\beta_i = V_i \cap \beta$. Assume that $|\beta_1| - |\alpha_1| \leq |\beta_2| - |\alpha_2| + 1$. If, for some $\eta \in [1,2]$, $k \geq 0$, $\mathcal{S}_1$ and $\mathcal{S}_2$ are (η, k)-approximation schedules for G_1, α_1, β_1 and G_2, α_2, β_2 (resp.), then their concatenation $\mathcal{S}_1, \mathcal{S}_2$ is an (η, k)-approximation schedule for G.*

Proof. First, observe that $\mathcal{S}_1, \mathcal{S}_2$ is indeed a valid schedule, since, as assumed, there is no α-node in V_1 that depends on a β-node in V_2. Let $a_i = |\alpha_i|$, $b_i = |\beta_i|$. Consider the size of the vertex cover during the reconfiguration of G_1. By the assumption, it is at most $\eta \max(a_1, b_1) + k + a_2 = \eta \max(|\alpha|, |\alpha| + b_1 - a_1) + k$, using $|\alpha| = a_1 + a_2$, $\eta \geq 1$. If $b_1 - a_1 \leq 0$, the size is at most $\eta|\alpha| + k$, as required. Otherwise, we have $b_1 - a_1 \geq 1$, so $\eta \max(a_1, b_1) + a_2 + k \leq \eta(b_1 + a_2) + k \leq \eta(|\beta| + a_2 - b_2) + k \leq \eta|\beta| + k$, where we used $|\beta| = b_1 + b_2$, and $b_2 - a_2 \geq b_1 - a_1 - 1 \geq 0$. Next, suppose we continue with reconfiguring G_2. If we run this schedule backwards, we get a β-to-α reconfiguration schedule, starting with G_2 (cf. Observation 7). Renaming $\beta \leftrightarrow \alpha$, $\beta_i \leftrightarrow \alpha_i$, $\mathcal{S}_i \leftrightarrow \mathit{reverse}(\mathcal{S}_{3-i})$, $G_1 \leftrightarrow G_2$, the lemma conditions still apply, and the same analysis above shows that the size of the vertex cover is below $\eta \max(|\alpha|, |\beta|) + k$ when processing G_2 as well. ◀

The proof of Theorem 8 is based on repeated decompositions of graphs and recursive computation of schedules, and leverages the presence of small separators in planar and cactus graphs.

In the discussion below, we will only consider *monotone* sequential schedules, therefore, we assume that we are given a graph G with vertex covers α and β, $\alpha \cap \beta = \emptyset$. The latter implies that every edge e in G has one α-vertex and one β-vertex.

We will consider hereditary graph classes that have small separators. Recall that a graph class is *hereditary* if it is closed under taking induced subgraphs.

Let $f : \mathbb{N} \to \mathbb{R}_+$ be a positive non-decreasing function. A graph class $\mathcal{F}$ is f*-separable* if there is a constant $n_0 > 0$ such that for every $n > n_0$ and an n-node graph $G = (V, E) \in \mathcal{F}$, there is a subset $V' \subseteq V$ of vertices of size $|V'| \leq f(n)$ such that $G - V'$ consists of two subgraphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, and there is no edge in G between V_1 and V_2, and $\max(|V_1|, |V_2|) \leq 2n/3$. We call V' a *separator.*

Our proofs follow by a inductive application of Lemma 11. First, we give a simple generalization of [25, Thm. 2].

▶ **Theorem 12.** *Let $\mathcal{G}$ be a hereditary graph class, and $\eta \in [1, 2]$ and $k \geq 1$ be reals. Assume that for every 2-connected graph $G \in \mathcal{G}$ and vertex covers α, β in G, there is an (η, k)-approximation reconfiguration schedule. Then, such schedules exist for every graph in $\mathcal{G}$ with any pair of vertex covers.*

Proof. We prove the claim by induction on the graph size, the base cases being either graphs with at most 2 vertices, or 2-connected graphs in $\mathcal{G}$. Graphs with at most 2 vertices clearly have $(1, 1)$-approximation schedules, while 2-connected graphs have (η, k)-approximation schedules, by assumption. Let G be any graph in $\mathcal{G}$ with at least 3 vertices and with a vertex v such that $G - v$ is disconnected (i.e., G is not 2-connected). Let $G_1 = (V_1, E_1), G_2 = (V_2, E_2)$ be two subgraphs (each with at least one vertex) of $G - v$ with no edge between them. Assume, without loss of generality, that $|V_1 \cap \beta| - |V_1 \cap \alpha| \leq |V_2 \cap \beta| - |V_2 \cap \alpha|$. Let $V_1' = V_1 \cup \{v\}$ and $V_2' = V_2$, if $v \in \beta$, and $V_1' = V_1$ and $V_2' = V_2 \cup \{v\}$, otherwise. Clearly, $|V_1' \cap \beta| - |V_1' \cap \alpha| \leq |V_2' \cap \beta| - |V_2' \cap \alpha| + 1$. Since $|V_1'|, |V_2'| < |V|$, by the inductive assumption, there are (η, k)-approximation schedules $\mathcal{S}_1$ and $\mathcal{S}_2$ for $G[V_1'], V_1' \cap \alpha, V_1' \cap \beta$, and $G[V_2'], V_2' \cap \alpha, V_2' \cap \beta$, respectively; thus, Lemma 11 applies, proving the induction, i.e., that there is an (η, k)-approximation schedule for G, α, β. ◀

▶ **Theorem** (Cactus part of Theorem 8)**.** *Let G be a cactus graph with vertex covers α and β. There is a monotone sequential $(1, 2)$-approximation schedule for α, β.*

Proof. Theorem 12 implies that every cactus graph admits monotone $(1, 2)$-approximation schedules, and every forest admits $(1, 1)$-approximation schedules. The theorem asserts that it suffices to limit ourselves to 2-connected instances in each class. In forests, there are no 2-connected instances with more than 2 vertices, so we are done. For cacti, it is well-known that every 2-connected cactus is a cycle graph [16], and it is easy to find a $(1, 2)$-approximation schedule for any cycle: add any β-vertex, to reduce the question to a path, which by the remark above, has a $(1, 1)$-approximation schedule. ◀

What follows is another corollary of Lemma 11 that will imply the planar part of Theorem 8.

▶ **Theorem 13.** *Let $\mathcal{F}$ be a hereditary f-separable class, for a non-decreasing function f. For every $n \in \mathbb{N}$, let $T(n)$ be the minimum value such that for every n-vertex graph $H \in \mathcal{F}$, there is a $(1, T(n))$-approximation schedule for any two disjoint vertex covers in H. Then there is a constant $n_0 > 0$ such that for every $n > n_0$, $T(n) \leq T(\lfloor 2n/3 \rfloor) + f(n)$. In particular, $T(n) \leq c + f(n) \log_{3/2} n$ holds for a constant $c > 0$.*

Proof. Let $G \in \mathcal{F}$ be an n-vertex graph with vertex covers α, β, for which there is no $(1, T(n) - 1)$-approximation schedule. They exist by the minimality of $T(n)$. Let V' be a separator of size $|V'| \leq f(n)$ and $G_1 = (V_1, E_1), G_2 = (V_2, E_2)$ be the two disconnected subgraphs of $G - V'$ with $\max(|V_1|, |V_2|) \leq 2n/3$ (V' exists since $\mathcal{F}$ is a f-separating family). Since $\mathcal{F}$ is hereditary, we have $G_1, G_2 \in \mathcal{F}$. By minimality, T is a non-decreasing function. These observations imply that each of G_1, G_2 has a $(1, T(\lfloor 2n/3 \rfloor))$-approximation schedule. Let $\mathcal{S}_i$ be the batch sequence of such a schedule for G_i, $i = 1, 2$. By Lemma 11, either $\mathcal{S}_1, \mathcal{S}_2$ or $\mathcal{S}_2, \mathcal{S}_1$ (depending on the counts of α and β vertices) is a $(1, T(\lfloor 2n/3 \rfloor))$-approximation schedule for $G_1 \cup G_2$ (note that in this case, $V_s = \emptyset$). Assume, without loss of generality, it is the former. It follows then that $\mathcal{S}_1, V' \cap \beta, \mathcal{S}_2, V' \cap \alpha$ is a $(1, T(\lfloor 2n/3 \rfloor) + |V'|)$-approximation schedule, which implies that $T(n) \leq T(\lfloor 2n/3 \rfloor) + f(n)$, since $|V'| \leq f(n)$. The claim $T(n) \leq c + f(n) \log_{3/2} n$ follows from this recursion, since it ends within $\log_{3/2} n$ iterations, and $f(n)$ is a non-decreasing function. ◀

The most prominent class to which the theorem above applies is the class of planar graphs. The celebrated Planar Separator Theorem [32, 12] shows that planar graphs are $2\sqrt{n}$-separable.

▶ **Theorem** (Planar part of Theorem 8). *Let G be a planar graph with vertex covers α and β. Let $M = \max\{|\alpha|, |\beta|\}$. There is a monotone sequential $(1, O(\sqrt{M}))$-approximation schedule for α, β.*

Proof. We only consider monotone schedules, so we can assume $\alpha \cap \beta = \emptyset$. The Planar Separator Theorem and Theorem 13 imply that for n-vertex planar graphs, the additive approximation satisfies $T(n) \leq T(\lfloor 2n/3 \rfloor) + 2\sqrt{n}$, for n larger than a constant n_0. We have, therefore, $T(n) \leq O(1) + 2\sqrt{n} \cdot \sum_{i=0}^{\infty} (2/3)^{i/2} = O(1) + 11\sqrt{n}$. We apply the scheduling to $G[\alpha \oplus \beta]$, which is planar and has at most $2M$ vertices. We get a $(1, T(2M)) = (1, O(\sqrt{M}))$-approximation. ◀

Separator theorems are also known for a number of other graph classes. Graphs of genus g are $O(g\sqrt{n})$-separable [18], graphs with a forbidden minor with h vertices are $O(h\sqrt{n})$-separable [29], graphs with treewidth T are T-separable [2, Thm. 19], k-nearest neighbor graphs in $\mathbb{R}^2$ are $\sqrt{kn}$-separable [35]. In the theorem above, we merely select a representative class.

4 Distributed Computation of Schedules

The main objective of this section is to show the following result.

▶ **Theorem 4** (Cactus Reconfiguration). *For each $\varepsilon > 0$, there is a* LOCAL *algorithm that for any connected n-node graph G and vertex covers α and β such that $G[\alpha \oplus \beta]$ is a cactus graph, computes an $O(1/\varepsilon)$-batch $(1 + \varepsilon, 3)$-approximation schedule in $O(\log^*(n)/\varepsilon)$ rounds.*

Due to Observation 9, Theorem 4 can be applied to a broader class of graphs than cacti.

▶ **Corollary 14.** *For each $\varepsilon > 0$, there is a deterministic* LOCAL *algorithm with round complexity $O(\log^*(n)/\varepsilon)$ that for any connected n-node graph $G = (V, E)$ with vertex covers α and β, computes an $O(1/\varepsilon)$-batch $(1 + \varepsilon, 3)$-approximation schedule, if G belongs to one of the following graph classes* {cycles, trees, forests, chordal, cactus, even-hole-free, claw-free}.

Proof. To apply Theorem 4, it suffices to notice, as we did in Theorem 2, that for each graph class above, $G[\alpha \oplus \beta]$ is a cactus graph. ◀

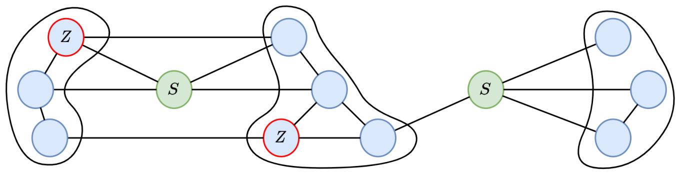

Figure 1 (d,x)-decomposition with $d=x=2$ and with $k=3$ clusters (one disconnected). Every node is either clustered or belongs to S, and every edge is either in a cluster or intersects $S\cup Z$.

Theorem 2 shows that even-hole-free graphs have a $(\lceil 1/\varepsilon\rceil+1)$-batch $(1+\varepsilon,3)$-approximation schedule, and by Theorem 10 this schedule is almost optimal. Corollary 14 gets almost the same schedule as in Theorem 2, with the same approximation factor and a length of the same order in $O(\log^*(n)/\varepsilon)$ rounds.

To efficiently compute a reconfiguration schedule in a distributed setting, we want to deal with large parts of the graph at the same time, e.g., by decomposing the graph into independent/non-interfering clusters. Thus, the central tool of this section is the notion of a *small separator decompositions* which allows simultaneous reconfiguration of "distant clusters" of the graph. In Section 4.1, we define these decompositions and show that they are sufficient to compute *good* reconfiguration schedules, both in terms of the schedule length and approximation factor, as well as the computation time. In Sections 4.2 and 4.3, we show that such decompositions can be computed efficiently, i.e., in $O(\log^*(n)/\varepsilon)$ rounds, on several graph classes, such as *cactus* and *even-hole-free* graphs. In fact, it is sufficient if $G[\alpha\oplus\beta]$ falls into the respective graph class. Then in Section 4.4 we combine these results to prove Theorem 4. Additionally, Section 4.5 contains a brief description of the ideas behind Theorem 5. Several further result on the distributed computation of schedules can be found in the full version of the paper. In particular, it contains lower bounds and our results for planar graphs. The small separator decomposition is also used in the full version to compute reconfiguration schedules for planar and outerplanar graphs.

4.1 Small Separator Decompositions

In this section, we introduce a purely graph-theoretic small separator decomposition (see Figure 1 for an illustration), give additional intuition and use it to quickly compute batch reconfiguration schedules with a distributed algorithm. Algorithms to compute such decompositions for cacti are deferred to Section 4.3, and those for general graphs to the full version.

Let $U\subseteq V$ be a subset of the vertices of a given graph $G=(V,E)$. The *weak diameter* of U is the maximum distance measured in G between any two vertices of U. The *strong diameter* (or simply diameter) is the diameter of the graph $G[U]$, that is, it is the maximum distance between any two vertices of U measured in the graph $G[U]$.

▶ **Definition 15** (Small Separator Decomposition). *Let $G=(V,E)$ be a graph with a subset $Z\subseteq V$ of* ghost nodes*. A collection of* clusters $C_1,\ldots,C_k\subseteq V$ *and a* separator set $S\subseteq V$ *is a* (weak) (d,x)-separator decomposition *with regard to Z if the following hold:*

1. *partition: $C_1,\ldots,C_k,S$ forms a partition of V,*
2. *the weak diameter of each C_i (i.e.* $\max\{dist_G(u,v)|u,v\in C_i\}$*) is upper bounded by d,*
3. *small separator set: $|S|\le x$,*
4. *S separates: There is* no *edge $\{u,v\}\in E$ with $u\in C_i\setminus Z$, $v\in C_j\setminus Z$, and $i\neq j$.*

When additionally $G[C_i]$ has diameter at most d, for all $1 \leq i \leq k$, we speak of a strong (d, x)-separator decomposition. *A d*-diameter clustering *$\mathcal{C} = \{C_i\}_{i=1}^{k}$ of a graph $G = (V, E)$ is a partitioning $V = \bigcup_{i=1}^{k} C_i$ of the vertex set into k disjoint subsets, for a parameter $k \geq 1$, such that for every $1 \leq i \leq k$, $G[C_i]$ is connected and has strong diameter at most d.*

The separating property of the set S in a (d, x)-separator decomposition can be grasped easily by setting $Z = \emptyset$. In this case, the clusters and the separator set partition all vertices of the graph such that there is no edge between any two clusters, that is, any path between two clusters has to go through S. If Z is nonempty, at least one endpoint of any inter-cluster edge has to be in Z. We emphasize that the clusters in a (d, x)-separator decomposition can be disconnected. If we want to compute a (d, x)-separator decomposition in the distributed setting we require that each cluster is equipped with a unique ID and a leader node that does not have to be contained in the cluster. Each node of the cluster needs to know the cluster ID and a path of length at most d to the cluster leader.

To prove Theorem 4, we show how to compute a small separator decomposition with parameters $d = O(1/\varepsilon)$, $x = \varepsilon|\alpha \oplus \beta|$ and $k = \max\{1, 2\varepsilon M\}$ (where $M = \max\{\alpha, \beta\}$) on graphs G for which $G[\alpha \oplus \beta]$ is a cactus graph. We stress that the usual approaches for computing (small diameter) decompositions, such as [1, 30, 39, 17], neither include any vertex-separators nor give any promise on the number of clusters. In simultaneous and independent work, a similar but different decomposition has been used to compute $(1+\varepsilon)$-approximations of the minimum vertex cover problem in the CONGEST model, deterministically and in $\text{poly}(\varepsilon^{-1}, \log n)$ rounds [13]. In the full version, we show how to use these methods to compute decompositions for general graphs in $\text{poly}\log n$ rounds, deterministically. Using the results about batch schedules for bounded arboricity graphs, this implies that we can also compute such schedules in the distributed setting (see the full version). A downside is that those techniques inherently require $\Omega(\log n)$ rounds and are not useful for proving Theorem 4. Therefore, new methods for computing our decomposition are needed to prove the following result.

Given a graph G and vertex covers α and β, we would like to compute a monotone schedule for each cluster independently. To do so without interference, the separator set S is needed. To save on the size of S, we observe that nodes in $V \setminus (\alpha \oplus \beta)$ never change, and so, can be ignored and are referred to as *ghost* nodes. Next, we show that such a decomposition is helpful to compute batch reconfiguration schedules.

▶ **Lemma 16.** *Let $G = (V, E)$ be a graph with two vertex covers α and β and a (d, x)-separator decomposition $C_1, \ldots, C_k, S$ with ghost nodes $Z \subseteq V \setminus (\alpha \oplus \beta)$. If each $G[C_i]$ admits a monotone ℓ-batch (η, c)-approximation schedule $\mathcal{S}_i$, then G admits a $(2\ell + 2)$-batch $(\eta, x + k \cdot c)$-approximation schedule $\mathcal{S}$.*

Proof. Let $I_\alpha = \{i : |C_i \cap \alpha| > |C_i \cap \beta|\}$ be the indices of clusters C_i with more α-nodes than β-nodes, and let I_β be the indices of the rest (for which $|C_i \cap \alpha| \leq |C_i \cap \beta|$). Consider the monotone schedule $\mathcal{S}_\alpha$ for $G_\alpha = G[\cup_{I_\alpha} C_i]$ that consists of a parallel execution of $\mathcal{S}_i$, for all $i \in I_\alpha$: the t-th batch of $\mathcal{S}_\alpha$ is the union of t-th batches of $\mathcal{S}_i$, $i \in I_\alpha$. Observe that $\mathcal{S}_\alpha$ is a (η, kc)-approximation for G_α. Validity follows from the fact that all edges between different C_i are adjacent to ghost nodes $Z \subseteq V \setminus (\alpha \oplus \beta)$, which are never touched by the *monotone* schedule $\mathcal{S}_\alpha$. For the approximation, note that at each step t, within each C_i, the current vertex cover is bounded by $\eta|C_i \cap \alpha| + c$, by the definition of $\mathcal{S}_i$ and I_α, hence the size of the current vertex cover in G_α at step t is at most $\eta \cdot \sum_{I_\alpha} |\alpha \cap C_i| + k \cdot c \leq \eta|\alpha| + kc$. By a symmetric construction, we get a monotone (η, kc)-approximation schedule $\mathcal{S}_\beta$ for $G_\beta = G[\cup_{I_\beta} C_i]$. By Lemma 11, $\mathcal{S}_\alpha, \mathcal{S}_\beta$ is a (η, kc)-approximation schedule for $G_\alpha \cup G_\beta = G \setminus S$ (again, we may ignore the edges between C_i, since they are covered by ghost nodes). The schedule for G is then $\mathcal{S} = (S \cap (\beta \setminus \alpha)), \mathcal{S}_\alpha, \mathcal{S}_\beta, (S \cap (\alpha \setminus \beta))$, where we handle the separator set S in a naïve way. By the properties of $\mathcal{S}_\alpha$ and $\mathcal{S}_\beta$ and by Lemma 11, $\mathcal{S}$ is a $(2\ell+2)$-batch $(\eta, kc+x)$-approximation schedule. ◀

Given a (d, x)-separator decomposition with k clusters, all phases in the proof of Lemma 16 can be executed in $O(d)$ rounds in the LOCAL model, and we obtain the following theorem.

▶ **Theorem 17.** *If the conditions of Lemma 16 hold, then a $(2\ell+2)$-batch $(\eta, x+k\cdot c)$-approximation schedule for G can be computed in $O(d)$ rounds in the* LOCAL *model.*

There are three parameters at play when using small separator decomposition to compute schedules: x – the separator set size, k – the number of clusters (both affect the approximation factor of the schedule), and d – the cluster diameter (which affects the time it takes to compute the schedule). In the next section, we show how to balance those three parameters on a graph G for which $G[\alpha \oplus \beta]$ is a cactus graph. In addition, recall that if $G[\alpha \oplus \beta]$ is a cactus graph, our batch compression results from Theorem 2 imply the existence of good monotone schedules for any subgraph, as needed by Lemma 16 and Theorem 17.

In the proof of Lemma 16 we reconfigure clusters of the given small separator decomposition in a specific order. The remark below explains why a naïve usage of small separator decompositions that does reconfigure all clusters at the same time can be hurtful for the approximation guarantee.

▶ Remark 18. Assume, for a given small separator decomposition, we can promise some good approximation on each cluster $C \in \mathcal{C}$, e.g., an (η, c)-approximation. Note that the approximation guarantee is only with respect to the maximum between $C_\alpha = C \cap \alpha$ and $C_\beta = C \cap \beta$. Thus, simultaneously reconfiguring a cluster C with $|C_\alpha| \gg |C_\beta|$ and a cluster C' with $|C'_\alpha| \ll |C'_\beta|$ at the same time might result in a much worse overall approximation.

For illustration, consider a graph composed of the disjoint union of two stars S_1 and S_2 of the same size (i.e., $K_{1,t}$ for some integer t) with α set to be the center of S_1 and the leaves of S_2 and β defined as its complement. This graph immediately has a 4-batch $(1,1)$-approximation, which adds $\beta \cap S_2$, removes $\alpha \cap S_2$, adds $\beta \cap S_1$ and then removes $\alpha \cap S_1$. On the other hand, see that a $(1,1)$-approximation on S_1 (resp. S_2) must add all of S_1 (resp. S_2) to the vertex cover before removing a single node from it. Therefore, applying the schedule of the stars simultaneously might result in a trivial $(2,0)$-approximation algorithm.

4.2 Cluster Merging

In the following Lemma, we show that given a partition of the graph into connected clusters, we can quickly merge some clusters in the partition to ensure that each cluster either has diameter bounded from below or consists of a whole connected component of the graph. Its proof is inspired by the well-known GHS and GKP algorithms for the distributed computation of a minimum-weight spanning tree [14, 15]. We use this subroutine in several parts of the paper; in particular, we use it at the end of this section to transform a small separator decomposition of $G[\alpha \oplus \beta]$ with potentially many distinct clusters into a small separator decomposition of all of G that only has few clusters (cf. Lemma 20), which is essential to keep the cost low when applying Lemma 16. In Section 4.3, we then use these results to compute suitable decompositions of cactus graphs.

We emphasize that the clusters in a d-diameter clustering need to be connected and that every vertex of the graph is contained in some cluster.

▶ **Lemma 19** (Cluster Merging). *Let $G = (V, E)$ be a graph with a d-diameter clustering $\mathcal{C} = (C_i)_{i \in k}$. For every $\varepsilon \in (0, 1)$, there is an $O(\log^*(n)/\varepsilon + d)$-round* LOCAL *algorithm that using $\mathcal{C}$, computes a $(O(1/\varepsilon) + d)$-diameter clustering $\mathcal{C}'$, where additionally, each cluster either has diameter at least $1/\varepsilon$ or consists of a whole connected component of G.*

Proof. We describe the algorithm that merges clusters from $\mathcal{C}$ to obtain $\mathcal{C}'$. To this end, assume that initially, $\mathcal{C}' = (C_i)_{i=1}^{k}$ is identical to $\mathcal{C}$. For simplicity of exposition, we consider the *cluster-graph* G' with the set $\mathcal{C}'$ of vertices, where two clusters are *adjacent* if any of their corresponding vertices are adjacent in G. We describe the algorithm using G', and explain below how this can be implemented in G. The algorithm consists of two stages, the first consists of $O(\log(1/\varepsilon))$ iterations, while the second happens only once.

Stage 1. This stage consists of iterations $t = 0, 1, 2, \ldots, \lceil \log_2(1/\varepsilon) \rceil$. Next, we describe an iteration t. A cluster C_i is *small* (in iteration t) if it has diameter at most 2^t and otherwise it is *large*. A small cluster is *isolated* if it is not adjacent to any other small cluster (it could still have edges to large clusters).

We update the clustering $\mathcal{C}'$ by merging several small clusters together. To this end, every small cluster C'_i (thought of as a node in G') picks an arbitrary edge $\{v, u\} \in E(G')$ connecting it to another small cluster C'_j, if such an edge exists. Let F be the set of selected edges. The edges in F are oriented as follows: if an edge $\{C'_i, C'_j\}$ was picked only by C'_i, it becomes an out-edge of C'_i, i.e., oriented as (C'_i, C'_j), otherwise (if C'_i and C'_j both picked $\{C'_i, C'_j\}$) it is oriented arbitrarily. This gives us a pseudoforest $\vec{F}$ (in G'), i.e., a directed subgraph with maximum outdegree 1. We compute a maximal independent set I in $\vec{F}$ with the algorithm from [9].

We update the clustering $\mathcal{C}'$, as follows: every non-isolated small cluster that is not in I picks an arbitrary adjacent cluster in I and merges with it. That is, given a cluster $C'_i \in I$, let $C'_{i_1}, \ldots, C'_{i_l}$ be the clusters that merge with C'_i. We remove C'_i and $C'_{i_1}, \ldots, C'_{i_l}$ from $\mathcal{C}'$ and add a new cluster $C' = C'_i \cup C'_{i_1} \cup \cdots \cup C'_{i_l}$. This completes the description of one iteration of Stage 1.

Stage 2. Every cluster with diameter smaller than $1/\varepsilon$ picks an arbitrary adjacent cluster, if there is any, and merges with it.

This completes the description of the algorithm.

Diameter bounds. Towards the rest of the analysis, let us first note that after each iteration t of Stage 1, due to the maximality of the independent set I, if a small cluster was not merged, then all of its neighbors in G' must have diameter greater than 2^t. Therefore, after the last iteration ($t = \lceil \log(1/\varepsilon) \rceil$) of Stage 1, every cluster with diameter smaller than $1/\varepsilon$ (i.e., a small cluster) that is not a whole connected component of G, is adjacent to a cluster of diameter at least $1/\varepsilon$, and hence is merged with such a cluster in Stage 2. Thus, after Stage 2, every cluster is either an entire connected component of G or has diameter at least $1/\varepsilon$.

Next, let us see that the diameter of a cluster does not increase abruptly within a single step of the algorithm. In iteration t of Stage 1, only clusters with diameter at most 2^t participate in a merge, and whenever a cluster merges with a set of other clusters they together induce a subgraph of G' of diameter at most 2 (which implies that the longest chain of such clusters is of length 3). Thus, after iteration t, every new cluster has diameter at most $3 \cdot 2^t + 2$, that is, 2^t for every cluster in any chain of merged clusters and the additive 2 for the edges connecting the clusters in the chain. Therefore, after Stage 1, every cluster has diameter at most $\max\{d, 3 \cdot \lceil 1/\varepsilon \rceil + 2\} = d + O(1/\varepsilon)$. In Stage 2, every cluster of diameter smaller than $1/\varepsilon$ that is not a separate component of G merges into another cluster, hence, after this stage, the maximum diameter of a cluster is increased by at most $2 + 2/\varepsilon$. This implies that the maximum diameter of a cluster is still at most $d + O(1/\varepsilon)$.

Runtime and implementation in G. First, we can use a preprocessing stage (before Stage 1) of $O(d)$ rounds, to let each cluster elect a leader and compute its diameter. Note that in iteration t of Stage 1, all clusters that participated have diameter bounded by 2^t and thus one round of an algorithm on the cluster-graph in iteration t can be simulated in $O(2^t)$ rounds in G. Computing a maximal independent set with the algorithm from [9] uses $O(\log^* \eta)$ *cluster-graph rounds*, where η is the maximum ID of a node in $\vec{F}$ (here, $\eta = \text{poly}(|V|)$, as each C_i can assume the ID of its leader). Therefore, the first stage has runtime

$$\sum_{t=1}^{\lceil \log(1/\varepsilon) \rceil} O(\log^*(n) \cdot 2^t) = O\left(\frac{\log^* n}{\varepsilon}\right).$$

Similarly, Stage 2 takes $O(1/\varepsilon)$ rounds in G. ◀

The value of ε in Lemma 19 can either be a constant or chosen subconstant and can depend on n without violating the correctness of the claim.

Next, we use Lemma 19 to show that any small separator decomposition of $G[\alpha \oplus \beta]$ can be transferred into a small separator decomposition of G with few clusters ($O(\varepsilon M)$ with $M = \max\{|\alpha|, |\beta|\}$), where the ghost node set can be chosen in order to apply Theorem 17.

▶ **Lemma 20.** *For any $\varepsilon > 0$, $d \geq 1$, there is a deterministic* LOCAL *algorithm that, given a graph $G = (V, E)$ with two vertex covers α and β and a (d, x)-separator decomposition of $G[\alpha \oplus \beta]$ with an empty ghost node set, computes a $(d + O(1/\varepsilon), x)$-separator decomposition of G with a ghost node set $Z = V \setminus (\alpha \oplus \beta)$ and $\max\{1, 2\varepsilon M\}$ clusters in $O(\log^*(n)/\varepsilon + d)$ rounds.*

We emphasize that Lemma 20 can deal with disconnected input clusters as long as their weak diameter is small, which is aligned with the definition of a (d, x)-separator decomposition. As such, the output clusters of Lemma 20 are not necessarily connected (i.e. each cluster might be formed of several disconnected components).

Proof of Lemma 20. Let $D = \alpha \oplus \beta$, $X = \alpha \cap \beta$ and let $\mathcal{C} = \{C_1, \dots, C_l\}$ be the (d, x)-separator decomposition of $G[D]$ with separator set $S \subseteq D$, where each connected component of a cluster of the input separator decomposition is its own cluster. Note that $D \cup X = \alpha \cup \beta$. We say that $\mathcal{C}$ is the **first clustering**.

The **second clustering** is formed by the clusters in $\mathcal{C}$ and a new cluster for every vertex that is not contained in $C_1 \cup \dots \cup C_l$. Each such new cluster consists of a single node. We obtain a d-diameter clustering of G (with the same d as in the first clustering).

Third clustering. Apply Lemma 19 to the second clustering and obtain a $(d + O(1/\varepsilon))$-diameter clustering $\mathcal{C}'$ of G. Let us show that the number of obtained clusters is at most $\max\{1, 2\varepsilon M\}$. Indeed, assuming $\mathcal{C}'$ consists of at least 2 components (otherwise there is nothing to prove), Lemma 19 implies that, since G is connected, each cluster has diameter at least $1/\varepsilon$. Since every vertex cover of a path contains at least half of its vertices, the diameter $1/\varepsilon$ implies that every vertex cover of G must have at least $1/2\varepsilon$ nodes in every cluster, and so, we conclude that there are at most $2\varepsilon M$ clusters.

Fourth clustering. Now, notice that each cluster C' of $\mathcal{C}'$ consists of S-nodes, clusters C_i (from the clustering $\mathcal{C}$), and Z-nodes, where we defined $Z = V \setminus D$. We remove from each C' its S-nodes, and claim that this yields a $(d + O(1/\varepsilon), x)$-separator decomposition with a ghost node set Z and at most $\lceil 2\varepsilon M \rceil$ clusters.

1. Before removing the S-nodes, each cluster C' had a strong diameter of $O(1/\varepsilon)$. After the removal, it might become disconnected, but must still have a weak diameter of $O(1/\varepsilon)$.
2. By the definition of S (in the first clustering), we have $|S| \leq x$.
3. From the definition of a d-diameter clustering, before removing the S-nodes, $\mathcal{C}'$ was a partition of V. Hence, S together with the new clusters again form a partition of V.
4. S separates: Note that every component C_i of $\mathcal{C}$ is contained in a distinct cluster C' of $\mathcal{C}'$, and for every C', $C' \setminus Z$ is composed only of such C_i components. Therefore, together with the fact that S separates $\{C_i\}_i$, this shows that S separates $\{C'_j \setminus Z\}_j$. ◀

4.3 Computing Small Separator Decompositions for Cactus Graphs

In this section we show that we can efficiently compute small separator decompositions, if $G[\alpha \oplus \beta]$ is a cactus graph. We prove the following lemma.

▶ **Lemma 21** (Cactus-Core Decomposition). *For each $\varepsilon > 0$, there is a* LOCAL *algorithm that for any connected n-node graph $G = (V, E)$ and vertex covers α, β such that $G[\alpha \oplus \beta]$ is a cactus graph, computes a (d, x)-separator decomposition with $d = O(1/\varepsilon)$, $x = \varepsilon|\alpha \oplus \beta|$, a ghost node set $Z = V \setminus (\alpha \oplus \beta)$ and $\max\{1, 2\varepsilon M\}$ clusters in $O(\log^*(n)/\varepsilon)$ rounds.*

We begin with the following observation on cactus graphs that is crucial for bounding the number of separators in our algorithm. Recall that a graph H is a minor of a graph G if H can be obtained from G by removing edges, vertices, or contracting edges (merging their endpoints).

▶ **Observation 22** (Cactus graphs). *1. The family of cactus graphs is closed under taking minors. 2. Every connected cactus graph on n vertices contains at most $3n/2$ edges.*

Proof. For the first claim, observe that if after removing vertices or edges, or contracting edges, an edge belongs to at least two cycles (i.e., the obtained graph is not cactus), then it will belong to at least two cycles after reverting the above operations too. To bound the number of edges, observe that all cactus graphs are simple, and since each edge belongs to at most 1 cycle, one can remove an edge from each cycle to obtain a tree, hence the number of edges is $m = n - 1 + c$, where c is the number of cycles. Also, each cycle contains at least 3 edges, and every edge belongs to at most one cycle, hence, $c \leq m/3$. Thus, $m - n + 1 \leq m/3$, which implies that $m < 3n/2$. ◀

In the proof of the following statement we use the cluster merging procedures developed in Section 4.2 to compute a decomposition of the graph with small diameter clusters; then we add one vertex of each inter-cluster edge to the separator set and use Observation 22 to show that the separator set is small.

▶ **Lemma 23** (Small separator set, many clusters). *For any $\varepsilon > 0$ and for any n-node connected cactus graph $G = (V, E)$ with a ghost set $Z = \emptyset$, there exists a (d, x)-separator decomposition with $d = O(1/\varepsilon)$ and $x = \varepsilon n$, and with the additional property that the strong diameter of each cluster of the decomposition is at most d. Furthermore, such a decomposition can be found in $O(\log^*(n)/\varepsilon)$ rounds in* LOCAL.

Proof. Apply Lemma 19 on the 0-diameter initial clustering of G, with every cluster consisting of a single node $v \in V$. This yields a clustering $\mathcal{C}' = \{C'_1, \dots, C'_k\}$ with each cluster having a strong diameter $O(1/\varepsilon)$, in $O(\log^*(n)/\varepsilon)$ rounds. If $k = 1$, we let $S = \emptyset$ to get the desired decomposition. On the other hand, if $k \geq 2$, recall that Lemma 19 gives in this case clusters of diameter at least $1/\varepsilon$ (as G is connected), and so, the total number of clusters is bounded by $k \leq \varepsilon n$.

It follows from the definition of a d-diameter clustering and the assumption that G is connected that each cluster induces a connected subgraph of G. Let us now bound the number of inter-cluster edges I (i.e., edges with endpoints in different clusters) for $k \geq 2$. First, observe that there can be at most 2 inter-cluster edges connecting any two given clusters, since otherwise, using the fact that each cluster induces a connected subgraph, we can find two cycles that share an edge, which contradicts the assumption that G is a cactus graph. Let us form the *cluster-graph* $H = (\mathcal{C}', E')$, where two vertices $C'_i, C'_j \in \mathcal{C}'$ are adjacent if and only if there is an inter-cluster edge in G with one endpoint in C'_i and the other in C'_j. The observation above implies that an edge in E' can correspond to at most 2 inter-cluster edges, i.e., $|E'| \geq |I|/2$. On the other hand, since each cluster C'_i induces a connected subgraph of G, observe that H is a simple minor of G. Recall from Observation 22 that cactus graphs are closed under taking minors, hence H is a cactus graph, which implies by Observation 22 that the number of its edges can be bounded by

$$|\mathcal{C}'| - 1 + |M(H)| \leq \frac{3}{2}|\mathcal{C}'| = \frac{3}{2}k,$$

where $M(H)$ is a maximum matching of H. Putting all together, thus the number of inter-cluster edges is at most $|I| \leq 2|E'| \leq 3k$.

To construct S, for each inter-cluster edge, arbitrarily add one of its two endpoints to S and remove that node from its cluster, while the other endpoint remains in its cluster. These removals might disconnect clusters in $\mathcal{C}'$, creating a new clustering $\mathcal{C}$ with (possibly) more clusters; however, this process does not introduce new edges between clusters. Thus S separates the clusters $\mathcal{C}$ and we get that $|S|$ is exactly the number of inter-cluster edges, hence, at most $3k \leq 3\varepsilon n$. The clusters of $\mathcal{C}$ along with S provide the desired decomposition. Since ε was arbitrary, we get the claim of the lemma. ◀

To obtain Lemma 21 from the small separator decomposition provided by Lemma 23, we use Lemma 20 to reduce the number of clusters.

Proof of Lemma 21. Let $D = \alpha \oplus \beta$. By our assumption, $G[D]$ is a cactus graph with components $G_1, \dots, G_k$. We first create a small separator decomposition of $G[D]$, which we later extend to a decomposition of the whole graph.

Apply the cactus graph decomposition in Lemma 23 to each component G_i to obtain a clustering $\{C_{ij}\}_{j=1,\dots,k_i}$ and a separator set S_i, for each such G_i. We use those to form a clustering $\mathcal{C} = \{C_{ij}\}_{j=1,\dots,k_i, i=1,\dots,k}$ and a separator set $S = \bigcup_{i=1}^{k} S_i$ of $G[D]$. Note that by Lemma 23, each C_{ij} is a connected cactus graph with diameter $O(1/\varepsilon)$ and the size of S is bounded by $\sum_{i=1}^{k} \varepsilon|G_i| = \varepsilon|D|$. The obtained clustering is a (d,x)-separator decomposition of $G[D]$ with an empty ghost node set, separator set S, and parameters $d = O(1/\varepsilon)$ and $x = \varepsilon|D|$. Now, apply Lemma 20 to obtain a (d,x)-separator decomposition $\mathcal{C}' = \{C'_1, \dots, C'_{k'}\}$ of G with ghost node set $Z = V \setminus (\alpha \oplus \beta)$, separator set S and parameters $d = O(1/\varepsilon)$, $x = \varepsilon|D|$ and $k' = \max\{1, 2\varepsilon M\}$ (recall that $M = \max\{|\alpha|, |\beta|\}$).

For the runtime, see that applying Lemma 23 to each component takes $O(\log^*(n)/\varepsilon)$ rounds and Lemma 20 takes $O(\log^*(n)/\varepsilon + d) = O(\log^*(n)/\varepsilon)$ rounds. Thus, the algorithm consists of $O(\log^*(n)/\varepsilon)$ rounds in total. ◀

4.4 Proof of Theorem 4

Proof of Theorem 4. Let $D = \alpha \oplus \beta$. By our assumption, $G[D]$ is a cactus graph. Use Lemma 21 to compute a (d, x)-separator decomposition $\mathcal{C} = \{C_1, \ldots, C_k\}$ of G with ghost node set $Z = V \setminus (\alpha \oplus \beta)$, separator set S and parameters $d = O(1/\varepsilon)$, $x = \varepsilon|D|$ and $k = \max\{1, 2\varepsilon M\}$ (recall that $M = \max\{|\alpha|, |\beta|\}$).

Now, apply Theorem 17 to the clustering $\mathcal{C}$ to compute a reconfiguration schedule: For each cluster $C \in \mathcal{C}$, $G[D \cap C]$ is a cactus graph, hence, from Theorem 2, $G[C \cap D]$ (and thus also $G[C]$) has an $O(1/\varepsilon)$-batch $(1+\varepsilon, 3)$-approximation schedule. Thus, we can apply Theorem 17 on the clustering $\mathcal{C}$ and obtain an $O(1/\varepsilon)$-batch $(1+\varepsilon, \varepsilon|D|+k\cdot 3)$-approximation schedule, that is, in total we can upper bound the size by

$$(1+\varepsilon)M + \varepsilon|D| + k\cdot 3 \overset{|D|\le 2M}{\le} (1+\varepsilon)M + \varepsilon\cdot 2M + \max\{1, 2\varepsilon M\}\cdot 3 \le (1+9\varepsilon)M + 3,$$

showing that the schedule is a $(1+9\varepsilon, 3)$-approximation. Rescaling ε provides the result.

For the runtime, applying Lemma 21 takes $O(\log^*(n)/\varepsilon)$ rounds. Theorem 17 takes $O(d) = O(1/\varepsilon)$ rounds. Thus, the algorithm consists of $O(\log^*(n)/\varepsilon)$ rounds in total. ◀

4.5 Distributed Computation of Greedy Schedules

We conclude with a brief description of the ideas behind Theorem 5. We want to distributively simulate the following greedy algorithm from Theorem 3: process α-nodes in an increasing order of degree (in $G[\alpha \oplus \beta]$), and for each of them, say v, add all β-neighbors of v to the vertex cover, then remove v. A naïve idea is, for rounds $i = 1$ to Δ (max. degree), add the β-neighbors of all α-nodes of degree i to the vertex cover (in parallel), then remove those α-nodes. This does not work, e.g., if all nodes have the same degree.

To fix this, we need to process α-nodes in small groups (of size $\varepsilon \max\{|\alpha|, |\beta|\}$), but still in the degree order. This is achieved by computing an $O(1/\varepsilon)$-diameter clustering, $C_1, \ldots, C_k$, with Lemma 19, and in each cluster C, letting an ε-fraction of α-nodes of given degree i, $A^C_{i,j} \subseteq C$, be processed (for $j = 1, \ldots, O(1/\varepsilon)$ iterations), which effectively means an ε fraction of *all* degree i nodes are processed in parallel (namely, $\bigcup_{\text{clusters } C} A^C_{i,j}$). Overall, we get an $O(\Delta/\varepsilon)$-batch schedule with the desired approximation (essentially, using the proof of Theorem 3). To reduce the schedule length to $O(\lambda/\varepsilon^2)$, we use the fact that there is only a small fraction of nodes with degree much larger than the arboricity, and those nodes can be processed in a single batch. More formally, we first add to the vertex cover the set β' of all β-vertices of degree larger than λ/ε (they form an $\approx \varepsilon$ fraction of all), then using the bounded degree of vertices in $\beta \setminus \beta'$, we construct a $O(\lambda/\varepsilon^2)$-batch $(\beta \setminus \beta')$-to-α schedule using the algorithm described above, and revert it (cf. Observation 7), to finally obtain an α-to-β schedule. See the full version for details.

References

1 Baruch Awerbuch, Andrew V. Goldberg, Michael Luby, and Serge A. Plotkin. Network decomposition and locality in distributed computation. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 364–369, 1989.

2 Hans L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theor. Comput. Sci.*, 209(1-2):1–45, 1998. doi:10.1016/S0304-3975(97)00228-4.

3 Marthe Bonamy, Marc Heinrich, Takehiro Ito, Yusuke Kobayashi, Haruka Mizuta, Moritz Mühlenthaler, Akira Suzuki, and Kunihiro Wasa. Shortest reconfiguration of colorings under Kempe changes. In *37th International Symposium on Theoretical Aspects of Computer*

Science, STACS 2020, March 10-13, 2020, Montpellier, France, pages 35:1–35:14, 2020. `doi:10.4230/LIPIcs.STACS.2020.35`.

4 Marthe Bonamy, Paul Ouvrard, Mikaël Rabie, Jukka Suomela, and Jara Uitto. Distributed recoloring. In *32nd International Symposium on Distributed Computing, DISC 2018, New Orleans, LA, USA, October 15-19, 2018*, pages 12:1–12:17, 2018. `doi:10.4230/LIPIcs.DISC.2018.12`.

5 Paul S. Bonsma and Luis Cereceda. Finding paths between graph colourings: PSPACE-completeness and superpolynomial distances. *Theor. Comput. Sci.*, 410(50):5215–5226, 2009.

6 Paul S. Bonsma, Marcin Kaminski, and Marcin Wrochna. Reconfiguring independent sets in claw-free graphs. In R. Ravi and Inge Li Gørtz, editors, *Algorithm Theory - SWAT 2014 - 14th Scandinavian Symposium and Workshops, Copenhagen, Denmark, July 2-4, 2014. Proceedings*, volume 8503 of *Lecture Notes in Computer Science*, pages 86–97. Springer, 2014. `doi:10.1007/978-3-319-08404-6_8`.

7 Keren Censor-Hillel and Mikaël Rabie. Distributed reconfiguration of maximal independent sets. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, pages 135:1–135:14, 2019. `doi:10.4230/LIPIcs.ICALP.2019.135`.

8 Luis Cereceda, Jan van den Heuvel, and Matthew Johnson. Connectedness of the graph of vertex-colourings. *Discret. Math.*, 308(5-6):913–919, 2008. `doi:10.1016/j.disc.2007.07.028`.

9 Richard Cole and Uzi Vishkin. Deterministic coin tossing with applications to optimal parallel list ranking. *Inf. Control*, 70(1):32–53, July 1986. `doi:10.1016/S0019-9958(86)80023-7`.

10 Mark de Berg, Bart M. P. Jansen, and Debankur Mukherjee. Independent-set reconfiguration thresholds of hereditary graph classes. *Discret. Appl. Math.*, 250:165–182, 2018. `doi:10.1016/j.dam.2018.05.029`.

11 Erik D. Demaine, Martin L. Demaine, Eli Fox-Epstein, Duc A. Hoang, Takehiro Ito, Hirotaka Ono, Yota Otachi, Ryuhei Uehara, and Takeshi Yamada. Linear-time algorithm for sliding tokens on trees. *Theoretical Computer Science*, 600:132–142, 2015.

12 Hristo Djidjev and Shankar M. Venkatesan. Reduced constants for simple cycle graph separation. *Acta Informatica*, 34(3):231–243, 1997. `doi:10.1007/s002360050082`.

13 Salwa Faour and Fabian Kuhn. Approximating bipartite minimum vertex cover in the CONGEST model. In Quentin Bramas, Rotem Oshman, and Paolo Romano, editors, *24th International Conference on Principles of Distributed Systems, OPODIS 2020, December 14-16, 2020, Strasbourg, France (Virtual Conference)*, volume 184 of *LIPIcs*, pages 29:1–29:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. `doi:10.4230/LIPIcs.OPODIS.2020.29`.

14 R. G. Gallager, P. A. Humblet, and P. M. Spira. A distributed algorithm for minimum-weight spanning trees. *ACM Trans. Program. Lang. Syst.*, 5(1):66–77, January 1983. `doi:10.1145/357195.357200`.

15 Juan A. Garay, Shay Kutten, and David Peleg. A sublinear time distributed algorithm for minimum-weight spanning trees. *SIAM J. Comput.*, 27(1):302–316, 1998. `doi:10.1137/S0097539794261118`.

16 Dennis Geller and Bennet Manvel. Reconstruction of cacti. *Canadian Journal of Mathematics*, 21:1354–1360, 1969. `doi:10.4153/CJM-1969-149-3`.

17 Mohsen Ghaffari, Fabian Kuhn, and Yannic Maus. On the complexity of local distributed graph problems. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 784–797. ACM, 2017.

18 John R Gilbert, Joan P Hutchinson, and Robert Endre Tarjan. A separator theorem for graphs of bounded genus. *Journal of Algorithms*, 5(3):391–407, 1984. `doi:10.1016/0196-6774(84)90019-1`.

19 Robert A. Hearn and Erik 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.

20 Jan van den Heuvel. The complexity of change. In Simon R. Blackburn, Stefanie Gerke, and Mark Wildon, editors, *Surveys in Combinatorics 2013*, London Mathematical Society Lecture Note Series, pages 127–160. Cambridge University Press, 2013. doi:10.1017/CBO9781139506748.005.

21 Duc A. Hoang and Ryuhei Uehara. Sliding tokens on a cactus. In Seok-Hee Hong, editor, *27th International Symposium on Algorithms and Computation, ISAAC 2016, December 12-14, 2016, Sydney, Australia*, volume 64 of *LIPIcs*, pages 37:1–37:26. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.ISAAC.2016.37.

22 Takehiro Ito, Erik D. Demaine, Nicholas J.A. Harvey, Christos H. Papadimitriou, Martha Sideri, Ryuhei Uehara, and Yushi Uno. On the complexity of reconfiguration problems. *Theoretical Computer Science*, 412(12):1054–1065, 2011. doi:10.1016/j.tcs.2010.12.005.

23 Takehiro Ito, Naonori Kakimura, Naoyuki Kamiyama, Yusuke Kobayashi, and Yoshio Okamoto. Shortest reconfiguration of perfect matchings via alternating cycles. In *27th Annual European Symposium on Algorithms, ESA 2019, September 9-11, 2019, Munich/Garching, Germany*, pages 61:1–61:15, 2019. doi:10.4230/LIPIcs.ESA.2019.61.

24 Takehiro Ito, Marcin Kaminski, and Erik D. Demaine. Reconfiguration of list edge-colorings in a graph. *Discret. Appl. Math.*, 160(15):2199–2207, 2012. doi:10.1016/j.dam.2012.05.014.

25 Takehiro Ito, Hiroyuki Nooka, and Xiao Zhou. Reconfiguration of vertex covers in a graph. *IEICE Transactions on Information and Systems*, E99.D(3):598–606, 2016. doi:10.1587/transinf.2015FCP0010.

26 Matthew Johnson, Dieter Kratsch, Stefan Kratsch, Viresh Patel, and Daniël Paulusma. Finding shortest paths between graph colourings. *Algorithmica*, 75(2):295–321, 2016.

27 Marcin Kaminski, Paul Medvedev, and Martin Milanic. Complexity of independent set reconfigurability problems. *Theor. Comput. Sci.*, 439:9–15, 2012. doi:10.1016/j.tcs.2012.03.004.

28 Jun Kawahara, Toshiki Saitoh, and Ryo Yoshinaka. The time complexity of the token swapping problem and its parallel variants. In Sheung-Hung Poon, Md. Saidur Rahman, and Hsu-Chun Yen, editors, *WALCOM: Algorithms and Computation, 11th International Conference and Workshops, WALCOM 2017, Hsinchu, Taiwan, March 29-31, 2017, Proceedings*, volume 10167 of *Lecture Notes in Computer Science*, pages 448–459. Springer, 2017. doi:10.1007/978-3-319-53925-6_35.

29 Ken-ichi Kawarabayashi and Bruce A. Reed. A separator theorem in minor-closed classes. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 153–162. IEEE Computer Society, 2010. doi:10.1109/FOCS.2010.22.

30 N. Linial and M. Saks. Low diameter graph decompositions. *Combinatorica*, 13(4):441–454, 1993.

31 Nati Linial. Locality in distributed graph algorithms. *SIAM Journal on Computing*, 21(1):193–201, 1992.

32 Richard J. Lipton and Robert Endre Tarjan. Applications of a planar separator theorem. *SIAM J. Comput.*, 9(3):615–627, 1980. doi:10.1137/0209046.

33 Daniel Lokshtanov and Amer E. Mouawad. The complexity of independent set reconfiguration on bipartite graphs. *ACM Trans. Algorithms*, 15(1):7:1–7:19, 2019. doi:10.1145/3280825.

34 Adam Marcus, Daniel A. Spielman, and Nikhil Srivastava. Interlacing families I: bipartite Ramanujan graphs of all degrees. *Annals of Mathematics*, 182:307–325, 2015.

35 Gary L. Miller, Shang-Hua Teng, William P. Thurston, and Stephen A. Vavasis. Separators for sphere-packings and nearest neighbor graphs. *J. ACM*, 44(1):1–29, 1997. doi:10.1145/256292.256294.

36 Amer Mouawad, Naomi Nishimura, Venkatesh Raman, and Sebastian Siebertz. Vertex cover reconfiguration and beyond. *Algorithms*, 11(2):20, February 2018. doi:10.3390/a11020020.

37 Naomi Nishimura. Introduction to reconfiguration. *Algorithms*, 11(4):52, 2018. doi:10.3390/a11040052.

38 David Peleg. Distributed computing. *SIAM Monographs on Discrete Mathematics and Applications*, 5, 2000.

39 Václav Rozhon and Mohsen Ghaffari. Polylogarithmic-time deterministic network decomposition and distributed derandomization. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 350–363. ACM, 2020. `doi:10.1145/3357713.3384298`.

40 Marcin Wrochna. Reconfiguration in bounded bandwidth and tree-depth. *J. Comput. Syst. Sci.*, 93:1–10, 2018. `doi:10.1016/j.jcss.2017.11.003`.