Mending Partial Solutions with Few Changes

Darya Melnyk · Aalto University
Jukka Suomela · Aalto University
Neven Villani · Aalto University and École normale supérieure Paris-Saclay

Abstract. In this paper, we study the notion of mending, i.e. given a partial solution to a graph problem, we investigate how much effort is needed to turn it into a proper solution. For example, if we have a partial coloring of a graph, how hard is it to turn it into a proper coloring?

In prior work (SIROCCO 2022), this question was formalized and studied from the perspective of *mending radius*: if there is a hole that we need to patch, how *far* do we need to modify the solution? In this work, we investigate a complementary notion of *mending volume*: how *many* nodes need to be modified to patch a hole?

We focus on the case of locally checkable labeling problems (LCLs) in trees, and show that already in this setting there are two infinite hierarchies of problems: for infinitely many values $0 < \alpha \leq 1$, there is an LCL problem with mending volume $\Theta(n^\alpha)$, and for infinitely many values $k \geq 1$, there is an LCL problem with mending volume $\Theta(\log^k n)$. Hence the mendability of LCL problems on trees is a much more fine-grained question than what one would expect based on the mending radius alone.

We define three variants of the theme: (1) *existential* mending volume, i.e., how many nodes need to be modified, (2) *expected* mending volume, i.e., how many nodes we need to explore to find a patch if we use randomness, and (3) *deterministic* mending volume, i.e., how many nodes we need to explore if we use a deterministic algorithm. We show that all three notions are distinct from each other, and we analyze the landscape of the complexities of LCL problems for the respective models.

1 Introduction

If we have a partial solution to a graph problem, how much effort is needed to turn it into a proper solution? For example, if we have a partial coloring of a graph, how hard is it to turn it into a proper coloring? In this work we present three formalisms that capture the essence of this question; the first one is purely graph-theoretic while the other two are algorithmic:

1. **Existential mending volume:** How many labels do we need to *change* to "patch a hole" in the solution?
2. **Expected mending volume:** In expectation, how many nodes do we need to *explore* to learn enough about the input graph so that we can "patch a hole"?
3. **Deterministic mending volume:** In the worst case, how many nodes do we need to explore to learn enough about the input graph so that we can "patch a hole"?

We will define these concepts formally in Definition 4.1 and 5.3, but for now the following informal description will suffice to understand what we mean by "patching a hole". We are given a graph G, a partial solution λ for some graph problem Π, and some node v that is unlabeled in λ. We would like to find a new solution λ' such that node v is labeled in λ', and also all nodes that were already labeled in λ remain labeled in λ'. We say that λ' is a *mend* of λ at node v; we have "patched a hole" at v. Now the key complexity measure is the Hamming distance between

λ and λ', i.e., the number of nodes that we had to change. If for any G, λ, and v there is a mend λ' that is within distance $T_1(n)$ from λ, we say that the existential mending volume of Π is at most $T_1(n)$. If there is a randomized algorithm that after exploring in expectation $T_2(n)$ nodes around v can find a mend, we say that the expected mending volume is at most $T_2(n)$, and if there is a deterministic algorithm that after exploring in the worst case $T_3(n)$ nodes around v can find a mend, we say that the deterministic mending volume is at most $T_3(n)$.

1.1 Motivation

Mending volume is intimately connected with the analysis of *local search*. In particular, if the mending volume of problem Π is bounded by T, then we can start with any partial solution and walk towards a valid solution so that at each step we only need to consider modifications in which we change T labels.

Moreover, mending volume naturally captures the *reconfiguration effort* in computer systems. The system is initially in a valid state, but the physical structure of the system changes (e.g., a new component is installed), leading to an invalid state λ in which at least one component is unable to fulfill its task. We need to find a new configuration λ' in which all components again function correctly. Further, in order to minimize service disruptions, we should also ensure that λ' is as close to λ as possible.

1.2 Contributions

It is easy to come up with graph problems where mending is trivial or very hard—these are problems with existential mending volume $O(1)$ or $\Theta(n)$. The work of Panconesi and Srinivasan [20] shows that the mending volume of Δ-coloring in a graph of maximum degree $\Delta \geq 3$ is $O(\log n)$. But is the mending volume for problems of this flavor always $O(1)$, $\Theta(\log n)$, or $\Theta(n)$?

We formalize this question by considering *locally checkable labeling problems* (LCLs), as defined by Naor and Stockmeyer [19]; these are problems in which we are given a graph with some maximum degree Δ, and the task is to label the nodes with labels from some finite set Σ, subject to some local constraints. Graph coloring with $k = O(1)$ colors in a graph of maximum degree $\Delta = O(1)$ is a model example of an LCL problem.

We show that already in the case of trees, it is possible to construct two infinite hierarchies of problems: for infinitely many values $0 < \alpha \leq 1$, there is an LCL problem with mending volume $\Theta(n^\alpha)$, and for infinitely many values $k \geq 1$, there is an LCL problem with mending volume $\Theta(\log^k n)$.

This shows that there is a striking difference between existential mending volume that we study here and the *mending radius* that was defined recently in prior work [6]. In trees, the mending radius of any LCL problem is known to be $O(1)$, $\Theta(\log n)$, or $\Theta(n)$. Hence mending volume makes it possible to classify LCL problems into infinitely many classes, while mending radius only leads to three classes of problems.

We also explore the landscape of mending volume beyond the case of trees; the results are summarized in Table 1. In Section 5, we then further study the relation between existential, expected, and deterministic mending volumes. We show that there are LCL problems in which all three notions coincide, but that there are also problems that separate existential and randomized mending volumes, as well as problems that separate randomized and deterministic mending volumes. That is, in the worst-case partial solutions of some LCL problems, the most efficient mend can be well-hidden in the sense that it is hard to find by probing a graph. A summary of these results is presented in Table 2; we refer to Table 3 in Appendix B for more details on the landscape of possible mending volumes.

Table 1: An overview of the landscape of existential mending volume (∃MVol) for LCL problems on the classes of paths, trees and general graphs. Here ✓ denotes that LCL problems with this mending volume exist, × denotes that such LCL problems cannot exist, and ? denotes an open question. See Table 3 in Appendix B for the landscape of other notions of mending.

Setting	Possible mending volumes						
	$O(1)$	...	$\Theta(\log n)$	$\Theta(\log^k n)$ $k > 1$	...	$\Theta(n^\alpha)$ $0 < \alpha < 1$	$\Theta(n)$
Paths and cycles	✓	×	×	×	×	×	✓
Rooted trees	✓	×	✓	✓	×	✓	✓
Trees	✓	×	✓	✓	?	✓	✓
General graphs	✓	×	✓	✓	?	✓	✓

Table 2: An overview of the hierarchy of different measures of mending: MRad is the mending radius as defined in [6], ∃MVol is the existential mending volume, EMVol is the expected mending volume, DMVol is the deterministic mending volume, and $|\mathcal{N}_{\mathsf{MRad}}|$ is the maximum number of nodes in a neighborhood of radius MRad. In the table, $x \sim y$ indicates that x and y are asymptotically equivalent for all LCL problems, and $x \nsim y$ indicates that there is at least one LCL problem for which this does not hold. Whether ∃MVol $\sim$ EMVol holds in trees is an open question.

Any graph family	MRad	$\leq$	∃MVol	$\leq$	EMVol	$\leq$	DMVol	$\leq$	$\lvert\mathcal{N}_{\mathsf{MRad}}\rvert$
Paths and cycles	MRad	$\sim$	∃MVol	$\sim$	EMVol	$\sim$	DMVol	$\sim$	$\lvert\mathcal{N}_{\mathsf{MRad}}\rvert$
Infinite trees	MRad	$\nsim$	∃MVol	$\sim$	EMVol	$\nsim$	DMVol	$\sim$	$\lvert\mathcal{N}_{\mathsf{MRad}}\rvert$
Trees	MRad	$\nsim$	∃MVol		EMVol	$\nsim$	DMVol	$\sim$	$\lvert\mathcal{N}_{\mathsf{MRad}}\rvert$
General	MRad	$\nsim$	∃MVol	$\nsim$	EMVol	$\nsim$	DMVol	$\sim$	$\lvert\mathcal{N}_{\mathsf{MRad}}\rvert$
Reference			Sect. 4.4		Sect. A.4		Sect. 5.3.1		Sect. B.2

2 Related work

One of the first papers that make explicit use of the fact that some LCL problems have a logarithmic mending volume is by Panconesi and Srinivasan [20]. They compute a Δ-coloring of a graph by recoloring an augmenting path of length up to $O(\log n)$ whenever there is a conflict. However, their main interest is solving the problem in a distributed message passing model and they therefore mainly focus on the mending radius instead of the mending volume of this problem.

The idea of refining the radius measure into a volume measure in the study of the landscape of LCL problems can be attributed to Rosenbaum and Suomela [22], who show similarities and differences between the models. The volume complexity for LCL problems has been further refined by Grunau at al. [7]. However, the focus of these papers is only on solvability (constructing a solution from nothing) rather than mendability (editing a partial solution to the closest complete solution) of a problem. They nevertheless highlight the fact that merely looking at the radius complexity does not capture all details of what information within that radius is actually necessary, and some problems that have the same radius complexity exhibit very different volume complexities.

Mending radius. Balliu et al. [6] introduced the first formal graph-theoretic notion of mending radius. The authors show how to use mending as a tool for algorithm design and analyze the complexities of mending on paths, rooted trees and grids.

In contrast to the definition of mending radius, our definition of mending volume captures more complexity classes of problems. A concrete example of the mending volume being more

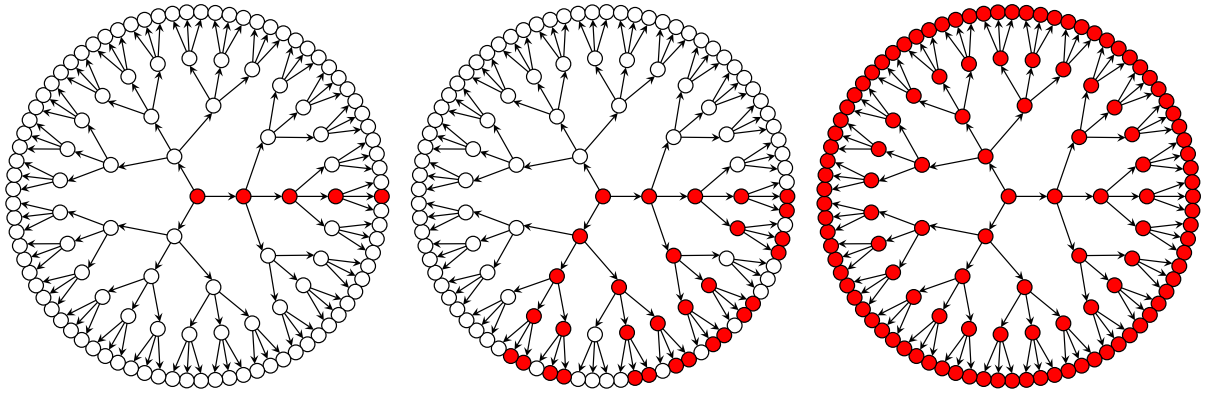


Figure 1: From left to right, solutions of R_1, R_2, R_3 (as defined in Problem 1) with the least number of red labels are visualized. In the case of R_2 (middle), each red vertex starting from the root in the center has two of its three children colored red, and this continues down to the leaves. The radius in this example is 4 and its growth rate as the graph gets larger is $\Theta(\log_3 n)$, the volume is $2^{4+1}-1$ which grows as $\Theta(n^{\log 2/\log 3})$. On each of these three solutions the set of vertices recolored red has the same radius $\Theta(\log_3 n)$, yet the volume of the red zone is $\Theta(\log_3 n)$ for R_1, $\Theta(n^{\log 2/\log 3})$ for R_2, and $\Theta(n)$ for R_3.

accurate than the mending radius is the group of three problems R_1, R_2, R_3 defined in Problem 1. Assume that we start with a partial solution where the root is uncolored and all other nodes are colored white. Naturally, any mending algorithm must color the root red, and start updating the descendants of the root down to the leaves. The mending radius of all three problems is therefore $\Theta(\log_3 n)$. The mending volume, however, differs for all three problems R_i, and the corresponding complexities are discussed in Figure 1.

Also other papers made use of mending radius, mainly as an algorithm design tool. Chechik and Mukhtar [10] design an algorithm for 6-coloring planar graphs using the observation that some small structures can be properly colored for any proper coloring of their surrounding vertices. Such structures can be removed from the graph temporarily while coloring the rest of the vertices. Similar observations have been made for computing a Δ-coloring [20] and solving an edge-orientation with maximum out-degree $(1+\varepsilon)a$ [14]. Recently, it has been shown that mending algorithms with a constant radius can also be transformed into self-stabilizing algorithms in anonymous networks [11]. On the other hand, there were also attempts to extract an explicit notion of mending, although using different definitions of partial solutions and complexity measures. This includes for example König and Wattenhofer [16], and Kutten and Peleg [18]. König and Wattenhofer [16] consider only faults that are an addition or a deletion of a single vertex or edge at a time, and hence feature only at most a constant number of unlabeled vertices. Kutten and Peleg [18] are interested in the time needed to compute a complete solution as a function of the initial number of failures.

Local search. The idea of mending volume is closely related to local search in optimization problems (in the context of traditional centralized algorithms). Often one starts with a suboptimal solution and tries to converge to a better solution from there. Usually a problem is first solved by computing some possibly random initial variable assignment that satisfies the constraints, see e.g. [8, 12, 21]. Then, a local search algorithm is applied to find a better solution in the vicinity of the previous one.

Problem 1 R_i

Input: A balanced rooted ternary tree
Labels: red and white
Task: Color the vertices so that the root is red, and every red vertex has at least i red children.

A classic application of local search in combinatorial optimization is the traveling salesman problem; local search is often applied to hard problems in order to achieve a good approximation of the optimal solution [1,2]. On the negative side, Johnson et al. [15] showed that an exponential number of iterations may be needed if the cost function can take exponential values. Ausiello and Protasi [4] later defined the class of guaranteed local optima (GLO) problems where the values of the cost function are bounded by a polynomial and showed that such problems can be solved in a polynomial number of iterations. Halldórsson [13] showed that local search can help to improve worst-case approximation guarantees by starting with a greedy solution and improving it locally using local search. He provides approximation results for various problems, such as the independent set, k-dimensional matching and k-set packing in nearly-linear sequential time. Chandra and Halldórsson [9] later showed an $2(k+1)/3$ approximation algorithm for the weighted k-set packing problem, thus improving a previous result from Bafna et al. [5], and Arkin and Hassin [3].

3 Preliminaries

Our definition of the mending volume is built along the lines of the definition of the mending radius in [6]: we define the mending volume as a measure entirely independent of any distributed computing model and we place ourselves in the context of Locally Checkable Labeling problems (LCLs) first introduced in [19]. We use the same definition of partial solutions as [6] in order to make our results comparable. A reader who is familiar with the notions of graph labeling problems—and LCLs in particular—as well as with the specific definition of partial solutions from [6] may skip directly to the next section in which we introduce a formal definition of the mending volume.

3.1 Locally checkable labelings

LCLs are labeling problems on bounded-degree graphs. In these problems, an input graph with maximum degree $\Delta = O(1)$ is given and the task is to produce an assignment of labels to vertices in a way that satisfies some predetermined local constraints. The specification of an LCL problem is done by means of a local verifier.

Definition 3.1 (Local verifier)**.** A *verifier* ϕ is a function that maps tuples (G, λ, v) to $\{\mathsf{happy}, \mathsf{unhappy}\}$, where v is a vertex and λ a labeling of G. We say that the verifier ϕ *accepts* λ if $\phi(G, \lambda, v) = \mathsf{happy}$ for all v, otherwise it *rejects* λ.

In addition, ϕ is local if, for some constant radius r, whenever (G_1, λ_1) and (G_2, λ_2) coincide over the radius-r neighborhood of v_1 and v_2 then they have the same image according to ϕ. That is, $(G_1, \lambda_1)_{|\mathcal{N}_r(v_1)} \simeq (G_2, \lambda_2)_{|\mathcal{N}_r(v_2)}$ implies $\phi(G_1, \lambda_1, v_1) = \phi(G_2, \lambda_2, v_2)$

An LCL problem is entirely characterized by a finite set of labels and a local verifier.

Definition 3.2 (Locally Checkable Labeling)**.** A *Locally Checkable Labeling* problem Π is represented by a finite set of labels Σ, a class of input graphs $\mathcal{G}$, and a local verifier ϕ. An instance of Π is a graph $G \in \mathcal{G}$. A solution is a labeling λ of G over Σ that is accepted by ϕ.

3.2 Partial solutions

Mending takes as input an incomplete labeling and extends it into one that is a little closer to being complete. Since graph labelings were defined to be complete over all vertices, the most natural way to define partial solutions is to extend the set of labels with one fresh label $\perp$ that is interpreted as "unlabeled", and adapt the local constraints to allow labelings that involve this new label. We will often refer to vertices that are labeled $\perp$ simply as "unlabeled vertices" or "holes".

A desirable definition of partial solutions should satisfy the following three properties:

1. A partial solution without any hole is a complete solution;
2. the empty labeling (the constant function $\lambda_\perp : \_ \mapsto \perp$) is a partial solution;
3. a sub-solution of a partial solution is also a partial solution. That is, if λ is a partial solution then any labeling

$$\lambda_S : x \mapsto \begin{cases} \lambda(x) & \text{if } x \in S \\ \perp & \text{otherwise} \end{cases}$$

is a partial solution.

As stated in [6], the following is a simple way to satisfy all of these constraints: extend the verifier ϕ' to be happy whenever an unlabeled vertex is visible in the radius-r neighborhood, otherwise fall back to the same rules as ϕ.

Definition 3.3 (Partial solution)**.** For $\Pi = (\Sigma, \mathcal{G}, \phi)$, where ϕ has radius r, define a relaxation $\Pi^* = (\Sigma^* \coloneqq \Sigma \sqcup \{\perp\}, \mathcal{G}, \phi^*)$ of Π to allow empty labels.

For a labeling λ' over Σ^*, define $\phi^*(G, \lambda', v)$ as follows: if there exists a node $u_\perp$ within distance r of v such that $\lambda'(u_\perp) = \perp$, then $\phi^*(G, \lambda', v) \coloneqq \mathsf{happy}$; otherwise, let λ be any labeling over Σ that agrees with λ' on $G_{|\mathcal{N}_v(r)}$ and set $\phi^*(G, \lambda', v) \coloneqq \phi(G, \lambda, v)$.

We define $\mathrm{dom}_\Sigma(\lambda')$ to be the set of vertices that λ' labels with labels from Σ. A labeling (resp. solution) of Π^* is called a *partial labeling* (resp. *partial solution*) of Π.

One can easily check that all the desirable properties stated above are satisfied by Definition 3.3; this fact is also proven in [6]. Note that this definition of partial solutions has a notion of locality that is consistent between labelings and partial labelings: the verifiers ϕ and ϕ^* have the same locality radius.

We can now define what it means to mend a partial solution: a mend of λ is a new partial solution with one specific vertex no longer labeled $\perp$, and no additional $\perp$ labels.

Definition 3.4 (Mend)**.** For a partial solution λ of Π on an instance G, we say that λ' is a *mend* of λ at $v \in G$ if the following hold:

Validity: λ' is a partial solution.

Progress: $\mathrm{dom}_\Sigma(\lambda) \cup \{v\} \subseteq \mathrm{dom}_\Sigma(\lambda')$, that is, no $\perp$ was added and v is no longer labeled $\perp$.

The mending problem $\mathsf{Mend}(\Pi)$ associated with an LCL Π is the following task: given $G \in \mathcal{G}$, λ solution of Π^* and v hole of λ, produce λ' a mend of λ at v.

4 Complexity landscape of existential mending volume

Having defined LCLs and partial solutions, we can now introduce mending volume. In this section, we consider an existential definition of the mending volume. This definition (see Section 4.1) is a purely graph-theoretic measure of the optimal solution for a worst-case instance of a mending problem. Later, in Section 4.2, we develop a technique for designing LCLs that have a specific existential mending volume on infinite rooted trees. In Section 4.3, we show that these problems can be transferred to finite and non-oriented trees while keeping the same mending volume complexity. Finally, in Sections 4.4 and 4.5, we apply these design techniques to obtain problems that have mending volume $\Theta(n^\alpha), 0 < \alpha < 1$ or $\Theta(\log^k n), k \in \mathbb{N}^*$, thereby providing examples of complexities that the mending volume exhibits that were not observed previously in the study of the mending radius.

4.1 Existential mending volume: Definition

For two labelings λ and λ', we define $\operatorname{diff}(\lambda, \lambda') \coloneqq \{v\colon\ \lambda(v) \neq \lambda'(v)\}$ such that $|\operatorname{diff}(\lambda, \lambda')|$ is the Hamming distance between two partial solutions. We define the existential mending volume of a problem Π as the distance between the partial solution and the optimal mend for the worst-case instance (G, λ, v) of $\mathsf{Mend}(\Pi)$. Here, G is an input graph from the family on which Π is defined, λ is a partial solution, and v is a hole s.t. $\lambda(v) = \perp$ at which λ must be mended.

Definition 4.1 (Existential mending volume)**.**

$$\exists\mathsf{MVol}(\Pi) \coloneqq \max_{G,\lambda,v} \min \left\{ |\operatorname{diff}(\lambda, \lambda')|\colon\ \lambda' \text{ mend of } \lambda \text{ at } v \right\}$$

4.2 Mending in infinite rooted trees

In the following sections, we will establish a landscape of possible complexities that the existential mending volume can exhibit. For a summary, please refer to Table 1 that was introduced earlier. To this end, we describe examples of LCL problems that have logarithmic, polylogarithmic and polynomial existential mending volumes. The statement of these examples is made easier by the fact that all problems we show are of a specific type and we refer to them as *propagation problems*. The complexity analysis of problems in this class is very straightforward for two reasons: (1) they admit a simple matrix description by an encoding shown in Section 4.2.2, and (2) we only need to study their behavior in infinite trees thanks to results from Section 4.3. The advantage of infinite trees is that the complexity analysis is simplified by the absence of cycles, high-degree nodes, leaves, and other irregularities of the input graph. This restriction of only considering infinite regular rooted trees also has the complementary effect of illustrating that even simple problems already exhibit a rich variety of mending volume complexities. Since any propagation problem with mending volume T can be transformed into a problem on general trees or graphs with the same mending volume T, our choice does not restrict the generality of our results.

4.2.1 Propagation problems

In this section, we define propagation problems on infinite rooted trees, with the goal to use them as a design tool for LCLs that exhibit specific mending volume complexities.

Definition 4.2 (Infinite Δ-regular rooted trees)**.** We call trees that satisfy the following properties

- there are infinitely many vertices;
- exactly one vertex is distinguished as the root;
- every vertex admits a unique directed path to the root;
- every vertex has exactly Δ incoming edges.

infinite Δ-regular rooted trees, or simply infinite rooted trees when Δ is clear from the context.

Note that this class of graphs only consists of a single graph for a fixed Δ. On this class of input graphs, we define propagation problems as any LCL problem that is constructed according to the procedure explained in Definition 4.3.

Definition 4.3 (Construction of a propagation problem)**.** On the label set Σ, distinguish two special labels—the initial label l_0, and the wildcard label $l_\_$. Let $\Sigma' \coloneqq \Sigma \setminus \{l_\_\}$. Choose some $\mu : \Sigma' \times \Sigma' \to \mathbb{N}$ and some $\Delta \geq \max_{l \in \Sigma'} \sum_{l' \in \Sigma'} \mu(l, l')$. This defines an LCL on infinite Δ-regular rooted trees, with locality 1, where the radius-1 neighborhood of v labeled by λ is accepted if all of the following constraints are satisfied:

- $\lambda(v) = l_0$ if v is the root;
- if $\lambda(v) = l \neq l_-$ then, for every $l' \in \Sigma'$, there are at least $\mu(l, l')$ children of v labeled l'.

In other words, we only allow labeling constraints of the form "any vertex labeled l must have at least $\mu(l, l')$ children labeled l'" or "the root must be labeled l_0". The requirement $\Delta \geq \max_{l \in \Sigma'} \sum_{l' \in \Sigma'} \mu(l, l')$ is chosen such that all constraints are compatible with each other. Note that there are no constraints involving the wildcard label l_- in this definition: it may appear as a child of any other label, and it may have any labels as its own children. In particular, the labeling where the root is unlabeled and all non-root vertices are labeled l_- is always a valid partial solution. We will show in Corollary 4.5 that this labeling is the worst-case instance for most propagation problems.

Since the input graphs on which these problems are defined are infinite, and since these problems often produce mends that have infinite volume, we study the volume not in terms of the total number of modified labels but in terms of the number of modified labels at distance at most $d_{\max}$ from the hole.

4.2.2 Matrix representation

Let M be a matrix of size $|\Sigma'| \times |\Sigma'|$ defined as $M[l, l'] = \mu(l, l')$. Observe that the coefficient $M^d[l, l']$ of the d-th power of M is the tightest lower bound on how many children labeled l' a vertex labeled l must have at distance d for a complete solution to be accepted. Indeed, by induction, a vertex labeled l must have at least $M[l, l']$ children labeled l' at distance 1; it must then have at least $\sum_{l'' \in \Sigma'} M^d[l, l'']M[l'', l'] = M^{d+1}[l, l']$ children labeled l' at distance $d+1$. We argue in Theorem 4.4 that this provides bounds for $\exists\mathsf{MVol}$.

We write $\|L_l M^d\| \coloneqq \sum_{l' \in \Sigma'} M^d[l, l']$, where L_l is the line vector with a 1 only in position l. We show in Theorem 4.4 that this quantity expresses both upper and lower bounds on the mending volume up to distance $d_{\max}$ of the propagation problem described by M.

Theorem 4.4 (Mending complexity of a propagation problem)**.** *The mending volume up to distance $d_{\max}$ of a propagation problem represented by M is between $\sum_{d=0}^{d_{\max}} \|L_{l_0} M^d\|$ and $\max_{l \in \Sigma'} \sum_{d=0}^{d_{\max}} \|L_l M^d\|$*

Proof. We start with the lower bound. Recall that in the input graph all vertices have degree exactly Δ. Consider an initial partial labeling λ in which the root is initially unlabeled, and all other vertices are labeled l_-. A mend λ' of λ at the root will have to be a complete solution, and thus require the root to be labeled l_0. By the previous observation, at distance d, there must be at least $M^p[l_0, l']$ vertices in λ' labeled l' that must have been modified during the mending. This way, $\sum_{d=0}^{d_{\max}} \|L_{l_0} M^d\|$ is a lower bound for how many labels were modified at distance at most $d_{\max}$.

We can now show the upper bound. An important characteristic of the family of propagation problems is that the output of the verifier depends only on a portion of the labels of the children. Once sufficiently many children are labeled correctly, the remaining ones have no impact. This means that no initial configuration can force more than $M^d[l, l']$ labels l' to be added at distance d from a vertex v labeled l: in the worst case, it suffices to arbitrarily choose $M[l, l']$ children at each level for each and color them accordingly while ignoring all the other children. Thus the worst-case instance has mending cost at distance $d_{\max}$ no more than $\max_{l \in \Sigma'} \sum_{d=0}^{d_{\max}} \|L_l M^d\|$ □

Corollary 4.5 (Worst-case instance of a propagation problem)**.** *If l_0 is such that $\|L_{l_0} M^d\| = \Omega(\max_{l \in \Sigma'} \|L_l M^d\|)$ then the initial instance where the root is unlabeled and all other vertices are labeled l_- is the worst-case instance.*

The condition for Corollary 4.5 is satisfied at least for problems where all labels are reachable from l_0 in the sense that for every $l' \in \Sigma'$ there exists some $d_{l'}$ for which $M^{d_{l'}}[l_0, l'] \neq 0$.

4.2.3 Landscape of the growth rate of matrix exponentiation

In this section, we turn to a study of possible growth rates of the quantity $\|L_l M^d\|$ introduced in Section 4.2.2. This quantity is bounded by $|\Sigma'| \times \max_{l' \in \Sigma'} M^d[l, l']$. In order to determine the mending volume of a propagation problem, it is sufficient to look at the growth rate as a function of d of $\max_{l,l' \in \Sigma'} M^d[l, l']$—the greatest coefficient of M^d. We will denote it as $\max M^d$.

In the following analysis, we make use of the interpretation of M as the adjacency matrix of a graph G_M. G_M is a directed graph with one vertex for each element of Σ'. For every pair $(v_l, v_{l'})$ there are exactly $M[l, l']$ directed edges from v_l to $v_{l'}$. Further, there are $M^d[l, l']$ walks of length exactly d from v_l to $v_{l'}$ in G_M. Let $c(l)$ be the number of cycles in G that contain v_l. We say that a vertex v_l is of type 0 (*resp.* 1 or 2) if $c(l) = 0$ (*resp.* $c(l) = 1$ or $c(l) \geq 2$). We will show that the type of the vertices fully determines the growth rate of $|M^p|$: if some vertex is part of several cycles, then there are exponentially many paths of length d from that vertex to itself. Otherwise, if all vertices are part of at most one cycle, then there are only polynomially many paths of length d from one vertex to another.

Lemma 4.6. *Consider a vertex v_l of type 2. It holds that $M^d[l, l] = \Omega((1 + \beta)^d)$ for some $\beta > 0$.*

Proof. Let $C_1, C_2, \dots$ be the $c(l)$ distinct cycles that contain v_l. Let $L_1, L_2, \dots$ denote their lengths respectively, and let $L \coloneqq \operatorname{lcm}(L_1, L_2, \dots)$. There are at least $c(l)$ walks of length L from v_l to itself, each following only one of the cycles C_j L/L_j number of times. Hence, for all k, there are at least $c(l)^k$ walks of length $d \coloneqq kL$ from v_l to itself and therefore $M^d[l, l] \geq c(l)^{d/L}$ walks for infinitely many values of d. Thus $M^d[l, l] = \Omega((c(l)^{1/L})^d)$. □

Corollary 4.7. *Let vertex v_l be of type 2 and reachable from v_{l_0}. Then $M^d[l_0, l] = \Omega((1 + \beta)^d)$ for some $\beta > 0$.*

Lemma 4.8. *If there is no vertex of type 2 reachable from v_{l_0} then $|M^d|_{l_0} = O(d^k)$ for some constant k.*

Proof. For each vertex v_l, we denote $C(l)$ to be the cluster of v_l, defined as follows: $C(l) \coloneqq \{l\}$ if $c(l) = 0$; otherwise, $C(l) \coloneqq \{l' \mid v_l \to^* v_{l'} \to^* v_l\}$ describes the vertices in the same cycle as v_l. Since $c(l) \leq 1$, the clusters form a partition of Σ'. We use K to denote the number of clusters.

Construct G'_M whose vertices are the clusters of G_M, by contracting each cluster into a single vertex while keeping duplicate edges between different clusters, and removing edges inside a cluster. The resulting graph G'_M is acyclic. Any walk W of length exactly d in G_M from v_l to $v_{l'}$ is uniquely defined by

- a walk W' in G'_M from $C(l)$ to $C(l')$, let $C(l) = C_1 \to C_2 \to \cdots \to C_{|W|} = C(l')$ be this walk;
- the length d_i of the walk within C_i, for each $1 \leq i \leq |W|$ (because no vertex is of type 2, there is only one such walk for a given length).

Note that $d_1 + \cdots + d_{|W|} + (|W| - 1) \leq d$ and $|W| \leq K \leq |\Sigma'|$. There are finitely many walks W' in G'_M and for each of them the number of possible tuples $(d_1, \dots, d_{|W|})$ is bounded by d^K. Thus the number of walks W of length d is polynomially bounded by $O(d^K)$. □

We observe further that if there is a walk in G'_M that goes through two or more cycles, then there are at least $\Omega(d)$ paths of length d. Whereas if G'_M contains only isolated cycles, then there are at most $O(1)$ paths of length d. Thus Theorem 4.9 holds.

Theorem 4.9 (Landscape of $\max M^d$)**.** *The growth rate of $p \mapsto \max |M^d|$ is either eventually zero, or $\Theta(1)$, or $O(d^p)$ for some value $p \geq 1$, or $\Omega((1 + \beta)^d)$ for some $\beta > 0$.*

Problem 2 Generalization of Π to finite and non-Δ-regular trees

Input: Any tree
Labels: Same as Π
Task: Any vertex of degree exactly Δ must satisfy the labeling constraints from Π

Combining these results with the known bounds from Theorem 4.4 stating how to relate the mending volume to the growth of $\max M^d$, we obtain Corollary 4.10.

Corollary 4.10 (Landscape of the mending volume on infinite rooted trees)**.** *The mending volume up to distance $d \coloneqq \log_\Delta n$ of a propagation problem is either*

- $O(1)$ *if* $\max M^d$ *is eventually zero;*
- *or* $\Theta(\log n)$ *if* $\max M^d = \Theta(1)$*;*
- *or* $O(\log^k n)$ *for some* $k > 1$ *if* $\max M^d = O(d^p)$*;*
- *or* $\Omega(n^\alpha)$ *for some* $0 < \alpha < 1$ *if* $\max M^d = \Omega((1+\beta)^d)$.

This concludes the survey of the landscape of propagation problems on infinite rooted trees. We found that there are complexity classes $O(1)$, $\mathrm{poly}(\log n)$ and $\Omega(n^\alpha)$, with a gap between $\omega(\log n)$ and $o(\log^2 n)$. In Sections 4.4 and 4.5 we will look more closely at the classes of growths $\Omega(n^\alpha)$ and $\Theta(\log^k n)$ to show that infinitely many values of α and k can appear.

4.3 Finite and non-regular trees

Some of the presented results from Section 4.2.3 are fortunately applicable to trees even if they are no longer infinite rooted and regular. Indeed there is a straightforward translation that transforms a propagation problem on infinite Δ-regular rooted trees into one that has the same growth properties, but can be defined on finite rooted trees where some vertices are of degree lower than Δ. This construction is shown in Problem 2.

We now argue that the fact that these trees are finite does not affect the conclusions made earlier about the possible complexities. The worst-case instance can namely still be constructed as a balanced finite Δ-regular tree. We prove that (1) randomness does not decrease the performance and (2) the mending process is just as efficient in the case that the tree is unbalanced as it is in a balanced tree.

Theorem 4.11 (Generalization to finite unbalanced trees does not change the mending volume complexity)**.** *If Π is a propagation problem, then its generalization Π' to unbalanced trees has the same mending volume complexity.*

Proof. Assume that we wish to label a tree of size $n+1$ rooted in v. Each of the Δ subtrees that are children of v have size $n/\Delta + d_i$ for $1 \le i \le \Delta$ where $\sum_{i=1}^{\Delta} d_i = 0$, and we wish to assign a new labeling to each of them. The growth rate f_j^{bal} $(1 \le j \le \delta)$ of the number of labels that would need to be modified for a balanced tree is uniquely determined by its assigned root label l_i. A key observation is that as f_j^{bal} is defined by some $\sum_{d=0}^{d_{\max}} \|L_l M^d\|$, it is either eventually zero or eventually convex. The total number of modified labels if the tree was balanced would be

$$f^{\mathrm{bal}}(n+1) = 1 + \sum_{i=1}^{\delta} f_i^{\mathrm{bal}}(n/\delta).$$

Assume inductively that the true number of modified labels in any non-balanced tree of size n' is less than $f_j^{\mathrm{bal}}(n')$, i.e. that a balanced tree is the worst-case input. Let $c_{i,j}$ denote this number for the subtree i if it were assigned root label j. The average performance of an

algorithm that distributes the required labels randomly among all children with equal probability is then

$$\begin{aligned}
c' &= \operatorname{avg}_{\sigma\in\mathfrak{S}_\delta} 1 + \sum_{i=1}^{\delta} c_{i,\sigma(i)} \\
&\le \operatorname{avg}_{\sigma\in\mathfrak{S}_\delta} 1 + \sum_{i=1}^{\delta} f^{\mathrm{bal}}_{\sigma(i)}(n/\delta + d_i) && \text{induction hypothesis} \\
&\le 1 + \sum_{i=1}^{\delta} \operatorname{avg}_{\sigma\in\mathfrak{S}_\delta} f^{\mathrm{bal}}_{i}(n/\delta + d_{\sigma(i)}) && \text{reassign indices} \\
&\le 1 + \sum_{i=1}^{\delta} f^{\mathrm{bal}}_{i}(n/\delta) && \text{by convexity of all } f^{\mathrm{bal}}_j \\
&\le f^{\mathrm{bal}}(n+1).
\end{aligned}$$

The induction hypothesis holds for a balanced tree of size 1. Since the optimal mend has volume at most the expected volume of a mend picked at random, it follows that the existential mending volume on unbalanced trees is the same as the existential mending volume on balanced trees. □

4.4 Application: $\exists\mathsf{MVol} = n^{\Theta(1)}$

In the following two sections, we show examples of problems that exhibit polynomial and polylogarithmic complexities, with a particular focus on showing which values of $0 < \alpha < 1$ and $k \ge 1$ can appear for complexities $\Theta(n^\alpha)$ and $\Theta(\log^k n)$.

The prior analysis resulting in Corollary 4.10 suggests that, in order to construct a problem with volume $n^{\Theta(1)}$, we should consider a propagation problem whose matrix M has exponential growth for $d \mapsto \max M^d$. A good candidate is the problem described by $M = \begin{pmatrix} 2 \end{pmatrix}$ for $\Delta = 3$. It describes the following problem:

Problem 3 Polynomial propagation

Input: An infinite rooted Δ-regular tree
Labels: red and white
Task: Color the vertices according to the following rules, by order of precedence:

- any labeling is valid for a vertex that does not have exactly 3 children;
- any labels are valid for the children of a vertex labeled white;
- if a vertex is red, it needs at least two of its children to be red;
- the root has to be red.

Using the terminology of Definition 4.3, red is the initial label, and white is the wildcard label. From the initial labeling consisting of the root being unlabeled and all other vertices being white, a mend needs to recolor 2^d vertices at layer d from white to red. For a balanced ternary tree, this will produce a total of $2^{\log_3 n+1} - 1$ recolored vertices, i.e. $\Theta(n^{\ln 2/\ln 3})$.

We further argue that by slightly adjusting the parameters, we can engineer any rational power of n: the problem described by $M = \begin{pmatrix} 2^p \end{pmatrix}$ for $\Delta = 2^q$, where $q > p > 1$, will exhibit complexity $\Theta(n^{\ln 2^p/\ln 2^q}) = \Theta(n^{p/q})$.

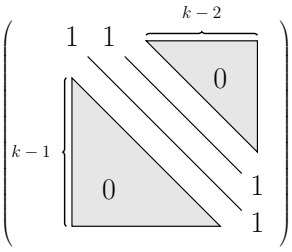


Figure 2: M_k of size $k \times k$ exhibits growth $\Theta(p^k)$ for the root label l_1

4.5 Application: $\exists\mathbf{MVol} = (\log n)^{\Theta(1)}$

We now show how to construct a problem that has mending volume $\log^k n$ for any chosen $k \geq 1$. This time, Corollary 4.10 suggests to look for a matrix for which $d \mapsto \max M^d$ has growth rate $\Theta(p^k)$. This is satisfied by the matrix illustrated in Figure 2: its size is k, and it has entries 1 along and immediately above the diagonal, and entries 0 everywhere else. A solution to the problem described by this matrix has the following form: a path from a leaf to the root is labeled l_0 including both ends. From each vertex labeled l_0 there is a path labeled l_1 from another leaf, and so on until each vertex labeled l_{k-2} being the endpoint of a path labeled l_{k-1} from a leaf.

5 Algorithmic definitions of volume mending

The mending volume introduced in Definition 4.1 and studied throughout Section 4 does not consider the complexity of finding a mend. For distributed systems, where no entity has a complete view of the input graph, a naive algorithm to compute the optimal mend may need to query an exponential number of vertices compared to how many of them will actually be relabeled in the end. For such applications, it may be more appropriate to consider alternative measures of mending where the cost is not just the number of labels that were modified but also the number of vertices that need to be queried before an algorithm with only local knowledge of the graph can compute a mend.

5.1 Mending with local knowledge

We will now focus on the process of computing the mend and make sure that each step of the computation can be completed with only local knowledge. A single-step definition like Definition 4.1 cannot express such restrictions. Therefore we will introduce a step-by-step definition of a process that computes a mend.

Such a process should take as input a graph G, an initial partial solution λ, and a hole v of λ that needs to be mended. We consider the following model describing the knowledge that the process has access to: initially it only knows $W = \{v\}$ and the list of its direct neighbors. Whenever a vertex v' that is a direct neighbor of W is queried, the process can add v' to W and it acquires knowledge of all the neighbors of v'. Thus at each step of the computation, the process has access to the connected set W of all vertices previously explored, and it can choose to explore any direct neighbor of W as its next vertex. Such an exploration model is similar to local computation algorithms discussed in [23] where we can learn the graph by probing it one node at a time.

The process can stop its exploration whenever the set of explored vertices W contains a mend in the following sense: there exists a mend of λ at v —λ'— in which only vertices from W are relabeled, i.e. such that $\{v' : \lambda'(v') \neq \lambda(v')\} \subseteq W$. We choose to model this exploration process with a Markov chain: each state represents a possible value for W. Having explored W,

the process may choose that its next vertex to explore should be v'. In this case, the Markov chain will assign a nonzero probability to the transition $W \to W \cup \{v'\}$.

Definition 5.1 (Mender)**.** For an input graph $G = (V, E)$, an initial partial labeling λ, and a hole v of λ, let $\mathcal{M}_{G,\lambda,v}$ be a Markov chain over $\mathcal{P}(V)$ the subsets of V. We call $\mathcal{M}$ a local *mender* if the following properties hold for every G, λ, v:

Progress: $\mathcal{M}_{G,\lambda,v}(W, W') > 0$ implies $W \subseteq W'$.

Termination: $\mathcal{M}_{G,\lambda,v}(W, W) = 1$ iff there exists a mend of λ at v contained in W.

Locality: $\mathcal{M}_{G,\lambda,v}(W, \cdot)$ depends only on $\mathcal{N}_1^G(W)$, i.e. W and its direct neighborhood.

Progress states that the set of explored vertices can only grow from one step to the next, and termination expresses that the computation will halt as soon as a mend is found. Locality ensures that at all steps of the computation the information accessible to the mender is exactly what it has already explored.

Observation 5.2 ($\mathcal{M}_{G,\lambda,v}$ has a stationary distribution)**.** *Observe the following:*

1. *There must exist a mend of λ at $v \in V$, so that V is absorbing. Since all other states of the Markov chain are subsets of V, the conditions of* Termination *and* Progress *from Definition 5.1 are compatible.*
2. *All non-absorbing states eventually lead to an absorbing state in finitely many steps with nonzero probability since there are only finitely many subsets of V and hence finitely many states.*
3. *There are no loops involving more than one state.*

From the above properties it follows that $\mathcal{M}_{G,\lambda,v}$ is an absorbing Markov chain, and therefore it has a stationary distribution that we denote $\mathcal{M}^*_{G,\lambda,v}$.

We call all subsets of V that contain a mend *final states*. Due to the existence of a stationary distribution, a final state W_F is reachable from the initial state W_0 if and only if $\mathcal{M}^*_{G,\lambda,v}(W_0, W_F) > 0$. We write $\mathcal{F}^{\mathcal{M}}_{G,\lambda,v}(W_0)$ for the set of final states reachable from W_0.

In Definition 5.3, we formally define the three notions of mending volume. We therefore use the formalism of Markov chains developed in Definition 5.1. These mending volume complexities are defined as measures of the sizes of the elements of $\mathcal{F}^{\mathcal{M}}_{G,\lambda,v}(W_0)$, where $W_0 = \{v\}$ the hole to be mended. For all of these measures we consider the worst-case instance by taking a max over all possible values of λ and v.

Definition 5.3 (Complexity)**.** Consider a mender $\mathcal{M}$ and a graph G. We define the best-case volume, expected volume, worst-case volume, and radius as measures of $\mathcal{F}^{\mathcal{M}}_{G,\lambda,v}(\{v\})$:

$$\exists\mathsf{MVol}'(\mathcal{M}, G) \coloneqq \max_{\lambda,v} \min\left\{|W_F| \;:\; W_F \in \mathcal{F}^{\mathcal{M}}_{G,\lambda,v}(\{v\})\right\} \qquad \text{best-case volume}$$

$$\mathsf{EMVol}(\mathcal{M}, G) \coloneqq \max_{\lambda,v} \sum_{W_F} \mathcal{M}^*_{G,\lambda,v}(\{v\}, W_F) \cdot |W_F| \qquad \text{expected volume}$$

$$\mathsf{DMVol}(\mathcal{M}, G) \coloneqq \max_{\lambda,v} \max\left\{|W_F| \;:\; W_F \in \mathcal{F}^{\mathcal{M}}_{G,\lambda,v}(\{v\})\right\} \qquad \text{worst-case volume}$$

$$\mathsf{MRad}(\mathcal{M}, G) \coloneqq \max_{\lambda,v} \min_{W_F \in \mathcal{F}^{\mathcal{M}}_{G,\lambda,v}(W)} \max_{w \in W_F} d(\{v\}, w) \qquad \text{best-case radius}$$

Intuitively, $\exists\mathsf{MVol}'$ (best-case mending volume) is the size of the smallest reachable final state, EMVol (expected mending volume) is the expected size of an explored set given by the

probability distribution of the mender, DMVol (deterministic mending volume) is the size of the largest reachable final state, and MRad (mending radius) is the smallest radius of a final state.

We have also redefined the notion of existential mending volume $\exists\mathsf{MVol}'$ using the formalism of Markov chains. As suggested by the notation, this measure coincides with $\exists\mathsf{MVol}$ introduced in Definition 4.1. We prove this fact in Lemma A.1 using a straightforward construction of a Markov chain that explores all possible mends so that the "min" in the definition of $\exists\mathsf{MVol}'$ results in the optimal mend.

The worst-case volume is written DMVol, i.e. *Deterministic Mending Volume*, because an equivalent definition of DMVol is to remove all randomness from $\mathcal{M}$ and define DMVol as the deterministic size of the final state. This connection is explored in detail in Section B.2. The definition of MRad also coincides with the mending radius introduced in [6], as is shown in Lemma A.2.

Further, for a function T on integers and a family of input graphs $\mathcal{G}$, we say that Π has $\exists\mathsf{MVol}' = O(T)$, if there exists $\mathcal{M}$ that has performance $\exists\mathsf{MVol}'(\mathcal{M}, G_n) = O(T(n))$ on all $G_n \in \mathcal{G}_n$ instances of size n. That is, there is a mender which performs at least as well as T asymptotically. Similarly, we say that Π has $\exists\mathsf{MVol}' = \Omega(T)$ if for all $\mathcal{M}$ there exists $G_n \in \mathcal{G}_n$ such that $\exists\mathsf{MVol}'(\mathcal{M}, G_n) = \Omega(T(n))$, i.e., no mender can guarantee performance asymptotically better than T on all instances. The above notation extends to all combinations of o, O, ω, Ω, Θ and $\exists\mathsf{MVol}'$, EMVol, DMVol, MRad.

Having concluded the definitions, we will next lay some foundations for a rough landscape of the existing mending complexities, extended with the new measures introduced in Definition 5.3. The first step will be to establish the hierarchy between all the measures of complexity introduced previously, and to handle some simple cases such as the setting of paths and cycles.

5.2 Hierarchy of complexities

We first establish that $\mathsf{MRad} \leq 2r \times \exists\mathsf{MVol}$, where r is the radius of the local verifier that defines the problem Π. Consider an initial labeling λ, and a mend of λ at v called λ'. Denote $D = \{v : \lambda(v) \neq \lambda'(v)\}$ the set of relabeled vertices. We argue that $\mathcal{N}_r^G(D)$ is connected. Assume the opposite, and consider a maximal subset of D denoted D' that does not contain v where $\mathcal{N}_r^G(D')$ is connected. Construct a new labeling λ'' as follows

$$\lambda'' : x \mapsto \begin{cases} \lambda(x) & \text{if } x \in D' \\ \lambda'(x) & \text{otherwise.} \end{cases}$$

Observe that λ' is a valid partial labeling: for every vertex, its neighborhood in λ'' is identical to its neighborhood either in λ or in λ'. In addition, λ'' coincides with λ' over the neighborhood of v. It is therefore a mend of λ at v. Since λ'' coincides with λ over D', it modifies fewer labels than λ'. This shows that λ' is not minimal.

Therefore if λ' is a minimal mend then the radius-r neighborhood of the vertices it relabels is connected. By definition, any mend modifies a label at distance at least MRad, therefore it must also modify at least one label every distance $2r$ on a path from v to distance at least MRad. This shows $\mathsf{MRad} \leq 2r \times \exists\mathsf{MVol}$.

Observe further that there is an easy strategy to ensure that $\mathsf{DMVol} \leq |\mathcal{N}_{\mathsf{MRad}}|$: consider a naive mender $\mathcal{M}$ that always selects the next step to be the neighborhood of the current state with probability 1, i.e. $\mathcal{M}_{G,\lambda,v}(W, \mathcal{N}_1^G(W)) = 1$. By construction, it will halt after MRad steps on the final state $\mathcal{N}_{\mathsf{MRad}}^G(\{v\})$ since by definition the radius-MRad neighborhood always contains a mend. Using the minimum, the expectation, and the maximum in Definition 5.3 leads to obvious inequalities $\exists\mathsf{MVol}' \leq \mathsf{EMVol} \leq \mathsf{DMVol}$.

These observations provide the following hierarchy of mending complexities (we ignore constant multiplicative factors here):

$$\mathsf{MRad} \leq \exists\mathsf{MVol} \leq \mathsf{EMVol} \leq \mathsf{DMVol} \leq |\mathcal{N}_{\mathsf{MRad}}| \tag{1}$$

As shown in [6], the mending radius can only exhibit complexities on trees that are $O(1)$, $\Theta(\log n)$, or $\Theta(n)$, and this is further restricted to $O(1)$ or $\Theta(n)$ on paths and cycles. Since our definition of MRad is equivalent, the same gaps hold. In the following settings, the hierarchy from Equation 1 collapses:

- if MRad is $\Theta(n)$, so are all the other greater measures;
- if MRad is $O(1)$, then $|\mathcal{N}_{\mathsf{MRad}}|$ is also $O(1)$ and also all the smaller measures.

Hence the landscape is only diverse when the MRad is logarithmic and the other measures are between $\Omega(\log n)$ and $O(n)$. In Section 5.3, we will exhibit separations between these measures on trees. They will necessarily rely on problems that have radius $\Theta(\log n)$ since no separations exist otherwise.

5.3 Separations

Equation 1 shows a chain of inequalities between the different measures of mending that we have introduced. We can complement this equation by showing that all measures we have introduced are distinct from each other. This is done by proving that well-chosen LCL problems and classes of graphs feature *separations* between these measures: on some instance, one measure of mending is arbitrarily smaller than another.

In Section 5.3.1 we show that EMVol is distinct from DMVol, and in Section 5.3.2 we show that ∃MVol is distinct from EMVol. These results are complemented by Section A in which we show that

- DMVol is distinct from $|\mathcal{N}_{\mathsf{MRad}}|$ on balanced trees (Section A.2);
- MRad is distinct from ∃MVol using propagation problems (Section A.3);
- ∃MVol is distinct from EMVol (Section A.4); in this case, the separation does not rely on a promise unlike in Section 5.3.2, but it only works on general graphs instead of trees.

5.3.1 EMVol is distinct from DMVol on trees and general graphs

Most propagation problems described in Section 4—for example the one with polynomial complexity from Section 4.4—feature a separation between EMVol and DMVol when executed on finite trees that are not necessarily balanced. Indeed when mending such a propagation problem on an unbalanced tree, the expected mending volume is the same as the existential mending volume as proved in Section 4.3. However there is no way to deterministically guarantee that the subtree explored is not a very unbalanced one, which could have size up to $\Theta(n)$. This way, we have already shown several examples of problems with expected mending volume $o(n)$ but deterministic mending volume $\Theta(n)$.

5.3.2 ∃MVol is distinct from EMVol on trees with a global promise

For this separation, we again resort to a promise. In fact we conjecture that on non-promise families of trees ∃MVol is equivalent to EMVol. The promise is as follows: we guarantee that the input tree is balanced and that there exists at least one vertex with a degree of exactly 2. The chosen problem is described in Problem 4.

A solution to this problem requires finding a specific vertex of a tree that has degree 2. Even with access to randomness, there is no algorithm that can guarantee to find such a vertex in fewer than $\Theta(n)$ queries. On the other hand, if there is a guarantee that such a vertex exists, then it is at distance $O(\log n)$ and can easily be found if one has access to a complete view of the graph. These properties make the EMVol linear, while the ∃MVol as well as the MRad are only logarithmic.

Problem 4 Degree-2 sink

Input: A balanced tree with at least one vertex of degree exactly 2
Labels: Orientations of the edges
Task: Orient the edges so that vertices have out-degree 0 only if they have degree exactly 2.

As mentioned earlier, a similar separation result between ∃MVol and EMVol is developed in Section A.4. It uses general graphs instead of trees, but it also does not need to rely on a global promise unlike the separation on trees shown here.

Acknowledgments

This work was supported in part by the Academy of Finland, Grant 333837.

References

[1] Emile Aarts and Jan Karel Lenstra, editors. *Local Search in Combinatorial Optimization*. Princeton University Press, 2003. doi:10.2307/j.ctv346t9c.

[2] Eric Angel. *A Survey of Approximation Results for Local Search Algorithms*, pages 30–73. 2006. doi:10.1007/11671541_2.

[3] Esther M. Arkin and Refael Hassin. On Local Search for Weighted k-Set Packing. *Mathematics of Operations Research*, 23(3):640–648, 1998. doi:10.1287/moor.23.3.640.

[4] Giorgio Ausiello and Marco Protasi. Local search, reducibility and approximability of NP-optimization problems. *Information Processing Letters*, 54(2):73–79, 1995. doi:10.1016/0020-0190(95)00006-X.

[5] Vineet Bafna, Babu Narayanan, and R. Ravi. Nonoverlapping local alignments (weighted independent sets of axis-parallel rectangles). *Discrete Applied Mathematics*, 71(1):41–53, 1996. doi:10.1016/S0166-218X(96)00063-7.

[6] Alkida Balliu, Juho Hirvonen, Darya Melnyk, Dennis Olivetti, Joel Rybicki, and Jukka Suomela. Local Mending. In *Structural Information and Communication Complexity*, 2022. doi:10.1007/978-3-031-09993-9_1.

[7] Sebastian Brandt, Christoph Grunau, and Václav Rozhoň. The Randomized Local Computation Complexity of the Lovász Local Lemma. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*, PODC'21, 2021. doi:10.1145/3465084.3467931.

[8] M. Chams, A. Hertz, and D. de Werra. Some experiments with simulated annealing for coloring graphs. *European Journal of Operational Research*, 32(2):260–266, 1987. Third EURO Summer Institute Special Issue Decision Making in an Uncertain World. doi:10.1016/S0377-2217(87)80148-0.

[9] Barun Chandra and Magnús M Halldórsson. Greedy Local Improvement and Weighted Set Packing Approximation. *Journal of Algorithms*, 39(2):223–240, 2001. doi:10.1006/jagm.2000.1155.

[10] Shiri Chechik and Doron Mukhtar. Optimal Distributed Coloring Algorithms for Planar Graphs in the LOCAL Model. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2019.

[11] Johanne Cohen, Laurence Pilard, Mikaël Rabie, and Jonas Sénizergues. Making Self-Stabilizing any Locally Greedy Problem, 2022. doi:10.48550/ARXIV.2208.14700.

[12] Philippe Galinier and Alain Hertz. A survey of local search methods for graph coloring. *Computers & Operations Research*, 33(9):2547–2562, 2006. Part Special Issue: Anniversary Focused Issue of Computers & Operations Research on Tabu Search. doi:10.1016/j.cor.2005.07.028.

[13] Magnús M. Halldórsson. Approximating Discrete Collections via Local Improvements. In *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '95, page 160–169, 1995. doi:10.5555/313651.313687.

[14] David G. Harris, Hsin-Hao Su, and Hoa T. Vu. On the Locality of Nash-Williams Forest Decomposition and Star-Forest Decomposition. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*, PODC'21, 2021. doi:10.1145/3465084.3467908.

[15] David S. Johnson, Christos H. Papadimitriou, and Mihalis Yannakakis. How easy is local search? *Journal of Computer and System Sciences*, 37(1):79–100, 1988. doi:10.1016/0022-0000(88)90046-3.

[16] Michael König and Roger Wattenhofer. On Local Fixing. In *Principles of Distributed Systems*, 2013. doi:10.1007/978-3-319-03850-6_14.

[17] Amos Korman, Jean-Sébastien Sereni, and Laurent Viennot. Toward more localized local algorithms: removing assumptions concerning global knowledge. *CoRR*, abs/1512.03306, 2015. arXiv:1512.03306.

[18] S. Kutten and D. Peleg. Tight fault locality. In *Proceedings of IEEE 36th Annual Foundations of Computer Science*, 1995. doi:10.1109/SFCS.1995.492672.

[19] Moni Naor and Larry Stockmeyer. What Can be Computed Locally? *SIAM Journal on Computing*, 24(6):1259–1277, 1995. doi:10.1137/S0097539793254571.

[20] Alessandro Panconesi and Aravind Srinivasan. The local nature of Δ-coloring and its algorithmic applications. *Combinatorica*, 15(2):255–280, 1995. doi:10.1007/BF01200759.

[21] Silvia Richter, Malte Helmert, and Charles Gretton. A Stochastic Local Search Approach to Vertex Cover. In *KI 2007: Advances in Artificial Intelligence*, pages 412–426, 2007. doi:10.1007/978-3-540-74565-5_31.

[22] Will Rosenbaum and Jukka Suomela. Seeing Far vs. Seeing Wide: Volume Complexity of Local Graph Problems. In *Proceedings of the 39th Symposium on Principles of Distributed Computing*, PODC '20, 2020. doi:10.1145/3382734.3405721.

[23] Ronitt Rubinfeld, Gil Tamir, Shai Vardi, and Ning Xie. Fast Local Computation Algorithms. In *2nd Symposium on Innovations in Computer Science (ICT)*, pages 223–238, 2011.

A Separations

As announced in Section 5.3 we complement Equation 1 by proving separations between the different measures of mending featured, as well as some equivalences between different definitions of the same measure.

A.1 Equivalent definitions

The study of the landscape of the different measures of mending in Section 5.2 assumes some equivalences between different definitions of the same measure. In particular we must prove that $\exists\mathsf{MVol}$ introduced in Definition 4.1 is asymptotically equivalent to $\exists\mathsf{MVol}'$ introduced in Definition 5.3. We must also show that MRad from Definition 5.3 is equivalent to the notion of mending radius defined in [6].

The usage of "min" in the definitions of $\exists\mathsf{MVol}'$ and MRad suggests that as long as a process explores sufficiently many configurations, it will have $\exists\mathsf{MVol}'$ and MRad close to the optimum. In particular, a mender that has all subsets as its reachable final states has the minimum possible size of a set that contains a mend as $\exists\mathsf{MVol}'$, and the minimum radius as MRad. Note that there exists such a mender, for example a mender that always explores one neighbor of its current explored set with equal probability.

Lemma A.1 (Equivalence of $\exists\mathsf{MVol}$ and $\exists\mathsf{MVol}'$)**.** $\exists\mathsf{MVol}$ *is asymptotically equivalent to* $\exists\mathsf{MVol}'$*.*

Proof. We can easily show $\exists\mathsf{MVol} = O(\exists\mathsf{MVol}')$: $\exists\mathsf{MVol}$ is the minimum size of a mend, while $\exists\mathsf{MVol}'$ is the minimum size of a connected set that contains a mend.

Reciprocally by the same argument as in Section 5.2, the radius-r neighborhood of the set of relabeled vertices is connected, and it thus is of greater size than the set whose size is measured by $\exists\mathsf{MVol}'$. Thus $\exists\mathsf{MVol}' \leq \Delta^r \times \exists\mathsf{MVol}$. □

Lemma A.2 (MRad is the mending radius)**.** MRad *defined in Definition 5.3 is equivalent to the mending radius defined in [6].*

Proof. Assuming that all connected subsets of V are reachable states of the mender, MRad is then by definition the minimum radius of a mend. □

A.2 DMVol is distinct from $|\mathcal{N}_{\mathsf{MRad}}|$ on balanced trees

As we will show later in Theorem B.2, the DMVol exhibits very few different complexities on trees, and it is always asymptotically of the same size as $\mathcal{N}_{\mathsf{MRad}}$. As such a separation between DMVol and $|\mathcal{N}_{\mathsf{MRad}}|$ needs to rely on a promise on the structure of the graph. A sufficient promise is that the input tree is a balanced binary tree. We will consider the problem of sinkless orientation, defined in Problem 5. Note that the verifier has radius 1, where valid configurations are those where either the vertex has at most one neighbor, or it has out-degree at least 1. A mending procedure works as follows: starting from the initial hole, explore exactly one path to any leaf. This path can be mended simply by ensuring that all of its edges are directed towards the leaf. Since any leaf is acceptable, the explored path can be computed locally in a deterministic manner, and it has size $O(\log n)$ as all leaves are at an at most logarithmic distance from the hole.

Problem 5 Sinkless orientation on balanced trees

Input: A balanced binary tree
Labels: Orientations of the edges
Task: Orient the edges so that all vertices with degree at least 2 have out-degree at least 1.

A worst-case instance is one where all edges are oriented towards the hole and labeling the hole would produce a sink. This case can require that at least the path from the hole to the nearest leaf is relabeled, which produces a mending radius of at least $\Omega(\log n)$ if all leaves are at an at least logarithmic distance from the hole.

Thus sinkless orientation on balanced trees has $\mathsf{DMVol}(\Pi) = \Theta(\log n)$ whereas $|\mathcal{N}_{\mathsf{MRad}}| = \Theta(n)$

A.3 MRad is distinct from ∃MVol on trees and general graphs

In Sections 4.5 and 4.4 we have constructed problems that exhibit ∃MVol either $\Theta(\log^k n)$ or $\Theta(n^\alpha)$ while having $\mathsf{MRad} = \Theta(\log n)$. These problems illustrate a separation between MRad and ∃MVol.

A.4 Separating EMVol from ∃MVol without a promise

We observed that EMVol is equal to ∃MVol in the case of propagation problems on both finite and infinite rooted trees. We will next show that EMVol and ∃MVol can be separated in general graphs. We will therefore construct a family of directed graphs and a simple problem that together lead to existential mending volume complexity $\exists\mathsf{MVol} = O(\log n)$ and expected mending volume complexity $\mathsf{EMVol} = \omega(\log n)$. We then show that this family of graphs can be extended to all undirected graphs while preserving the upper bound on ∃MVol. This construction will also naturally preserve the lower bound on EMVol.

A.4.1 Problem description

We first define the family of graphs $\mathcal{T} = \bigcup_{h\in\mathbb{N}} \mathcal{T}_h$ built by adding horizontal layers to a tree skeleton in the following way.

Definition A.3 ($\mathcal{T}$)**.** Start with the balanced binary rooted tree t_h that has height $h+1$ for any $h \in \mathbb{N}$. We will assign coordinates to each of the vertices, consisting of the distance from the root and the horizontal index. The root gets coordinate $(0,0)$ and the children of (i,j) are assigned coordinates $\{(i+1,2j),(i+1,2j+1)\}$.

We add an undirected edge between the sibling vertices $(i,j)-(i,j+1)$ for every $0 \le i < h$ and $0 \le j < 2^i - 1$. Additionally, consider a vertex (h,j_0) on the bottom layer for some $0 \le j_0 < 2^i$ and add a directed edge from $(h,j) \to (h,j+1)$ for every $0 \le j < j_0$ and from $(h,j) \leftarrow (h,j+1)$ for every $j_0 \le j < 2^i - 1$.

This way, the pair (h,j_0) uniquely defines one graph t_{h,j_0} with h layers and all edges on the bottom layer pointing towards the vertex at position j_0. We call the vertex (h,j_0) the *sink* of t_{h,j_0}.

The infinite family of graphs $\mathcal{T}$ is then defined as $\mathcal{T} \coloneqq \bigcup_{h\in\mathbb{N}} \mathcal{T}_h = \bigcup_{h\in\mathbb{N}} \left\{t_{h,j_0} \mid 0 \le j_0 < 2^h\right\}$.

One such graph is illustrated in Figure 3, with a height of 5 and a sink at position 4.

For this family of graphs $\mathcal{T}$, we define Problem 6. Observe that for any graph of $\mathcal{T}$, there is exactly one coloring that is complete and satisfies the labeling constraints from Problem 6.

Problem 6 Path to the sink

Input: Some $t_{h,j_0} \in \mathcal{T}$
Labels: red and black
Task: Color the vertices so that

- the root is red
- any red vertex that is not the sink has at least one red child.

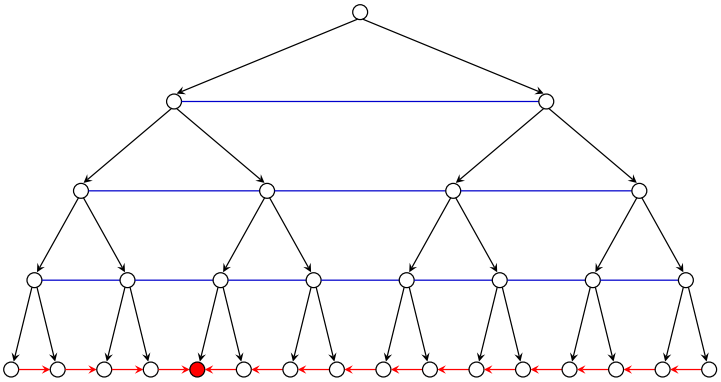


Figure 3: A visualization of a $t_{5,4}$. Elements of $\mathcal{T}$ are entirely characterized by their height (here 5) and the position of the sink (marked red at position 4). The $t_{5,4}$ features a skeleton in the form of a balanced binary oriented tree, to which horizontal unoriented links (in blue) are added. On the bottom layer, there are oriented edges (in red) towards the sink.

This case is visualized in Figure 4a for the input graph $t_{5,4}$. In this coloring, all vertices are colored white except for the unique path from the root to the sink. Note that there are many valid incomplete labelings as visualized in Figures 4b–4d. If some leaves are unlabeled, then the vertices above them can be colored red, and if the root is unlabeled then the entire graph can be colored black.

We will show in Sections A.4.2 and A.4.4 that this problem has logarithmic ∃MVol, yet its EMVol is $\omega(\log n)$. We conjecture the complexity of EMVol to even be $\Omega(\log^2 n)$. This shows that ∃MVol and EMVol are distinct from each other on oriented graphs, which is a separation that is not observed on trees and proven to not exist on rooted trees. This result extends to unoriented graphs thanks to an encoding of oriented graphs described in Section A.5.

A.4.2 Study of ∃MVol

We now show that Problem 6 has logarithmic ∃MVol. A mending algorithm that uses a complete view of the graph to compute a mend with volume $O(\log n)$ is described in Algorithm 1.

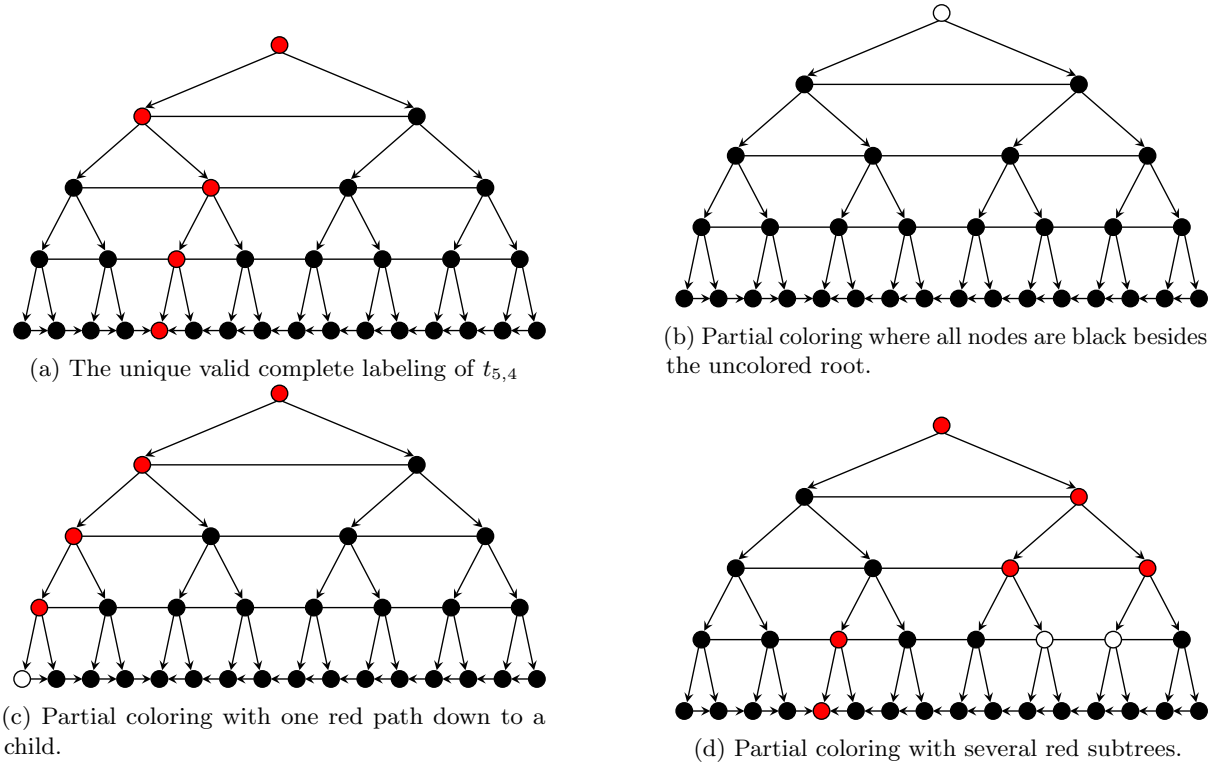
Lemma A.4 (Algorithm 1 is correct)**.** *When Algorithm 1 terminates, λ is a valid mend of the initial input labeling.*

Proof. Observe that Algorithm 1 immediately assigns a label to the hole that it mends, and it never writes a $\perp$ label. Therefore Algorithm 1 is correct if and only if it outputs a valid labeling. An invariant of Algorithm 1 is that at the beginning of any iteration of the main loop, all vertices other than current have a valid labeling.

This invariant is preserved: recoloring the current vertex black makes current correctly labeled, and can at most introduce an incompatibility in the labeling of its parent. In such a case, we immediately perform current ← parent which restores the invariant. The situations in which relabeling current does not introduce an inconsistency in the neighborhood of parent are the following: parent is already black, or parent is red and it has another red child. In these situations, the execution terminates with a valid labeling.

Phase 2 is only ever reached in the following situation: current is labeled red with two black children, the sink is one of its descendants, and all vertices other than current are correctly labeled. In this situation, relabeling a path from current to the sink to red restores all labeling constraints and results in a valid mend. □

Both phases terminate in no more than $2h = 2\log_2 n$ relabeling steps and thus the problem has ∃MVol $= O(\log n)$.



(a) The unique valid complete labeling of $t_{5,4}$

(b) Partial coloring where all nodes are black besides the uncolored root.

(c) Partial coloring with one red path down to a child.

(d) Partial coloring with several red subtrees.

Figure 4: Example of four possible colorings of $t_{5,4}$.

A.4.3 Simulating the problem in general oriented graphs

We now transfer the problem to work on general graphs without any promise on the structure of the input. As a first step, we will keep the orientation and assume that the directed edges as well as the parent-child relationship are part of the graph. We first show that the class $\mathcal{T}$ from Definition A.3 can be defined by local constraints on directed graphs. These constraints are listed in Problem 7: the graphs that satisfy all of these constraints are exactly the elements of $\mathcal{T}$.

We call a vertex *broken* if it does not satisfy all the constraints from Problem 7. A (sub)graph that has no broken vertices is called *well-formed*. Note that all graphs of $\mathcal{T}$ are well-formed. Conversely, a non-empty well-formed graph is in $\mathcal{T}$, as shown by the construction illustrated in Figure 5. Constraints 1a and 1b from Problem 7 guarantee that the graph has a binary tree structure, though it is not yet guaranteed to be balanced. Constraints 2b and 2c impose a unique way of adding edges between sibling vertices, and guarantee that the tree is balanced. The resulting structure can no longer be modified: 2a forbids any additional sibling of the root (shown in red in Figure 5d); 2a' similarly prevents internal vertices from having additional siblings (shown in yellow); finally there are no available vertices that could be siblings of the vertices on the border (shown in blue) because of constraints 2c and 2c'. Constraints 3a and 3b then impose an orientation of the edges between sibling vertices of the bottom layer once a sink is chosen.

It should be noted that a well-formed subgraph is not necessarily in $\mathcal{T}$. Any subgraph of a well-formed graph is well-formed, and there are many proper subgraphs from graphs of $\mathcal{T}$ that are not themselves in $\mathcal{T}$. However, for any well-formed subgraph $G' \subseteq G$, there exists a $t_{h',j'_0} \in \mathcal{T}$ of which G' is a subgraph.

In general graphs, we extend the problem so that graphs outside of $\mathcal{T}$ can also be labeled. In addition to the previous labeling rules from Problem 6, any labeling is valid for a vertex that is broken (see Problem 8).

Lemma A.5 (Preservation of ∃MVol on oriented graphs)**.** *Problem 8 has the same existential*

Algorithm 1 Mending procedure of Problem 6

Input
  t input graph from $\mathcal{T}$
  λ initial labeling
  v hole of λ
current $\leftarrow v$
Phase 1
loop
  λ(current) $\leftarrow$ black
  parent $\leftarrow$ parent of current
  other $\leftarrow$ other child of parent
  if λ(parent) = black **or** λ(parent) = $\perp$ **or** λ(parent) = λ(other) = red **then**
    return
  else if the sink is a descendant of parent **then**
    current $\leftarrow$ parent
    go to Phase 2
  else
    current $\leftarrow$ parent
  end if
end loop
Phase 2
Recolor red a path from current to the sink

mending volume as Problem 6, namely $\Theta(\log n)$.

Proof. The key property to the preservation of $\exists\mathsf{MVol} = O(\log n)$ is the following: for any vertex, there exists a vertex at distance $O(\log n)$ that is either a sink or broken.

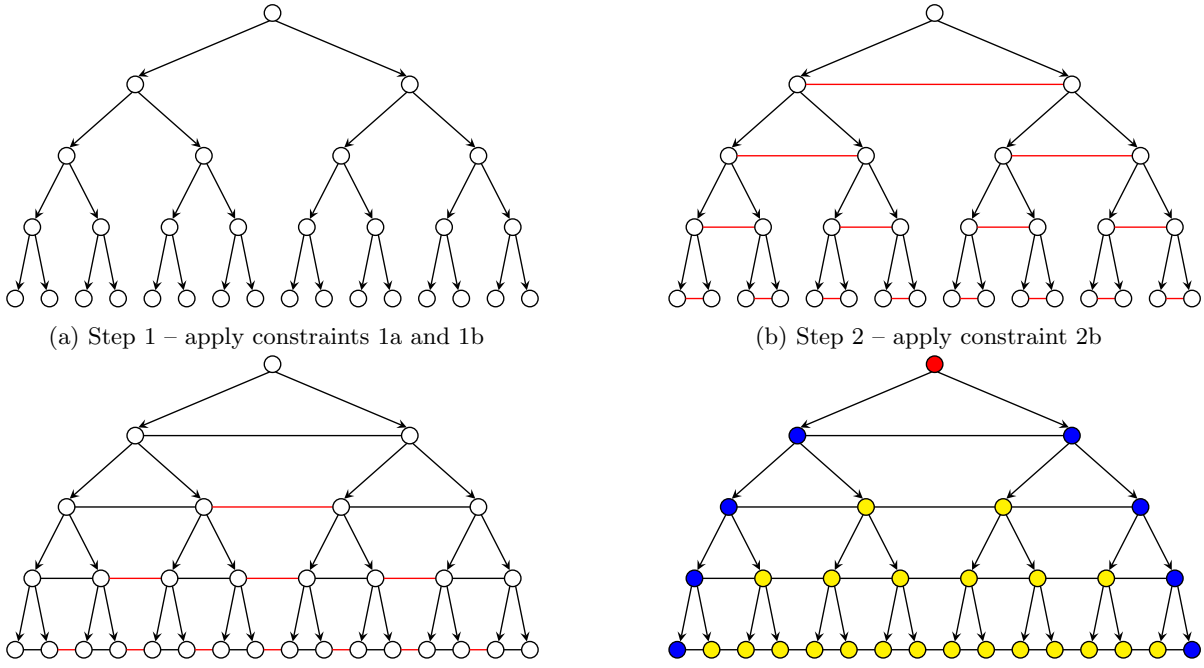
A modified version of Algorithm 1 where "the sink is a descendant of parent" is replaced with "the sink or a broken vertex is a descendant of parent" terminates when a sink or a broken vertex is found, and it leaves a valid labeling.

We now argue that it terminates in time $O(\log n)$. If no sink or broken vertex is found among the descendants, then it implies the existence of a balanced binary tree of non-broken vertices rooted at the current vertex. Each time we move to the parent of the current vertex, the size of the corresponding tree doubles. Thus only $O(\log n)$ many iterations can be done before the size of such a tree would exceed the total number of available vertices, and thus after $O(\log n)$

Problem 7 Structural constraints for $\mathcal{T}$

In a graph with some oriented edges, given a vertex v, we define the following constraints

1a. v has either zero or one parent u and there is an edge $u \to v$
1b. v has either zero or two children u_1, u_2 and there are edges $v \to u_i$
2a. v has no siblings if and only if it has no parent
2a'. v has exactly one sibling only if its parent has exactly one sibling
2a". v has no more than two siblings
2b. if v has children then they are siblings
2c. if v has children v_1 and v_2 and u is a sibling of v, then u has children u_1 and u_2 and exactly one of u_1 or u_2 is linked to exactly one of v_1 or v_2
2c' if v is a sibling of u then either they have the same parent, or their parents are siblings
3a. the edges of v to its siblings are oriented if and only if v has no children
3b. v does not have edges oriented outwards to both of its siblings



(a) Step 1 – apply constraints 1a and 1b

(b) Step 2 – apply constraint 2b

(c) Step 3 – apply constraint 2c

(d) Step 4 – no more links can be added

Figure 5: The construction of a well-formed graph results in a t_{h,j_0}: at each step of the construction, there is only a single way to add edges in a way that satisfies the imposed local constraints.

Problem 8 Path to the sink – oriented graphs

Input: An oriented graph with a parent-child relationship
Labels: red and black (same as in Problem 6)
Task: Vertices that satisfy all structural constraints from Problem 7 must also satisfy the labeling constraints from Problem 6

---

iterations there has to be either a sink or a broken vertex among the descendants of the current vertex. Once there is such a descendant, it must be at distance $O(\log n)$. □

Therefore Problem 8 has $\exists\mathsf{MVol} = O(\log n)$ in general oriented graphs. Section A.5 will further strengthen this result by showing that the orientation can be encoded in unoriented graphs. A similar technique can be applied to eliminate the need for the parent-child relation in the input graph.

A.4.4 Study of EMVol

We consider an arbitrary algorithm that relies on randomness and its local view of the graph in order to find a mend. Throughout this study, we assume that the graph is among $\mathcal{T}_h$. Assuming a graph of this structure, the performance of the algorithm is at least as good as the performance if such an assumption is not available. Note that any mend of the worst-case initial configuration, i.e., a configuration which consists of an unlabeled root and where all other vertices are black, must contain a path from the root to the sink that has to be recolored. Hence any algorithm that computes a mend must be able to find the sink. We show that the simple fact of finding the sink is a problem that cannot be solved in $O(\log n)$.

Consider an execution of the algorithm as the sequence of all explored vertices, and call the subsequence corresponding to the interval between the visit of two vertices at the bottom layer a *query*. Since the vertices of the bottom layer are the only vertices holding information on the

Algorithm 2 Generic structure of any algorithm for finding the sink in a graph of $\mathcal{T}$

Input
  t input graph from $\mathcal{T}$
  h height of t
$\mathsf{low} \leftarrow 1$; $\mathsf{high} \leftarrow 2^h$
while $\mathsf{low} < \mathsf{high}$ **do**
  choose $\mathsf{low} \leq l \leq \mathsf{high}$
  query l by exploring a path from any already explored vertex
  if the right edge of l is oriented away from l **then** ▷ the sink is right of l
    $\mathsf{low} \leftarrow l$
  else if the left edge of l is oriented away from l **then** ▷ the sink is left of l
    $\mathsf{high} \leftarrow l$
  else ▷ the sink is l
    $\mathsf{low}, \mathsf{high} \leftarrow l$
  end if
end while

position of the sink, the algorithm can be assumed to have a binary-search-like structure as presented in Algorithm 2.

Algorithm 2 describes a generic structure that fits any optimal algorithm able to find the sink in a graph from the family $\mathcal{T}$. Indeed since the only information about the position of the sink is recorded in the orientation of the edges on the bottom layer, the algorithm will necessarily explore a certain number of vertices from the bottom layer before it finds the sink. We can further assume that if the algorithm is optimal then it performs no useless query, i.e. if the algorithm has already determined that the sink is right of a certain vertex l then it will never again query vertices left of l.

We will separate the queries into two categories.

Definition A.6 (Short and long queries)**.** In Algorithm 2, if vertex l is queried while $\min(\mathsf{high} - l, l - \mathsf{low}) < h$, we call the query *short*. Otherwise, we call the query *long*.

We consider the following loose bounds: if there are k candidates for vertices that could be the sink (i.e. $\mathsf{high} - \mathsf{low} = k$), then a long query requires the exploration of at least $\omega(1)$ vertices and by performing it, it eliminates at most $n/2$ vertices from the list of candidates to be the sink; a short query requires the exploration of at least $\Omega(1)$ vertices, and, in the worst case, it eliminates at most h vertices from the list of candidates.

We show that no matter the order in which the algorithm performs long and short queries, it cannot guarantee to find the sink before exploring at least $\omega(\log n)$ vertices.

Lemma A.7. *If Algorithm 2 performs only long queries and explores $O(\log n)$ vertices, then it can eliminate at most $O(n^{1/4})$ of the candidates to be the sink.*

Proof. Considering that each long query requires the exploration of $\omega(1)$ vertices, the algorithm cannot perform more than $o(\log n)$ of these long queries before it exceeds the $O(\log n)$ limit placed on the total number of explored vertices. Write $(\log n)/f(n)$ for the amount of long queries performed, with $f(n) \to +\infty$ when $n \to +\infty$. These long queries reduce the number of candidates by a factor at most $2^{(\log n)/f(n)} = n^{1/f(n)}$. It follows that after all long queries have been performed there still remain $\Omega(n^{1-1/f(n)})$ candidates for the sink, which can be bounded from below by $\Omega(n^\alpha)$ for any $0 < \alpha < 1$. For example, picking $\alpha = 3/4$ provides the statement from Lemma A.7. □

In the following, we will show a lower bound on the number of short queries:

Theorem A.8. *Algorithm 2 must explore at least $\omega(\log n)$ vertices in order to find the sink.*

Proof. After performing a certain number of long queries as analyzed in Lemma A.7, the algorithm may no longer use long queries, and it only has access to short queries to find the sink among $n^{3/4}$ candidates. Assume a worst-case execution: each time the algorithm performs a query, the sink is revealed to be in the biggest of the two halves – and halt this execution when there are $n^{1/4}$ remaining candidates. To get to this point, there must have been at least

$$\frac{n^{3/4} - n^{1/4}}{\log N}$$

queries performed. Observe further that the probability that such a worst-case occurs is the probability that the actual sink is among the remaining $n^{1/4}$ vertices. This worst-case execution alone causes at least

$$\frac{n^{3/4} - n^{1/4}}{\log n} \frac{n^{1/4}}{n^{3/4}} = \frac{n^{1/4}}{\log n}(1 + o(1)) = \omega(\log n)$$

vertices to be explored on average. The total cost on average is at least as high as the weighted cost attributed solely to a few worst-case instances. Hence the entire algorithm cannot find the sink without exploring at least $\omega(\log n)$ vertices on average. □

Thus Theorem A.8 shows that no algorithm can find a mend for Problem 6 with expected volume $O(\log n)$. Since we have shown in Section A.4.2 that the $\exists\mathsf{MVol}$ is $\Theta(\log n)$, this shows a separation between $\exists\mathsf{MVol}$ and EMVol. Sections A.4.3 and A.5 show that unlike the separation shown in Section 5.3.2, this one does not need to rely on a promise.

This proves that there is a separation between $\exists\mathsf{MVol}$ and EMVol, since we have constructed a problem that one can solve by modifying $O(\log n)$ labels with a complete view of the graph and yet requires $\omega(\log n)$ queries when one only has access to randomness.

Note that many of the bounds we have used here are very loose, most notably the actual cost of a long query is estimated to be $\Omega(\log \log N)$ rather than $\omega(1)$. In practice, the average number of vertices visited seems to be closer to $\Omega(\log^2 N)$, even assuming that the tree has the correct structure. Further, it is not clear whether a randomized algorithm could even achieve such a bound if it also had to take into account inputs that do not satisfy the local constraints of $\mathcal{T}$.

A.5 Extension to unoriented graphs

The results from Sections A.4.2 and A.4.4 extend to unoriented graphs simply by encoding the orientation within the structure of the graph. One possible encoding of the orientation is the following: given an unoriented graph G, let *leaves* be the vertices with degree exactly 1. To interpret G as an oriented graph $\overline{G}$ with maximum degree Δ, extract the subset $\overline{V}$ of vertices that have at least $\Delta + 1$ leaves as neighbors. This provides the vertex set of $\overline{G}$. An edge between $\overline{u}$ and $\overline{v}$ is added if between the corresponding vertices u and v in G, there is a path of length exactly 2, connected to at most one leaf. If there is such a leaf on the vertex of the path closest to u, consider the edge oriented from $\overline{v}$ to $\overline{u}$. Otherwise, if there is no leaf then the edge is not oriented. Figure 6 visualizes this transformation.

A labeling λ of G is interpreted as a labeling $\overline{\lambda}$ of $\overline{G}$ by defining $\overline{\lambda}(\overline{v}) \coloneqq \lambda(v)$. Any local property of $\overline{\lambda}$ can be computed locally on λ. Problem 9 shows how this encoding can be used to define a version of Problem 6 that works on unoriented graphs.

B Landscape of mending volume

Table 1 provides an overview of the landscape only for the existential mending volume. We can perform the same work for the other measures that we have defined, which provides the

Problem 9 Path to the sink – unoriented graphs

Input: An unoriented graph G
Labels: red and black (same as in Problem 6 and Problem 8)
Task: Label G by λ so that $\overline{\lambda}$ is a labeling of $\overline{G}$ that satisfies the constraints of Problem 8

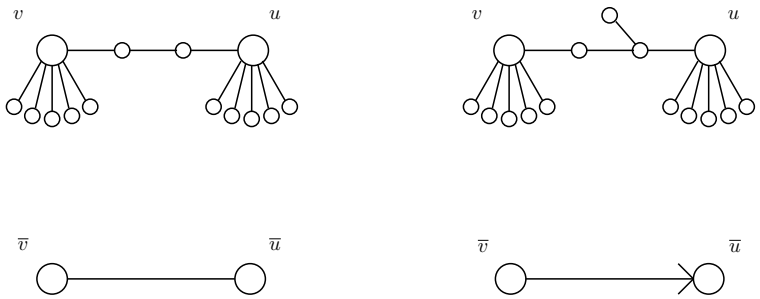


Figure 6: Oriented interpretation of an unoriented graph. Top: pairs of vertices in G; bottom: their interpretation in $\overline{G}$. This mapping can be computed locally.

results summarized in Tables 3a-3c. The results for the mending radius follow immediately from Lemma A.2 which makes the results from [6] applicable to MRad.

B.1 Landscape of EMVol

The proof of Theorem 4.11 shows that for propagation problems, the expected mending volume is equivalent to the existential mending volume. The same constructions apply to encode complexity results obtained in infinite rooted trees to general trees and general graphs.

Although we have provided a problem for which $\exists\mathsf{MVol} = o(\mathsf{EMVol})$ in Section A.4, this does not imply that the possible complexities are different, merely that they are obtained for different problems. We conjecture that Problem 6 has $\mathsf{EMVol} = \Theta(\log^2 n)$, which is a complexity that exists for the $\exists$MVol on other problems.

Thus to the best of our knowledge, EMVol exhibits the same possible complexities as $\exists$MVol.

B.2 Landscape of DMVol

The DMVol from Definition 5.3 can be shown to be a deterministic volume. Indeed if all transition probabilities are $\mathcal{M}_{G,\lambda,v}(W, W') \in \{0, 1\}$ then so is $\mathcal{M}^*_{G,\lambda,v}(\{v\}, W) \in \{0, 1\}$. The Markov chain collapses to a deterministic computational process that decides which vertices to explore next.

Our goal is to show that DMVol is either $O(1)$ or $\Theta(n)$ on most classes of graphs. This is stated by Theorem B.2.

We first prove that knowledge of the size of the instance does not increase the computational power. This means that the definition is equivalent to its variant where the next step chosen by $\mathcal{M}$ can depend on the size of the input graph, n. This invariance under the size of the instance is essential for Theorem B.2.

Lemma B.1 (Knowledge of the size of the instance is useless)**.** *The version* DMVol' *of the deterministic volume where* $\mathcal{M}_{G,\lambda,v}$ *can depend on* n *the size of the instance is asymptotically equivalent to* DMVol*.*

Proof. This proof uses a similar technique to one featured in [17]: if an upper bound on the size of the instance is necessary for the execution of the algorithm, we can simulate the algorithm by guessing increasingly large values of the size of the instance until the algorithm executes correctly.

Assume that a problem has $\mathsf{DMVol}' = \Theta(g(n))$ where n is known to the mender $\mathcal{M}$. Assume further that g is upper bounded by some known f, where f is computable and increasing function.

Table 3: An overview of the landscape of the existential mending volume (∃MVol), the expected mending volume (EMVol), the deterministic mending volume (DMVol), and the mending radius (MRad) for LCL problems on the classes of paths, trees and general graphs. Here ✓ denotes that LCL problems with this mending volume exist, × denotes that such LCL problems cannot exist, and ? denotes an open question. We assume $k > 1$ and $0 < \alpha < 1$.

(a) Landscape of ∃MVol and EMVol

Setting	Possible existential and expected mending volumes						
	$O(1)$	...	$\Theta(\log n)$	$\Theta(\log^k n)$	...	$\Theta(n^\alpha)$	$\Theta(n)$
Paths and cycles	✓	×	×	×	×	×	✓
Rooted trees	✓	×	✓	✓	×	✓	✓
Trees	✓	×	✓	✓	?	✓	✓
General graphs	✓	×	✓	✓	?	✓	✓

(b) Landscape of DMVol

Setting	Possible deterministic mending volumes						
	$O(1)$	...	$\Theta(\log n)$	$\Theta(\log^k n)$	...	$\Theta(n^\alpha)$	$\Theta(n)$
Paths and cycles	✓	×	×	×	×	×	✓
Rooted trees	✓	×	×	×	×	×	✓
Trees	✓	×	×	×	×	×	✓
General graphs	✓	×	×	×	×	×	✓

(c) Landscape of MRad

Setting	Possible mending radius						
	$O(1)$	...	$\Theta(\log n)$	$\Theta(\log^k n)$	...	$\Theta(n^\alpha)$	$\Theta(n)$
Paths and cycles	✓	×	×	×	×	×	✓
Rooted trees	✓	×	✓	×	×	×	✓
Trees	✓	×	✓	×	×	×	✓
General graphs	✓	×	✓	?	?	✓	✓

Let $f^{-1}(y)$ be the single value x such that $f(x-1) < y \leq f(x)$. Note that this is computable as well. Algorithm 3 presents a strategy that computes a mend of λ at x without the knowledge of n, by guessing the size of the graph by successively increasing it.

The main loop from Algorithm 3 must terminate since, eventually, $m \geq n$, and a mend is found for the first such m. Let $m_0, m_1, \ldots, m_k$ be the successive values of m at each iteration. Observe that $f(m_{i+1}) \geq 2 \cdot f(m_i) + 1$. Thus at each iteration the new explored vertices are as many as all previously visited vertices, so the total number of vertices explored does not exceed

$$\begin{aligned} 2 \times f(m_k) &\leq 4 \cdot f(m_{k-1}) && \triangleright \text{ by definition of } m_i \\ &\leq 4 \cdot f(n). && \triangleright \text{ since } f \text{ is increasing} \end{aligned}$$

Thus, only at a cost of a constant multiplicative factor, we can execute an algorithm that requires the knowledge of n in a context where n is unknown. Or equivalently, where the result must be independent of n. This means, that the mend computed by DMVol′ is independent of the size of the graph and thus equivalent to DMVol. □

We can now establish a complete landscape of the deterministic mending volume in Theorem B.2.

Algorithm 3 A simulation without knowledge of the size of the graph

$m \leftarrow 1$
repeat
    simulate an algorithm A computing DMVol' assuming size m
    if DMVol' explores more than $f(m) + 1$ vertices **then**
        abort the simulation
        $m \leftarrow f^{-1}(2 \cdot f(m))$
    end if
until A halts naturally

Theorem B.2 (DMVol is trivial)**.** *For any problem on paths, trees, or general graphs,* DMVol *is either* $\Theta(1)$ *or* $\Theta(n)$*.*

Proof. Assume that $\mathsf{DMVol} = \omega(1)$, then there is a sequence $(G_i)_{i \in \mathbb{N}}$ of graphs for which there are eventually arbitrarily many labels that are modified by the deterministic mender: $|M_i| \to \infty$ as $i \to \infty$, where M_i is the set of vertices of G_i whose label is modified. Define N_i to be the radius-r neighborhood of M_i, and note that the size of N_i is at most $|M_i| \times \Delta^r$. Since N_i is indistinguishable from G_i on all radius-r neighborhoods of elements of M_i, the same mending process operating on N_i will also explore exactly M_i. Hence for the sequence $(N_i)_{i \in \mathbb{N}}$ of graphs of arbitrary size, the mend is of size at least

$$|M_i| \geq \frac{|N_i|}{\Delta^r} = \Omega(n).$$

This proves a gap in deterministic volume: on the usual classes of paths, cycles, trees, and general graphs, there exists no problem with DMVol between $\omega(1)$ and $o(n)$. □

Corollary B.3 (A deterministic mender explores the entire neighborhood)**.** DMVol *is asymptotically equivalent to* $|\mathcal{N}_{\mathsf{MRad}}|$*.*

Proof. From [6] and the equivalence established in Lemma A.2, we know that MRad can only be one of $\Theta(1)$ or $\Theta(n)$ on paths and cycles, and $\Theta(1)$ or $\Theta(\log n)$ or $\Theta(n)$ on trees.

If MRad is $\Theta(1)$ then DMVol is $\Theta(1)$ as well. If MRad is $\Theta(\log n)$ or $\Theta(n)$ then DMVol is $\Theta(n)$. In paths, cycles, and trees, these are also the sizes of the neighborhoods of radius MRad. □