Mending Partial Solutions with Few Changes

Darya Melnyk
Aalto University, Finland

Jukka Suomela
Aalto University, Finland

Neven Villani
Aalto University, Finland
École Normale Supérieure Paris-Saclay, Université Paris-Saclay, France

Abstract

In this paper, we study the notion of mending: given a partial solution to a graph problem, how much effort is needed to take one step towards a proper solution? For example, if we have a partial coloring of a graph, how hard is it to properly color one more node?

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.

2012 ACM Subject Classification Theory of computation → Distributed computing models; Theory of computation → Parallel computing models

Keywords and phrases mending, LCL problems, volume model

Digital Object Identifier 10.4230/LIPIcs.OPODIS.2022.21

Related Version *Full Version*: https://arxiv.org/abs/2209.05363

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

1 Introduction

If we have a partial solution to a graph problem, how much effort is needed to take one step towards a proper solution? For example, if we have a partial coloring of a graph, how hard is it to properly color one more node? In this work we present a formalism that captures the essence of this question, the *mending volume*: how many labels do we need to change to "patch a hole" in the solution?

We will define this concept formally in Definition 5, 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' at node v that is within distance $T(n)$ from λ, where n is the number of nodes in G, we say that the mending volume of Π is at most $T(n)$.

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 – including the trivially computable empty solution, which is a partial solution in which all vertices are unlabeled – and walk towards a valid solution by patching holes one at a time 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. Whenever we detect a change close to component v, we can consider v to be a hole (unlabeled in λ); now 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 mending volume $O(1)$ or $\Theta(n)$. The work of Panconesi and Srinivasan [23] 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 [22]; 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 the mending volume that we study here and the *mending radius* that was defined recently in prior work [9]. In trees, the mending radius of any LCL problem is known to be $O(1)$, $\Theta(\log n)$, or $\Theta(n)$. Mending volume makes it possible to classify LCL problems into infinitely many classes, while mending radius only leads to three classes of problems. This is particularly relevant in balanced trees, as the diameter of a balanced tree is $\Theta(\log n)$ and hence knowing whether the mending radius is $\Theta(\log n)$ or $\Theta(n)$ does not tell us anything about the hardness of mending (both are essentially global). Mending volume, on the other hand, makes it possible to characterize the hardness in a much more fine-grained manner.

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 [23]. 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 – in which the complexity of patching a hole is exactly the mending radius – and they therefore mainly focus on the mending radius instead of the mending volume of this problem.

Table 1 An overview of the landscape of 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.

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	✓	×	✓	✓	?	✓	✓

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 [25], who show similarities and differences between the models. The volume complexity for LCL problems has been further studied by Grunau at al. [11]. 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. [9] 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 accurate than the mending radius is the 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. Assuming for now that this configuration is indeed the worst-case input (this is shown in Corollary 9 for a more general class of problems which includes all three of R_1, R_2 and R_3), the mending radius of each of these 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 have made use of mending radius, mainly as an algorithm design tool. Chechik and Mukhtar [14] 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. Similar observations have been made for computing a Δ-coloring [23] and solving an edge-orientation with maximum out-degree $(1 + \varepsilon)a$ [18]. Recently, it has

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.

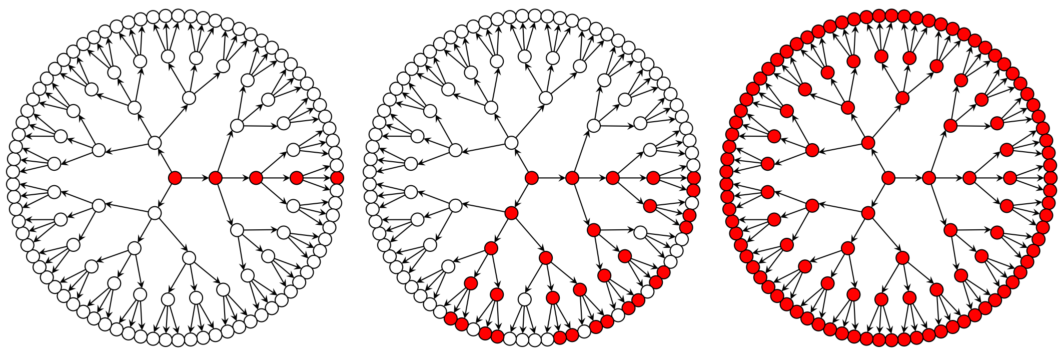


Figure 1 From left to right, solutions of R_1, R_2, R_3 (as defined in Problem 1) with the least number of red labels are visualized. In the case of R_2 (middle), each red vertex starting from the root in the center has two of its three children colored red, and this continues down to the leaves. The radius in all three of these examples is 4, and its growth rate as the graph gets larger is $\Theta(\log_3 n)$. The volume of R_2 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.

been shown that mending algorithms with a constant radius can also be transformed into self-stabilizing algorithms in anonymous networks [15]. On the other hand, there are also papers that have introduced an explicit notion of mending, although using different definitions of partial solutions and complexity measures – this includes for example König and Wattenhofer [20] and Kutten and Peleg [21]. König and Wattenhofer [20] consider only faults that are an addition or deletion of a single vertex or edge at a time, and hence their definition of partial solutions features only at most a small number of unlabeled vertex. They also classify the cost of fixing a fault only in terms of local (constant-radius) or global (non-constant radius). Kutten and Peleg [21] 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. [12, 16, 24]. Then, a local search algorithm is applied to find a better solution in the vicinity of the previous one.

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, 3]. On the negative side, Johnson et al. [19] showed that an exponential number of iterations may be needed if the cost function can take exponential values. Ausiello and Protasi [5] 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 [17] 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 [13] later showed a $2(k+1)/3$-approximation algorithm for the weighted k-set packing problem, thus improving a previous result from Bafna et al. [8] and Arkin and Hassin [4].

Self-stabilization. Mending can also be seen as an approach for the correction step of self-stabilizing algorithms. Self-stabilization is typically implemented by the conjunction of a failure detector [10] and a recovery procedure [6]. Similarly to prior work in [7] and [2], our approach applies to problems where inconsistencies are locally detectable, but recovering from failures usually requires global reconfiguration. Indeed the restriction of our study to the class of LCL problems guarantees that the consistency of the state can be checked with only a local view, but since the mending radius in [9] already covers the case of problems that can be corrected with constant locality, most of our new results apply to problems that are not locally correctable. Compared to [26], our setting is more restricted since for the problems we study the history of a vertex is not relevant, but we are also interested in optimality of the final solution in terms of its Hamming distance to the initial solution. We also have the same point of view as [26] that if the complexity is chosen to be the execution time, then it measures more properties related to the implementation or the model than inherent properties of the problem and hence we aim for a model-independent notion. Unlike [26] however, in cases where several failures occur simultaneously and are repaired sequentially through successive mending procedures each resulting in an intermediate partial solution, our measure of complexity considers only the distance from one partial solution to the next and it does not guarantee minimizing the total distance from the initial solution to the final complete solution. The other main difference is that we classify problems in terms of the distance to the nearest solution whereas the measure of locality in [26] is the radius of the view – i.e. the amount of information – that is required to compute said nearest solution.

3 Preliminaries

Our definition of the mending volume is built along the lines of the definition of the mending radius in [9]: 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 [22]. We use the same definition of partial solutions as [9] 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 [9] may skip directly to Section 4 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 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.

We write $\mathcal{N}_r(v)$ for the radius-r neighborhood of v and $(G, \lambda)_{|V}$ for the restriction of G and λ to the subgraph and labeling with only vertices from V, this constraint can be expressed more formally as: $(G_1, \lambda_1)_{|\mathcal{N}_r(v_1)} \simeq (G_2, \lambda_2)_{|\mathcal{N}_r(v_2)}$ 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 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 one step 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 $\lambda_\perp : x \mapsto \perp$) 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 [9], 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** (Partial solution). *For* $\Pi = (\Sigma, \mathcal{G}, \phi)$*, where* ϕ *has radius* r*, let* $\Sigma^* = \Sigma \sqcup \{\perp\}$*, and define a relaxation* $\Pi^* = (\Sigma^*, \mathcal{G}, \phi^*)$ *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* $\phi^*(G, \lambda', v) := \mathsf{happy}$*; otherwise, let* λ *be any labeling over* Σ *that agrees with* λ' *on* $G_{|\mathcal{N}_v(r)}$ *and set* $\phi^*(G, \lambda', v) := \phi(G, \lambda, v)$*.*

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; this fact is also proven in [9]. 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 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. We call such a tuple (G, λ, v) an instance of $\mathsf{Mend}(\Pi)$, and λ' a solution.

4 Complexity landscape of mending volume

Having defined LCLs and partial solutions, we can now introduce 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 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 Mending volume: Definition

For two labelings λ and λ', we define $\operatorname{diff}(\lambda, \lambda') := \{v\colon\ \lambda(v) \neq \lambda'(v)\}$ such that $|\operatorname{diff}(\lambda, \lambda')|$ is the Hamming distance between two partial solutions. We define the 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. $\lambda(v) = \bot$ at which λ must be mended.

▶ **Definition 5** (Mending volume). *The mending volume of problem* Π *is*

$$\mathsf{MVol}(\Pi) := \max_{G,\lambda,v} \min \{|\operatorname{diff}(\lambda, \lambda')| : \lambda' \textit{ is a mend of } \lambda \textit{ at } v\}.$$

4.2 Mending in infinite rooted trees

In what follows, we will establish the landscape of possible mending volumes. For a summary, please refer to Table 1 that was introduced earlier. To this end, we give examples of LCL problems that have logarithmic, polylogarithmic and polynomial mending volumes. The definitions of these problems are made easier by the fact that all of them are of a specific type that we call *propagation problems.*

The complexity analysis of problems in this class is 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 regular trees thanks to results from Section 4.3 which allow us to transfer complexity results from infinite regular trees to finite graphs. The advantage of infinite regular trees is that the complexity analysis is simplified by the absence of 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, as proven in Section 4.3, our choice does not restrict the generality of our results when it comes to existence results. This does however makes the impossibility results not directly applicable to the case of general graphs. In fact, we do not yet know if these results still hold for general graphs.

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 6** (Infinite Δ-regular rooted trees). *An infinite Δ-regular rooted tree (or simply infinite rooted trees when Δ is clear from the context) is a tree with the following properties:*

- *exactly one vertex is distinguished as the root;*
- *each vertex admits a unique directed path to the root;*
- *each vertex has exactly Δ incoming edges.*

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 7.

▶ **Definition 7** (Construction of a propagation problem). *In 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'" and "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 9 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, it will be useful to study the volume 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 $M[l, l'] = \mu(l, l')$. This means that the coefficient $M[l, l']$ indicates how many children labeled l' each vertex labeled l must have. 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. By induction, a vertex labeled l must have at least $M[l, l']$ children labeled l' at distance 1; each of its children labeled l'' at distance d (of which there are $M^d[l, l'']$ by inductive hypothesis) must then have at least $M[l'', l']$ children labeled l', which makes for a total of

$$\sum_{l'' \in \Sigma'} M^d[l, l''] M[l'', l'] = M^{d+1}[l, l']$$

children labeled l' at distance $d + 1$. We argue in Theorem 8 that this provides bounds for MVol.

We write $\|L_l M^d\| \coloneqq \sum_{l' \in \Sigma'} M^d[l, l']$, where L_l is the vector with a 1 only in position l. This corresponds to counting the total number of children of l at distance d that have any label among Σ', which is also the total number of modified labels at distance d when starting from the initial labeling that consists of an unlabeled root and all other vertices labeled l_{-}. We show in Theorem 8 that this quantity expresses both upper and lower bounds on the mending volume up to a fixed distance $d_{\max}$ of the propagation problem described by M.

▶ **Theorem 8** (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 all vertices in the input graph 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 from the root, there must be at least $M^p[l_0, l']$ vertices in λ' labeled l' that must have been modified during the mending. Therefore $\sum_{d=0}^{d_{\max}} \|L_{l_0} M^d\|$ 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 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 9** (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 9 is satisfied at least for problems where all labels are reachable from l_0 in the sense that any complete solution must contain every label at least once. Put otherwise, for every $l' \in \Sigma'$ there must exist some $d_{l'}$ for which $M^{d_{l'}}[l_0, l'] \neq 0$. This is also the case for every propagation problem we will define in the rest of this paper.

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 upper bounded by $|\Sigma'| \times \max_{l' \in \Sigma'} M^d[l, l']$ (approximate a sum by the number of elements multiplied by the greatest element). In order to determine the mending volume up to a multiplicative constant of a propagation problem in which all labels are reachable from the root label, we have thus established that 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. Here 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 if $c(l) = 0$, type 1 if $c(l) = 1$, and type 2 if $c(l) \geq 2$. Figure 2 shows an example of a matrix M and the corresponding graph G_M.

We will show that the types of the vertices fully determine 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 10.** *Assume that v_l is a vertex of type 2. It holds that $M^d[l, l] = \Omega((1 + \beta)^d)$ for some $\beta > 0$.*

$$M = \left(\begin{array}{ccccc} & A & B & C & D \\ A & 2 & 4 & 0 & 0 \\ B & 0 & 0 & 1 & 2 \\ C & 0 & 1 & 0 & 0 \\ D & 0 & 0 & 0 & 0 \end{array}\right)$$

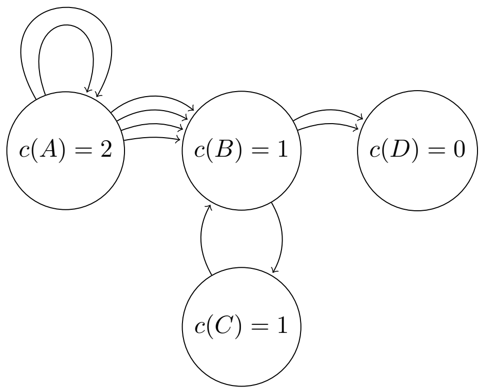


Figure 2 An example of matrix M describing a propagation problem over the label set $\Sigma' = \{A, B, C, D\}$. The corresponding G_M is shown on the right, where A is of type 2, B and C are of type 1, and D is of type 0.

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 $L := \operatorname{lcm}(L_1, L_2, \dots)$. 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 $d := kL$ from v_l to itself and therefore $M^d[l,l] \geq c(l)^{d/L}$ walks for infinitely many values of d. Thus $M^d[l,l] = \Omega((c(l)^{1/L})^d)$. ◀

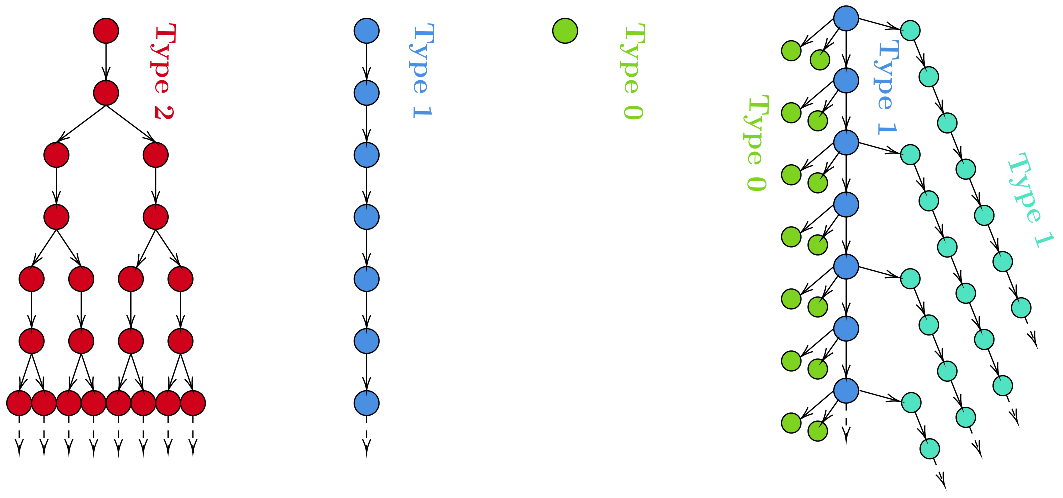
▶ **Corollary 11.** *Let vertex v_l be of type 2 and reachable from v_{l_0}. Then $M^d[l_0, l] = \Omega((1+\beta)^d)$ for some $\beta > 0$.*

In terms of the corresponding propagation problem, the interpretation of this fact is that any label l corresponding to a vertex of type 2 must have at least two descendants with the same label l at a certain fixed distance, each of which must also satisfy the same property. This produces an exponentially increasing amount of vertices labeled l as we move further away from the root.

▶ **Lemma 12.** *If there is no vertex of type 2 reachable from v_{l_0} then for all labels l, $M^d[l_0, l] = O(d^k)$ for some constant k.*

Proof. For each vertex v_l, we denote $C(l)$ to be the cluster of v_l, defined as follows: $C(l) := \{l\}$ if $c(l) = 0$; otherwise, $C(l) := \{l' \mid v_l \to^* v_{l'} \to^* v_l\}$ 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. We now consider all possible walks from l_0 to some label l by separating each step of the walk between those that stay in the same cluster and those that change cluster. The constraints on the cycles of the graph obtained from the assumption that every vertex is of type at most 1 give us information on how many times a step of the walk can change cluster.

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 as, otherwise, some vertex would be part of several cycles and would not be of type 0 or 1 as we have assumed. Any walk W of length exactly d in G_M from v_l to $v_{l'}$ is uniquely defined by



$M_2 = \begin{pmatrix} 0 & 1 \\ 2 & 0 \end{pmatrix}$ $M_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ $M_0 = \begin{pmatrix} 0 \end{pmatrix}$ $M_{1,1,0} = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$

Figure 3 Examples of subtrees obtained by considering the vertices with a label from the same cluster, for clusters of type 2 (left, red), type 1 (center left, blue) and type 0 (center right, green). Below each tree is an example matrix M that exhibits the behavior in question. As stated in Lemma 10, a cluster of type 2 produces an exponentially growing tree where some vertices have several children within the same cluster. This is different to a cluster of type 1, in which each vertex has exactly one child within the same cluster, and cluster of type 0 where there are no cycles and thus also no children from the same cluster. The last tree on the right illustrates Lemma 12, where, no matter how we combine clusters of types 1 and 0 (in this specific case the clusters are $\{1, 2\}$ of type 1, $\{3\}$ of type 1, and $\{4\}$ of type 0), the clusters can never achieve an exponential growth the same way a type 2 cluster can. They can therefore produce at most polynomial growth with a larger polynomial degree as we combine more clusters.

- 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 $d_1 + \cdots + d_{|W|} + (|W| - 1) \leq d$ since the total length of the walk is more than the sum of each walk inside a cluster, and $|W| \leq K \leq |\Sigma'|$ since each cluster is of size at least 1. There are finitely many walks W' in G'_M, since it is a finite acyclic graph, and for each of them the number of possible tuples $(d_1, \ldots, d_{|W|})$ is bounded by d^K. Thus the number of walks W of length d is polynomially bounded by $O(d^K)$. ◀

In contrast to Lemma 10, the interpretation of this situation in terms of the underlying propagation problem is as follows: if there are only vertices of type 0 or 1 then, from each vertex labeled l, there is at most one path that contains other vertices labeled l. Figure 3 shows what we can expect solutions to typical propagation problems satisfying the conditions from Lemmas 10 and 12 to look like.

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 13 holds.

▶ **Theorem 13** (Landscape of $\max M^d$). *The growth rate of $d \mapsto \max |M^d|$ is either*

1. *eventually zero;*
2. *or $\Theta(1)$;*
3. *or $O(d^p)$ for some value $p \geq 1$;*
4. *or $\Omega((1+\beta)^d)$ for some $\beta > 0$.*

By comparing Theorem 13 to Figure 3, we can see that all these families of growth rates are represented there: from left to right, $2^{d/2} = \Omega((1+\beta)^d)$, constant 1 which is of course in $\Theta(1)$, eventually zero, and $d/2 + 3 = O(d^p)$.

We can combine Theorem 13 with the previously established bounds from Theorem 8 stating how to relate the mending volume to the growth of $\max M^d$. We thus obtain the following corollary:

▶ **Corollary 14** (Landscape of the mending volume on infinite rooted trees). *The mending volume up to distance $d := \log_\Delta n$ 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 $O(1)$, $\text{poly}(\log n)$ 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 From infinite regular rooted trees to general trees

Some of the results from Section 4.2.3 are 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 detailed in Problem 2, which shows how for any generic problem Π we can remove the constraints of finiteness and Δ-regularity while preserving the mending volume complexity.

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

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 the mending process is just as efficient in the case that the tree is unbalanced as it is in a balanced tree, by proving the following intermediate result: a labeling picked at random among a set of possible labelings of an unbalanced tree is on average asymptotically as efficient as the optimal labeling of a balanced tree. This implies that the optimal labeling of an unbalanced tree is asymptotically as efficient as the optimal labeling of a balanced tree. To show this, we use the fact that the optimal labeling must be at least as efficient as the average performance of several valid labelings.

▶ **Theorem 15** (Generalization to finite unbalanced trees). *If Π is a propagation problem, it has the same mending volume complexity on unbalanced trees as it does on balanced Δ-regular trees.*

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 eventually concave in n (which possibly includes eventually constant). This is because there are at most Δ as many modified labels at any distance $d+1$ as at the previous distance thus $\|L_l M^{d+1}\| \le \Delta \|L_l M^d\|$, yet from d to $d+1$ the total number of vertices grows by a factor exactly Δ.

The total number of modified labels in a tree of size $n+1$ is thus

$$f^{\text{bal}}(n+1) = 1 + \sum_{j=1}^{\Delta} f_j^{\text{bal}}(n/\Delta).$$

Assume inductively that the true number of modified labels in any non-balanced tree of size n' is less than $K \cdot f_j^{\text{bal}}(n')$, i.e. that a balanced tree is asymptotically 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

$$\begin{aligned}
c' &= \operatorname{avg}_{\sigma \in \mathfrak{S}_\Delta} 1 + \sum_{i=1}^{\Delta} c_{i,\sigma(i)} && \\
&\le \operatorname{avg}_{\sigma \in \mathfrak{S}_\Delta} 1 + \sum_{i=1}^{\Delta} K \cdot f_{\sigma(i)}^{\text{bal}}(n/\Delta + d_i) && \text{induction hypothesis} \\
&\le 1 + K \cdot \sum_{j=1}^{\Delta} \operatorname{avg}_{\sigma \in \mathfrak{S}_\Delta} f_j^{\text{bal}}(n/\Delta + d_{\sigma(j)}) && \text{reassign indices} \\
&\le 1 + K \cdot \sum_{j=1}^{\Delta} f_j^{\text{bal}}(n/\Delta) && \text{by concavity of all } f_j^{\text{bal}} \\
&\le K \cdot f^{\text{bal}}(n+1).
\end{aligned}$$

The induction hypothesis trivially holds for some big enough K once given n that satisfies the main requirement of functions f_j^{bal} reaching their eventual concavity, hence the result for all big enough sizes of trees.

For now this result is implicitly restricted by the condition that n be divisible by Δ, indeed $f_j^{\text{bal}}(n/\Delta)$ is not defined in general for fractional n/Δ. The result extends to all sizes of trees – still under the condition that they are big enough – from the following observation: between n and the next larger integer divisible by Δ there is at most a constant multiplicative factor Δ which translates to at most a multiplicative Δ additional labels being modified. This multiplicative factor has no impact on the asymptotic behavior. ◀

We further show that if the labeling assumes an orientation of the edges then a variant of the problem with the same mending volume complexity can be defined on unoriented graphs by encoding the orientation in the graph structure. One possible encoding of the orientation

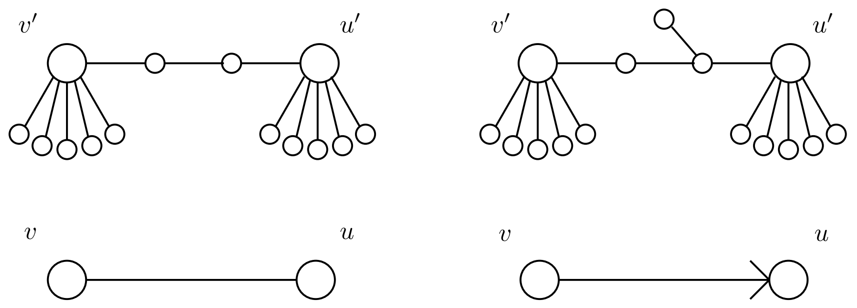


Figure 4 Oriented interpretation of an unoriented graph. Top: pairs of vertices in G; bottom: their interpretation in $\overline{G}$. Note that G is a tree if and only if $\overline{G}$ is a tree.

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 4 visualizes this transformation.

A labeling λ of G is interpreted as a labeling $\overline{\lambda}$ of $\overline{G}$ by defining $\overline{\lambda}(\overline{v}) := \lambda(v)$.

Any local property that $\overline{\lambda}$ should satisfy can be verified locally on λ: a property of $(\overline{G}, \overline{\lambda})$ that can be computed from the restriction $(\overline{G}, \overline{\lambda})_{|\overline{V}}$ to a neighborhood $\overline{V}$ can also be computed from $(G, \lambda)_{|\mathcal{N}_2(V)}$, where $V = \{u \mid \overline{u} \in \overline{V}\}$.

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

In this section and the next, we show examples of problems that exhibit polynomial and polylogarithmic complexities, with a particular focus on showing which values of $0 < \alpha < 1$ and $k \geq 1$ can appear for complexities $\Theta(n^{\alpha})$ and $\Theta(\log^k n)$.

The prior analysis resulting in Corollary 14 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.

One can notice that this problem happens to be the same as one previously introduced in Definition 1 and for which an example solution is shown in Figure 1 (middle). Using the terminology of Definition 7, 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. $\Theta(n^{\ln 2/\ln 3})$.

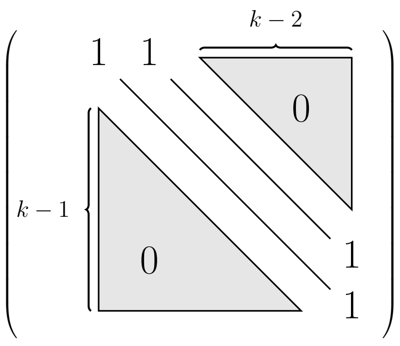


Figure 5 M_k of size $k \times k$ exhibits growth $\Theta(p^k)$ for the root label l_0 corresponding to line 1.

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 $\Theta(n^{\ln 2^p / \ln 2^q}) = \Theta(n^{p/q})$.

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

We now show how to construct a problem that has mending volume $\Theta(\log^k n)$ for any chosen $k \geq 1$. This time, Corollary 14 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 5: 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 Conclusions and discussion

In this work, we have defined the mending volume and shown that it is able to distinguish more complexity classes of problems than the mending radius from [9], which is the most closely related previously studied notion.

Cost of reading vs. cost of writing. Observe that the mending volume as defined here does not yet determine the size of the neighborhood that needs to be *explored* in order to decide what to change. Indeed the definition of the mending volume as the optimum over all possible mends implies that in order to compute a volume-optimal mend one would need to explore all possible mends and thus would have to gather information on the entire neighborhood up to the mending radius. This suggests that the mending volume is advantageous primarily in situations in which the cost of gathering information is negligible compared to the cost of actually performing the modifications.

In the full version of this work we also explore *algorithmic* versions of mending volume, which capture the idea of how many nodes need to be explored in order to find a mend.

Patching one hole vs. patching all holes. While our measure guarantees optimality of a single mend at a specific vertex, it does not guarantee optimality of the final solution when executing several mends sequentially. It also does not prevent the fact that some labels

]

might be overwritten repeatedly each time a new mend is computed to patch a different hole. Compare this to [26] for example, which provides strong guarantees of monotonicity (each node changes label at most once) and goal optimality (the final solution is the nearest one in terms of Hamming distance).

This shortcoming is not unexpected: a closer look at the problems we study here reveals that they fall under Condition 1 of Theorem 1 of [26]: the optimal configuration can depend on the initial configuration at an arbitrary distance of the vertex of interest. This means that finding the optimal solution could require global information. Thus, the kind of problems that the mending volume is best at classifying is precisely a class of problems to which the results of [26] do not apply.

Open questions. Although our encoding from propagation problems to general graphs in Section 4.3 makes our existential results applicable to the case of general trees and general graphs, the impossibility results do not work analagously. One major unsolved question is whether there are problems that have a mending volume strictly between polylogarithmic and polynomial, either in general trees or in general graphs.

References

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

2 Yehuda Afek, Shay Kutten, and Moti Yung. Memory-efficient self stabilizing protocols for general networks. In Jan van Leeuwen and Nicola Santoro, editors, *Distributed Algorithms*, pages 15–28, Berlin, Heidelberg, 1991. Springer Berlin Heidelberg.

3 Eric Angel. *A Survey of Approximation Results for Local Search Algorithms*, pages 30–73. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. doi:10.1007/11671541_2.

4 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.

5 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.

6 Baruch Awerbuch, Boaz Patt-Shamir, and George Varghese. Self-stabilization by local checking and correction (extended abstract). In *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science*, SFCS '91, pages 268–277, USA, 1991. IEEE Computer Society. doi:10.1109/SFCS.1991.185378.

7 Baruch Awerbuch, Boaz Patt-Shamir, George Varghese, and Shlomi Dolev. Self-stabilization by local checking and global reset. In Gerard Tel and Paul Vitányi, editors, *Distributed Algorithms*, pages 326–339, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

8 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.

9 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.

10 Joffroy Beauquier, Sylvie Delaët, Shlomi Dolev, and Sébastien Tixeuil. Transient fault detectors. *Distrib. Comput.*, 20(1):39–51, July 2007. doi:10.1007/s00446-007-0029-x.

11 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.

12 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.

13 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.

14 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.

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

16 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.

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

18 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.

19 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.

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

21 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.

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

23 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.

24 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.

25 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.

26 Yukiko Yamauchi and Sébastien Tixeuil. Monotonic stabilization. In Chenyang Lu, Toshimitsu Masuzawa, and Mohamed Mosbah, editors, *Principles of Distributed Systems*, pages 475–490, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.