Simpler and Improved Replacement Path Coverings

Davide Bilò
University of L'Aquila, Italy

Shiri Chechik
Tel Aviv University, Israel

Keerti Choudhary
Indian Institute of Technology Delhi, India

Sarel Cohen
Reichman University, Herzliya, Israel

Martin Schirneck
Karlsruhe Institute of Technology, Germany

Abstract

An important tool in the design of fault-tolerant graph data structures are (L, f)-*replacement path coverings* (RPCs). An RPC is a family $\mathcal{G}$ of subgraphs of a given graph G such that, for every set F of at most f edges, there is a subfamily $\mathcal{G}_F \subseteq \mathcal{G}$ with the following properties.

1. No subgraph in $\mathcal{G}_F$ contains an edge of F.
2. For each pair of vertices s, t that have a shortest path in $G-F$ with at most L edges, one such path also exists in some subgraph in $\mathcal{G}_F$.

The *covering value* of the RPC is the total number $|\mathcal{G}|$ of subgraphs. The *query time* is the time needed to compute the subfamily $\mathcal{G}_F$ given the set F.

Weimann and Yuster [TALG'13] devised a randomized RPC with covering value $\widetilde{O}(fL^f)$ and query time $\widetilde{O}(f^2L^f)$. This was derandomized by Karthik and Parter [TALG'24], who also reduced the query time to $\widetilde{O}(f^2L)$. Their approach uses some heavy algebraic machinery involving error-correcting codes and an increased covering value of $O((cfL\log n)^{f+1})$ for some constant $c > 1$. We instead devise a much simpler derandomization via conditional expectations that lowers the covering value back to $\widetilde{O}(fL^{f+o(1)})$ and decreases the query time to $\widetilde{O}(f^{5/2}L^{o(1)})$, assuming $f = o(\log L)$.

We also investigate the optimal covering value of any (L, f)-replacement path covering (deterministic or randomized) for different parameter ranges. We provide a new randomized construction as well as improving a known lower bound, also by Karthik and Parter. For example, for $f = o(\log L)$, we give an RPC with $\widetilde{O}((L/f)^f\, L^{o(1)})$ subgraphs and show that this is tight up to the $L^{o(1)}$ term.

2012 ACM Subject Classification Theory of computation → Data structures design and analysis; Theory of computation → Pseudorandomness and derandomization; Mathematics of computing → Graph algorithms

Keywords and phrases derandomization, fault tolerance, replacement path coverings, sensitivity data structures

Digital Object Identifier 10.4230/LIPIcs.ICALP.2026.35

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `http://arxiv.org/abs/2604.27966`

Funding *Shiri Chechik*: This project received funding from the European Research Council (ERC) under the European Union's Horizon 2020 Research and Innovation program, grant agreement No. 803118 "The Power of Randomization in Uncertain Environments".
Keerti Choudhary: The author is supported by the Indian Anusandhan National Research Foundation (ANRF) under the Mathematical Research Impact-Centric Support (MATRICS) scheme, grant agreement No. MTR/2025/001601.
Martin Schirneck: The author is supported by the German Research Foundation (DFG), grant agreement No. 556899211 "Design, Analysis, and Engineering of Enumeration Algorithms".

53rd International Colloquium on Automata, Languages, and Programming (ICALP 2026).
Editors: Sayan Bhattacharya, Danupon Nanongkai, Michael Benedikt, and Gabriele Puppis;
Article No. 35; pp. 35:1–35:19

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Graphs are powerful models in computer science representing various types of relationships between entities encountered in different applications. While a wide range of algorithms and data structures has been developed for static graphs, where edges and vertices remain unchanged over the whole lifetime of the application, real-world networks are often subject to failures. Many traditional graph algorithms need to recompute their solutions entirely when some component of the input changes, even if it is only a small number edges. In many practical cases, there is an a priori upper bound on the number of simultaneous failures. Moreover, while it may be unpredictable where the faults occur, they are often transient due to an inherent repair mechanism in the network. This motivates the *fault-tolerant* or *sensitivity* setting in data structure research. There, a preprocessing algorithm is given the underlying graph without failures and a *sensitivity* parameter f. Queries to the data structure then specify up to f edges and the task is to quickly report the properties of the graph in which the specified edges failed. Over the last two decades, substantial advancements have been made in developing such *sensitivity oracles* for fundamental graph problems like connectivity [21, 22, 23, 34], shortest paths [3, 4, 11, 12, 14, 17, 18, 21, 24, 26], diameter and excentricity [5, 8, 27], and routing [13]. Recently, similar ideas have also been applied to NP-hard problems, such as vertex cover, k-path [1, 7], or k-clique [6].

We are concerned with *f-edge fault-tolerant distance sensitivity oracles* (f-DSOs), which are sensitivity oracles for pairwise graph distances. In more detail, an f-DSO for a graph $G = (V, E)$ is queried with triplets (s, t, F) consisting of two vertices $s, t \in V$ and a set $F \subseteq E$ of at most f edges. The output of the oracle is the length $d(s, t, F)$ of the shortest path from s to t in the modified graph $G-F$. This value $d(s, t, F)$ is called the *replacement distance* and any shortest s-t-path in $G-F$ is a *replacement path*. Weimann and Yuster, in their seminal work [35], presented a non-trivial f-DSO supporting multiple edge failures. It has separate logic and data structures for so-called hop-short and hop-long replacement paths. Here, a replacement path is *hop-short* if it has at most L edges, where L is some positive integer parameter. For such paths, Weimann and Yuster gave an insightful construction that is now known as an *(L, f)-replacement path covering* (RPC).[1] They defined a collection $\mathcal{G}$ of $\widetilde{O}(fL^f)$ random subgraphs[2] of G that with high probability[3] has the following property. Whenever two vertices s, t and a set F of at most f edges are such that s and t indeed have a hop-short replacement path in $G-F$, then there exists a subgraph $G^* \in \mathcal{G}$ such that G^* contains no edge of F and at least one such replacement path is retained in G^*.

The usefulness of RPCs for distance sensitivity oracles stems from the following observation. Suppose we are given a query set F. Scanning through $\mathcal{G}$ and filtering for those subgraphs that have no edge of F gives a subfamily $\mathcal{G}_F$. For vertices s and t, let $\widehat{d}_F(s, t)$ be the minimum s-t-distance among all graphs in $\mathcal{G}_F$. $G^* \in \mathcal{G}_F$ guarantees that $\widehat{d}_F(s, t)$ is equal to the *true* replacement distance $d(s, t, F)$ whenever s and t have a replacement path with at most L edges. Even if s and t only have hop-long paths, we still have $\widehat{d}_F(s, t) \geqslant d(s, t, F)$. The computed value never underestimates the replacement distance since any shortest path that contributed to $\widehat{d}_F(s, t)$ only uses edges from $G-F$. A caveat of the construction in [35] is that, in order to find $\mathcal{G}_F$, all graphs of the family $\mathcal{G}$ need to be scanned, taking time $\widetilde{O}(f^2L^f)$.

[1] The name was introduced later by Karthik and Parter in (the conference version of) [29].
[2] We use n for the number of vertices of the input graph G, and m for the number of its edges. For a positive function $g(m, n, L, f)$, we let $\widetilde{O}(g)$ stand for $O(g \cdot \mathsf{polylog}(n))$.
[3] *With high probability* (w.h.p.) means with probability at least $1 - n^{-c}$ for some constant $c > 0$.

Several applications of replacement path coverings have since been explored in the context of DSOs [2, 3, 11, 25, 29], k-path sensitivity oracles [6], fault-tolerant spanners [9, 19, 20], and fault-tolerant strong-connectivity preservers [10]. Additionally, the concept has also been utilized in distributed computing [15, 28, 31, 32, 33]. In light of these applications, we adopt the following definition that is equivalent to the one by Weimann and Yuster [35] but focuses on the subfamily $\mathcal{G}_F$ instead of the individual subgraph G^*. The idea is that $\mathcal{G}_F$ provides good estimates for *all* pairs (s,t) simultaneously. We have to formulate the definition carefully so that it also applies to settings in which there are several shortest s-t-paths in $G-F$, some of which may have more than L edges.

▶ **Definition 1** (replacement path coverings, covering value, query time)**.** *Let L and f be positive integers and $G=(V,E)$ a graph. An* (L,f)-replacement path covering *for G is a family $\mathcal{G}$ of spanning subgraphs of G that has a subfamily $\mathcal{G}_F \subseteq \mathcal{G}$ for every set $F \subseteq E$ of $|F| \leqslant f$ edges such that the following two properties hold.*

1. *No subgraph in $\mathcal{G}_F$ contains an edge of F.*
2. *For all $s,t \in V$ such that there exists a shortest path from s to t in $G-F$ with at most L edges, at least one subgraph in $\mathcal{G}_F$ also has such a path.*

The covering value *of the (L,f)-replacement path covering is the number of subgraphs in $\mathcal{G}$. Its* query time *is the time required to compute $\mathcal{G}_F$ from F.*

Another interesting parameter of RPCs is the number $|\mathcal{G}_F|$ of subgraphs relevant for a given query set F. In down-stream applications, this translates to the number of instances that need to be processed in the hop-short case to find replacement distance $d(s,t,F)$.

Alon, Chechik, and Cohen [2] derandomized several distance sensitivity oracles and asked whether also (L,f)-replacement path coverings in general can be derandomized. This would provide a new deterministic tool for the design of sensitivity data structures also for other applications beyond shortest paths. Karthik and Parter [29] answered this question affirmatively using algebraic error-correcting codes. Their RPC has a covering value of $O((cfL\log_2 n)^{f+1})$ for some constant $c>1$. We abbreviate this to $\widetilde{O}(fL)^{f+1}$ (the exponent outside of the parentheses is intentional). This is more than the original number of $\widetilde{O}(fL^f)$ subgraphs [35]. In [29, Section 1.3], the authors explain that the increase is a direct consequence of the determinism of their construction. However, the fact that their preprocessing is reproducable is also the reason for their much better $\widetilde{O}(f^2L)$ query time. Very recently, Bilò, Choudhary, Cohen, Friedrich, and Schirneck [6] presented a new RPC for the parameter range $f=o(\log L)$ simultaneously reducing the covering value and query time (more details below). Their construction is *randomized*, which raises the question, whether randomness is required to achieve this improved performance.

Our first result rejects this hypothesis. We completely derandomize the result by Bilò et al. [6] without any loss in the parameters. We thereby obtain a deterministic (L,f)-replacement path covering whose covering value is only slightly larger than the construction by Weimann and Yuster [35], but achieves a query time that is sub-polynomial in L. A summary of the related work and our own result can be found in Table 1.

▶ **Theorem 2.** *Let G be a graph (possibly directed and positively edge-weighted) with n vertices. Let f and L be two positive integers, which may depend on n, such that $f=o(\log L)$. There exists a deterministic (L,f)-replacement path covering for G with covering value $\widetilde{O}(fL^{f+o(1)})$ and query time $\widetilde{O}\big(f^{5/2}L^{o(1)}\big)$. The size of the computed subfamily $\mathcal{G}_F$ is $\widetilde{O}(fL^{o(1)})$.*

For sensitivities up to $f=o(\log n)$, the covering value becomes $L^f n^{o(1)}$ and the query time as well as the size of $\mathcal{G}_F$ increase to $n^{o(1)}$.

Table 1 Comparison of (L, f)-replacement path coverings for sensitivity $f = o(\log L)$. Randomized results hold with high probability.

Covering Value	Query Time	Size of $\mathcal{G}_F$	Randomization	Reference
$\widetilde{O}(fL^f)$	$\widetilde{O}(f^2L^f)$	$\widetilde{O}(fL^{f-\lvert F\rvert})$	randomized	[35]
$\widetilde{O}(fL)^{f+1}$	$\widetilde{O}(f^2L)$	$\widetilde{O}(fL)$	deterministic	[29]
$\widetilde{O}(fL^{f+o(1)})$	$\widetilde{O}(f^{\frac{5}{2}}L^{o(1)})$	$\widetilde{O}(fL^{o(1)})$	randomized	[6]
$\widetilde{O}(fL^{f+o(1)})$	$\widetilde{O}(f^{\frac{5}{2}}L^{o(1)})$	$\widetilde{O}(fL^{o(1)})$	deterministic	Theorem 2
$\widetilde{O}\big(fe^f(\frac{L}{f})^{f+o(1)}\big)$	$\widetilde{O}\big(f^{\frac{5}{2}}e^f(\frac{L}{f})^{o(1)}\big)$	$\widetilde{O}(fe^f)$	randomized	Theorem 5

The derandomization by Karthik and Parter [29] of the original RPC requires setting up the error-correcting codes and deriving from the so-called hit-and-miss hash functions. We provide a much simpler algorithm to obtain a deterministic construction. Conceptually, we show that RPCs are amenable to the method of conditional expectations, avoiding heavy algebraic machinery. This hopefully makes our approach more applicable also to other tools for building fault-tolerant data structures. A disadvantage of our technique is that it requires small sensitivity. On the one hand, the best covering value and query time are achieved for $f = o(\log L)$. On the other hand, the derandomization itself requires access to the pre-computed answers to all $O(n^2 m^f)$ queries. (See Section 2 for more details.)

Besides the derandomization, another contribution of the work by Karthik and Parter [29] is a lower bound on the covering value of any (L, f)-replacement path covering. They showed that whenever $(L/f)^f \leqslant n$, there exists an n-vertex graph for which the family $\mathcal{G}$ must contain $\Omega(\,(L/f)^f)$ subgraphs. This leaves an $\widetilde{O}(f^{f+1})$ gap to the covering value in [35]. We aim at narrowing this gap, starting with a new lower bound.

▶ **Theorem 3.** *For all positive integers n, f, L, with $L \geqslant 2$, there is an n-vertex graph G s.t. any (L, f)-replacement path covering for G has covering value at least* $\min\Big\{\sum_{i=0}^{f-1}\binom{L-2}{i}, n\Big\}$.

Since the sum of binomial coefficients can be a bit unwieldy, we provide closed forms for certain ranges of f and L.

▶ **Corollary 4.** *Let the notation be the same as in Theorem 3.*

1. *If there exists a constant $\varepsilon > 0$ such that $2 \leqslant f \leqslant (1-\varepsilon)\frac{L}{2}$, then any (L, f)-replacement path covering must have covering value* $\Omega\Big(\min\Big\{\sqrt{f\,e^f}\,\frac{L^{f-1}}{f^f},\ n\Big\}\Big)$.
2. *If $f \geqslant \frac{L}{2}$, any (L, f)-replacement path covering must have covering value* $\Omega\big(\min\{2^L, n\}\big)$.

Compared to the lower bound in [29], the one in Corollary 4 (i) trades a factor L for $\sqrt{f\,e^f}$, which is asymptotically larger whenever $f \geqslant 2\ln L$. To also tackle the gap for $f = o(\log L)$, we improve the upper bound instead. Recall that in this range the $\widetilde{O}(fL^f)$ covering value by Weimann and Yuster [35] is still the best known. We decrease this by a factor $(f/e)^f$ by giving a tighter analysis of the construction algorithm of the (randomized) sampling trees by Bilò et al. [6] which also underpinned our derandomization.

▶ **Theorem 5.** *Let G be a graph (possibly directed and positively edge-weighted) with n vertices. Let f and L be two positive integers, which may depend on n, such that $f = o(\log L)$. There exists a randomized (L, f)-replacement path covering for G that with high probability has covering value $\widetilde{O}\big(fe^f(\frac{L}{f})^{f+o(1)}\big)$ and query time $\widetilde{O}\big(f^{\frac{5}{2}}e^f(\frac{L}{f})^{o(1)}\big)$. The size of the computed*

Table 2 Upper and lower bounds on the covering value of (L, f)-replacement path coverings. With $\varepsilon > 0$ we denote an arbitrarily small constant.

Covering Value	Parameter Range	Reference
$\widetilde{O}\big((\frac{L}{f})^f L^{o(1)}\big)$	$f = o(\log L)$	Theorem 5
$(\frac{L}{f})^f n^{o(1)}$	$f = o(\log n)$	Theorem 5
$\Omega\big((\frac{L}{f})^f\big)$		[29]
$\widetilde{O}(fL^f)$	$2\ln L \leqslant f \leqslant (1-\varepsilon)\frac{L}{2}$	[35]
$\Omega\big(\sqrt{fe^f}\,\frac{L^{f-1}}{f^f}\big)$		Corollary 4.1
$\Omega(2^L)$	$f = \Omega(\log n)$; $\frac{L}{2} \leqslant f$	Corollary 4.2

subfamily $\mathcal{G}_F$ is $\widetilde{O}(fe^f)$. The (L, f)-replacement path covering also supports vertex failures. For sensitivities up to $f = o(\log n)$, the covering value becomes $(\frac{L}{f})^f n^{o(1)}$, the query time and size of $\mathcal{G}_F$ is $n^{o(1)}$.

Table 2 summarizes the new and known upper and lower bounds. In the range $f = o(\log L)$, we shrink the gap between the bounds to $\widetilde{O}(f^{1-o(1)}e^f L^{o(1)}) = \widetilde{O}(L^{o(1)})$. For $\Omega(\log L) \leqslant f \leqslant (1-c)\frac{L}{2}$, that gap is now also smaller than before, but currently remains at $\widetilde{O}(f^{f+\frac{1}{2}}e^{-\frac{f}{2}}L)$. In the general literature on distance sensitivity oracles for non-constant values of f, the sensitivity is expressed in terms of n (while $L = L(n)$ is only an internal parameter). A common setting is $f = o(\log n/\log\log n)$, see e.g. [4, 12, 35]. Theorems 2 and 5 also apply to the larger range of $f = o(\log n)$. There, we get a gap of $n^{o(1)}L/\sqrt{fe^f}$. If additionally $f \geqslant 2\ln L$, then this becomes $n^{o(1)}$, showing that our construction is near-optimal.

We conjecture that the true covering value is of order $\Theta(e^f(\frac{L}{f})^f) \cdot \mathsf{poly}(f)$, whenever this is not larger than n. To show this, new constructions are needed for both the upper and lower bound. As a possibly more accessible open question, we ask whether one can achieve a *deterministic* (L, f)-replacement path covering with the same parameters as in Theorem 5.

Outline. The remainder of the paper is structured as follows. In Section 2, we provide an overview of the sampling tree construction and our derandomization, which is then carried out in Section 3. In Section 4, we present our new randomized RPC. We prove the lower bound on the covering value in Section 5.

2 Overview

Our (L, f)-replacement path covering and the associated query data structure leverage a hierarchical sampling strategy by Bilò et al. [6], which in turn generalized earlier work by Weimann and Yuster [35]. We briefly review the randomized construction before discussing the challenges of its derandomization and how we overcome them. Throughout this overview, unless stated otherwise, we assume $f = o(\log L)$ to simplify the exposition.

The sampling tree framework. The randomized RPC consists of a forest of sampling trees that are constructed in levels. Each tree has height h and branching factor $\alpha = L^{f/h}$. Its nodes represent subgraphs of G starting with the edge-less graph in the root. A child node

inherits all edges from its parent and re-inserts additional edges from G independently with probability $1 - p = 1 - L^{-1/h}$. After h levels, in each subgraph that is stored in a leaf, any edge of G is present with probability $1 - L^{-1}$, matching the original model [35]. However, the critical difference is that the leaves are *not* independent. Any two leaves share at least all edges of their lowest common ancestor.

This locality is essential for query efficiency. The query algorithm performs a depth-first search starting from the root and, in each step, is traversing to the first child that contains no edge of the given query set F. This runs in time $O(fh\alpha) = O(hL^{(f/h)+o(1)})$, which is dramatically faster than examining all $\widetilde{O}(L^f)$ subgraphs as in [35]. However, the straightforward search may fail to reach a suitable leaf in any given tree with probability $1 - C^h$ for some constant $C > 1$. By repeating this search in $K = \widetilde{O}(fC^h)$ independent trees, the RPC achieves high success probability over all $n^2\binom{m}{\leqslant f} = O(n^2 m^f)$ queries (s, t, F).

Challenges of the derandomization. At first glance, derandomizing this construction via the method of conditional expectations appears straightforward: replace each random decision to include an edge in any of the subgraphs with a greedy choice that maximizes conditional expectations. Unfortunately, this approach fundamentally breaks the query structure. The randomized construction succeeds because of the following two reasons.

1. The failures are independent across trees, allowing boosting via repetition.
2. The probability for the search to fail in a fixed tree depends only on the height h.

The greedy derandomization, basing decisions only on local information, runs the danger of correlating failures across trees. Even worse, the search looks for *any* child whose subgraph has no edge of F. This means that if we bias the tree structure to make some children succeed more often, we may cause the failures to concentrate in other parts of the tree, ruining the balance needed for the query algorithm.

Consider the collection $\mathcal{C}$ of pairs (P, F) where $F \subseteq E$ is a set of at most f failing edges and $P \subseteq E$ are the edges of a shortest path in $G-F$ with hop-length not larger than L. The set $\mathcal{C}$ may contain a pair for each possible query. Ideally, the derandomization procedure should maximize the number of such pairs that will *eventually* be covered by the leaves of *any* of the trees. This is not the same as maximizing the likelihood that the depth-first search finds *some* suitable leaf for a given pair. The local greedy optimization and the global query structure are misaligned.

Our derandomization technique. We resolve this through a novel derandomization framework that carefully balances the competing objectives. Let x be a node in one of the trees and $A_x \subseteq E$ those edges that are *missing* in the subgraph associated to x. If x is a leaf, it covers a pair $(P, F) \in \mathcal{C}$ if and only if $F \subseteq A_x$ and $P \subseteq E \backslash A_x$. As mentioned above, maximizing those pairs for which all failing edges are removed ($F \subseteq A_x$) might inadvertently increase the number of pairs for which $P \cap A_x$ is too large. This is reflected in our definition of well-separated and poorly-separated pairs.

- *Well-separated pairs* are those for which the failing edges F are already removed at node x, but the replacement path P remains largely intact meaning that $|P \cap A_x|$ is small.
- *Poorly-separated pairs* are those for which the removal has already gone too far, damaging the replacement path beyond recovery at the current depth of x.

Let w be the parent of the current node x. We have to decide which edges to re-insert in the associated subgraph in addition to the ones inherited from w. That means, we construct the set A_x as a subset of A_w. We can restrict our attention to those pairs (P, F) that have

$F \subseteq A_w$ and for which $F \nsubseteq A_{w'}$ holds for all siblings w' of x that are processed before x by the deterministic query algorithm. Let this be the set $\mathcal{D}_x$. It contains much fewer pairs than the full collection $\mathcal{C}$. Let further $X_{P,F,x}$ be the indicator variable whether the pair $(P,F) \in \mathcal{D}_x$ is well separated w.r.t. x and define $Y_{P,F,x}$ analogously for poorly-separated pairs. The latter pairs cannot be handled by the current tree and must be deferred to a later one. However, we do not want to increase the total number of trees beyond our budget K. So rather than greedily maximizing well-separated pairs alone, exacerbating the imbalance, we optimize the expectation of

$$\sum_{(P,F)\in\mathcal{D}_x} \left(X_{P,F,x} - \frac{1}{2} Y_{P,F,x} \right).$$

The expectation is taken over the original random sampling process conditioned on all previous decisions taken by the derandomization. The coefficient $1/2$ in the sum serves as a tie breaker. Even if the number of well and poorly-separated pairs were equal, we have $\mathbb{E}[\sum_{(P,F)}(X_{P,F,x} - \frac{1}{2}Y_{P,F,x})] = \frac{1}{2}\mathbb{E}[\sum_{(P,F)} X_{P,F,w}] > 0$ for the expectation. Thus, we do not lose significant ground for the well-separated pairs.

We prove that this weighted objective maintains the properties that are critical for the query algorithm. The worst-case failure behavior across trees remains approximately independent, even in the now fully deterministic construction. Also, by carefully choosing the depth-dependent threshold of what counts as "small"[4] for $|P \cap A_x|$, we retain the same success guarantee for an individual tree as in the randomized case.

Improving the covering value. As a secondary contribution beyond the derandomization, we also lower the upper bound on number of subgraphs. We observe that the prior randomized construction in [6] was not optimized with respect to the specific structure of the query algorithm. The fixed-order search through the children of the current node gets stuck only when *all* children contain some edge of F. This observation allows us to use lower branching factors α and smaller height h, resulting in fewer leaves per tree.

However, these changes also affect the success probability in other ways. For example, fewer child nodes being available in each transition increase the failure probability. We provide a sharper analysis of the hierarchical sampling process of the trees to prove that the reduction of α and h can be counter-acted by a lower re-insertion probability $1-p$ and a slightly larger number of trees K, while still maintaining the properties of an RPC with high probability. This improves the covering value of the sampling tree framework by a factor f^f from $\widetilde{O}(L^{f+o(1)})$ to $\widetilde{O}((L/f)^{f+o(1)})$.

A new lower bound. Another contribution is an improved lower bound on the covering value of any (L,f)-replacement path covering (deterministic or randomized). Prior to our work, the only lower bound was $\Omega(\min\{(L/f), n\})$ by Karthik and Parter [29]. It already shows that our $\widetilde{O}((L/f)^{f+o(1)})$ construction for $f = o(\log L)$ is near-optimal. However, the bound leaves a gap for larger sensitivities. We improve it significantly by constructing a family of weighted directed acyclic graphs that demonstrates that $\Omega(\min\{\sqrt{fe^f}\, L^{f-1}/f^f, n\})$ subgraphs are necessary.

At a high level, it consists of a directed binary tree T (not related to the sampling trees). It is constructed from what we call *inner trees*. These are comb-like structures consisting of a rooted path in which each node has an attached leaf. The edge weights are set up in

[4] The precise meaning of small turns out to be $|P \cap A_x| \leqslant p^d L$, where d is the depth of x in the tree and p is such that $p^f = 1/\alpha$ for the branching factor α.

such a way that traveling along the back of the comb is for free while the leaf that is farthest from the root in hop-distance is actually the closest in weighted distance. The inner trees are iteratively combined to form the binary tree T by replacing the leaves of an inner tree with new appropriately scaled-down inner trees. The scaling is such that after f rounds, all leaves of the whole tree T have the same hop-distance L from its root. The leaves of T are all connected to a single sink node v.

For each such leaf x, we define a failure set F_x of at most f edges so that removing those edges makes x the leaf that is closest to the root in weighted distance. In other words, the unique shortest replacement path in $G-F_x$ from the root to the sink v goes through x. This creates a barrier for small (L, f)-replacement path coverings since any RPC for G must contain a different subgraph for every leaf. A combinatorial argument then shows that there are at least $\sum_{i=0}^{f-1} \binom{L-2}{i}$ such leaves.

3 Deterministic Replacement Path Coverings

Let $\mathcal{C}$ be a collection of pairs (P, F) of disjoint sets of edges, satisfying $|P| \leqslant L$ and $|F| \leqslant f$. Here, P corresponds to a shortest path between a source-target pair in $G-F$. Our goal is to cover each pair $(P, F) \in \mathcal{C}$, meaning, in at least one tree we construct, there is a leaf x such that the associated subgraph G_x contains no edge of F but all edges of P. Moreover, the query algorithm that starts in the root of that tree and always recurses into the first child that does not have an edge of F reaches x. The remaining notation follows [6]. We remark, however, that we slightly adjust the parameters to our needs. (They thus also differ from the values reported in the first part of the overview in Section 2.) Nevertheless, this will lead to the same asymptotic covering value and query time.

3.1 Preliminaries

For a positive integer ℓ, we use $[\ell]$ to denote the set $\{1, 2, \ldots, \ell\}$. Our data structure consists of K disjoint rooted trees $\{T_i\}_{i\in[K]}$, whose nodes all represent a spanning subgraph of G. Each sampling tree has height h and any internal node has exactly α children. The parameters K, h, and α will be optimized later. A single tree has α^h leaves and $O(\alpha^h)$ nodes in total. We associate with each node x in a tree T_i a set $A_x \subseteq E$. Namely, x represents the graph $G_x = G-A_x$. The sets A_x is computed deterministically. However, let us first consider the following random construction. If x is a root, we simply set $A_x = E$. Now let y be a child of x, its set $A_y \subseteq A_x$ is obtained by selecting each edge in A_x independently with probability $p = \alpha^{-1/f}$. For our derandomization, it will also be important to adjust the parameters such that $p^h L < 1$ holds. The random construction is iterated until height h. All random choices are made independently. The total number of stored subgraphs is $O(K\alpha^h)$ and the RPC is given by the family $\mathcal{G}$ of all subgraphs that are stored in the leaves of all the K sampling trees. The following lemma summarizes the structural properties of the randomized construction.

▶ **Lemma 6** (Lemma 6 in [6]). *Let $i \in [K]$ and x be a node of the tree T_i at depth d. For any edge $e \in E$, the probability of it being included in A_x is $\mathbb{P}[e \in A_x] = p^d$. For any other edge $e' \in E$, $e' \neq e$, the events $[e \in A_x]$ and $[e' \in A_x]$ are independent.*

Given a failure set $F \subseteq E$ with $|F| \leqslant f$, Algorithm 1 computes the subfamily $\mathcal{G}_F$. Recall that the set $A_x \subseteq E$ contains those edges that are *removed* from the subgraph G_x of the node x. The trees $T_1, \ldots, T_K$ are searched individually, starting in the respective roots. In each step, the algorithm always chooses the *first* child node y of the current node x that

Algorithm 1 Query algorithm on input F.

```
1 G_F ← ∅;
2 for i = 1 to K do
3     x ← root of T_i;
4     while x is not a leaf do
5         Let y be the first child of x satisfying F ⊆ A_y. If no such child exists,
6         then continue to outer for-loop;
7         x ← y;
      /* Node x is a leaf at depth h such that F ⊆ A_x.                    */
8     G_F ← G_F ∪ {G_x};
```

satisfies $F \subseteq A_y$. If no such child exists, the next tree is searched. If eventually a leaf is reached, the subgraph stored there is added to $\mathcal{G}_F$. Observe that no graph in $\mathcal{G}_F$ contains a failing edge from F since it is explicitly checked before recursing to y. Up to K subgraphs are collected in time $O(fK\alpha h)$.

3.2 Well-Separated Pairs

We classify the pairs $(P, F) \in \mathcal{C}$ with respect to their relevance to a given node x in a tree. Recall that we want that the graph G_x associated with x contains no edge of F. If so, the pair (P, F) is *active* for x. Additionally, the number of edges of the path P that are missing in x should be "small", where the exact amount depends on the depth of x in the tree. If so, the pair is *well separated*. If a pair is active for x, but still too many path edges are missing from G_x, then the pair is *poorly separated*.

▶ **Definition 7** (well-separated, poorly-separated, active, and passive pairs). *Let $i \in [K]$ and x be a node of the tree T_i at depth d. Let further $(P, F) \in \mathcal{C}$ be a failure set-path pair.*

1. *(P, F) is* well separated *with respect to x if $F \subseteq A_x$ and $|P \cap A_x| \leqslant p^d L$.*
2. *(P, F) is* poorly separated *with respect to x if $F \subseteq A_x$ and $|P \cap A_x| > p^d L$.*
3. *(P, F) is* passive *with respect to x if $F \nsubseteq A_x$; otherwise, it is* active *w.r.t. x.*

We also say that the node x is active *for (P, F), if the pair (P, F) is active w.r.t. x.*

We use the different categories of pairs as follows. Let $(P, F) \in \mathcal{C}$ be a pair, where the path P has endpoints $s, t \in V$. Algorithm 1 traverses a path from the root to a leaf in the tree T_i by always selecting, at each node x, the first child y of x with $F \subseteq A_y$, that is, the first child node that is *active* for (P, F). We claim that, in order for this traversal to succeed for the query (s, t, F) in T_i, it is enough to maintain the following property.

If y is the first child node of x that is active for (P, F),
then (P, F) is also well separated with respect to y.

To see this, recall that the probability parameter p satisfies $p^h L < 1$. Along the path traced by Algorithm 1, the fraction of edges of P contained in the associated sets A_x geometrically decreases. In the leaf at depth $d = h$, we have both $F \subseteq A_x$ as well as $|P \cap A_x| \leqslant p^h L < 1$ and thus $P \cap A_x = \emptyset$. In other words, $G_x = G - A_x$ contains no edge of F but all of P.

Algorithm 2 Derandomization of the Sampling Trees $T_1, \ldots, T_K$.

Let $T_1, \ldots, T_K$ be the collection of (random) sampling trees rooted at nodes $r_1, \ldots, r_K$;
for $i = 1$ **to** K **do**
  $\mathcal{D}_{r_i} \leftarrow \mathcal{C} \setminus \bigcup\{\mathcal{D}_x \mid x \text{ is a leaf in } T_{i_0} \text{ for some } i_0 < i\}$;
  Derandomize-Subtree$(T_i, r_i, \mathcal{D}_{r_i})$;

To track the well and poorly-separated pairs, we define the following binary variables. Let x be a node in some tree T_i and let $(P, F) \in \mathcal{C}$ be a pair that is active w.r.t. x.

$$X_{P,F,x} = \begin{cases} 1 & \text{if } (P,F) \text{ is well separated w.r.t. } x; \\ 0 & \text{otherwise.} \end{cases}$$

$$Y_{P,F,x} = \begin{cases} 1 & \text{if } (P,F) \text{ is poorly separated w.r.t. } x; \\ 0 & \text{otherwise.} \end{cases}$$

Let $r_1, r_2, \ldots, r_K$ denote the respective roots (at depth $d = 0$) of $T_1, T_2, \ldots, T_K$. At least for the r_i, we do not have to worry about separation, namely, $X_{P,F,r_i} \equiv 1$. The follows immediately from the definition of well-separated pairs and the roots representing empty graphs. This sets up a starting point both for the derandomization of the tree T_i and later for the query algorithm.

▶ **Observation 8.** *For $i \in [K]$, all pairs in $\mathcal{C}$ are well separated with respect to the root of T_i.*

During the derandomization, we have to keep track of the pairs in $\mathcal{C}$ that are covered by the trees we have already processed.

▶ **Definition 9** (handled pairs)**.** *For any index $i \in [K]$, a pair $(P, F) \in \mathcal{C}$ is* handled *by the tree T_i if (P, F) is well separated with respect to every node in the root-to-leaf path in T_i that is traced by Algorithm 1 with query set F.*

▶ **Definition 10** (the set $\mathcal{D}_x$)**.** *Let $i \in [K]$ and x be a node of the tree T_i. The set $\mathcal{D}_x \subseteq \mathcal{C}$ contains those pairs (P, F) that satisfy the following properties.*

1. *(P, F) is not handled by any of the trees $T_1, \ldots, T_{i-1}$.*
2. *The path in the tree T_i that Algorithm 1 traces on input F leads from the root r_i to x.*
3. *(P, F) is a well-separated pair with respect to each node in T_i on the path from r_i to x.*

3.3 Derandomization Algorithm

Algorithm 2 outlines the recursive derandomization procedure for the trees $T_1, \ldots, T_K$. The recursive procedure *Derandomize-Subtree* considers a node x in the tree T_i. It goes through the children $y_1, \ldots, y_\alpha$ in the same (arbitrary) order in which they are processed at query time by Algorithm 1. Suppose that the current child node is y_j, meaning that x as well as $y_1, \ldots, y_{j-1}$ are already derandomized. It has access to the deterministic edge sets $A_x, A_{y_1}, \ldots, A_{y_{j-1}}$ and needs to compute A_{y_j}. Let $\mathcal{D}_x \subseteq \mathcal{C}$ be the set of pairs described in Definition 10 and consider the following subset.

$$\mathcal{D} = \{(P, F) \in \mathcal{D}_x \mid (P, F) \text{ is passive with respect to } y_{j_0} \text{ for some } j_0 < j\}.$$

Intuitively, the pairs in $\mathcal{D}$ are those for which the query algorithm starting in r_i has not only reached node x, but will also not recurse in any of its child nodes before checking y_j.

Procedure Derandomize-Subtree$(T, x, \mathcal{D}_x)$.

Let $y_1, \dots, y_\alpha$ be the child nodes of x in T;
for $j = 1$ **to** α **do**
  $\mathcal{D} \leftarrow \{(P,F) \in \mathcal{D}_x \mid (P,F)$ is passive w.r.t. y_k for every $k < j\}$;
  derandomize the set A_{y_j} edge by edge so that the conditional expectations of $\sum_{(P,F)\in\mathcal{D}} \left(X_{P,F,y_j} - \frac{1}{2}Y_{P,F,y_j}\right)$ is maximized in each step;
  $\mathcal{D}_{y_j} \leftarrow \{(P,F) \in \mathcal{D} \mid (P,F)$ is well separated w.r.t. $y_j\}$;
  if y_j *is not a leaf* **then** Derandomize-Subtree$(T, y_j, \mathcal{D}_{y_j})$;

We aim to maximize the number of pairs in $\mathcal{D}$ that are well separated with respect to y_j, while at the same time, we do not want too many pairs from $\mathcal{D}$ to become poorly separated with respect to y_j. It is crucial to strike a balance here, as the poorly-separated pairs may not be handled by T_i and may thus unduly increase the number of trees required. For each edge $e \in A_x$, we decide one by one whether to include it in A_{y_j}. To make this decision, we compare the respective expected values of

$$\sum_{(P,F)\in\mathcal{D}} \left(X_{P,F,y_j} - \frac{1}{2}Y_{P,F,y_j}\right)$$

conditional on the event $[e \in A_{y_j}]$ and on $[e \notin A_{y_j}]$. The expectation is taken over the random process that just removes any edge $e' \in A_x \backslash \{e\}$ for which we have not made a decision yet independently with probability p. Finally, we set $\mathcal{D}_{y_j}$ to be those pairs in $\mathcal{D}$ that are now well separated with respect to y_j. Note that this indeed satisfies Definition 10 for node y_i.

3.4 Correctness and Analysis

In order to prove that our approach indeed yields a deterministic (L, f)-replacement path covering, we need to assign values to the parameters. We leave the total number K of trees open until the end of this section, and instead start by fixing the branching factor to $\alpha = (2L)^{\frac{f}{h}}$ as well as the the sampling probability to $p = (2L)^{-\frac{1}{h}}$. Both parameters depend on the height h, which will also be set later. For now, it is enough to observe that the definitions ensure both $p^f = \frac{1}{\alpha}$ as well as $p^h L = \frac{1}{2} < 1$.

Let x be a non-leaf node in any of the trees and let y be one of its child nodes. Recall the random construction in which the edge set A_y is obtained from A_x by sampling any edge $e \in A_x$ independently with probability p. (See Subsection 3.1 for more details.) The following lemma describes how this random construction behaves with respect to the passive, well-separated, and poorly-separated pairs.

▶ **Lemma 11.** *Let $i \in [K]$ be an index, x a non-leaf node in the tree T_i, $(P,F) \in \mathcal{C}$ a well-separated pair w.r.t. x, and y a child node of x. Let $\mathbb{P}_1 = \mathbb{P}[(P,F)$ is passive w.r.t. $y]$, $\mathbb{P}_2 = \mathbb{P}[(P,F)$ is well separated w.r.t. $y]$, and $\mathbb{P}_2 = \mathbb{P}[(P,F)$ is poorly separated w.r.t. $y]$.*

1. $\mathbb{P}_1 \leqslant 1 - \frac{1}{\alpha}$; **2.** $\mathbb{P}_2 \geqslant \frac{1}{2}(1 - \mathbb{P}_1)$; **3.** $\mathbb{P}_3 \leqslant \frac{1}{2}(1 - \mathbb{P}_1)$.

Proof. Let $d \in [h]$ be the depth of the child node y. Since (P,F) is well separated w.r.t. x, we have $F \subseteq A_x$ and $|P \cap A_x| \leqslant p^{d-1}L$. Each element of A_x lies in A_y with probability p, so the probability of (P,F) being passive at y is $\mathbb{P}_1 = \mathbb{P}[F \not\subseteq A_y] = 1 - p^{|F|} \leqslant 1 - p^f = 1 - \frac{1}{\alpha}$.

We now calculate the probability $\mathbb{P}_2$ of (P,F) being a well-separated pair w.r.t. y, meaning that both $F \subseteq A_y$ and $|P \cap A_y| \leqslant p^d L$ hold. Note that the two conditions are independent (see Lemma 6) and we have $\mathbb{P}[F \subseteq A_y] = (1-\mathbb{P}_1)$. So we only need to estimate $\mathbb{P}\big[|P \cap A_y| \leqslant p^d L\big]$. Observe that $|P \cap A_y|$ follows the binomial distribution $\mathrm{Bin}(|P \cap A_x|, p)$ and thus has median $p \cdot |P \cap A_x| \leqslant p^d L$. Therefore, we get

$$\mathbb{P}\Big[|P \cap A_y| \leqslant p^d L\Big] \geqslant \mathbb{P}\Big[|P \cap A_y| \leqslant p \cdot |P \cap A_x|\Big] = \frac{1}{2}.$$

This gives $\mathbb{P}_2 = \mathbb{P}\big[|P \cap A_y| \leqslant p^d L\big] \cdot \mathbb{P}[F \subseteq A_y] \geqslant \frac{1}{2}(1-\mathbb{P}_1)$. The estimate for $\mathbb{P}_3$ follows immediately from $\mathbb{P}_1 + \mathbb{P}_2 + \mathbb{P}_3 = 1$. ◀

Finally, we can use the randomized construction to prove properties of its derandomization.

▶ **Lemma 12.** *Let $i \in [K]$ and x a non-leaf node in the tree T_i with children $y_1, \dots, y_\alpha$. The sets $\mathcal{D}_{y_1}, \dots, \mathcal{D}_{y_\alpha}$ computed by* Derandomize-Subtree$(T_i, x, \mathcal{D}_x)$ *satisfy $\sum_{j=1}^{\alpha} |\mathcal{D}_{y_j}| \geqslant \frac{2}{11}|\mathcal{D}_x|$.*

Proof. Fix a child y_j. Let $\mathcal{D} = \{(P,F) \in \mathcal{D}_x \mid (P,F) \text{ is passive w.r.t. } y_{j_0} \text{ for some } j_0 < j\}$. We process the derandomization of A_{y_j} so that the value of $\sum_{(P,F)\in\mathcal{D}} \left(X_{P,F,y_j} - \frac{1}{2}Y_{P,F,y_j}\right)$ is at least its expectation. Fix a pair $(P,F) \in \mathcal{D}$. Due to $\mathcal{D} \subseteq \mathcal{D}_x$ this pair is well-separated w.r.t. the node x, so we can apply Lemma 11. Let $\mathbb{P}_1$, $\mathbb{P}_2$, and $\mathbb{P}_3$ be defined as above for the child node $y = y_i$. We then have $\mathbb{E}[X_{P,F,y_j}] = \mathbb{P}_2 \geqslant \frac{1}{2}(1-\mathbb{P}_1)$ as well as $\mathbb{E}[Y_{P,F,y_j}] = \mathbb{P}_3 \leqslant \frac{1}{2}(1-\mathbb{P}_1)$. This implies that the (expected and thus deterministic) value of that sum is at least

$$\sum_{(P,F)\in\mathcal{D}} \left(X_{P,F,y_j} - \frac{1}{2}\cdot Y_{P,F,y_j}\right) \geqslant |\mathcal{D}| \left(\frac{1}{2}(1-\mathbb{P}_1) - \frac{1}{4}(1-\mathbb{P}_1)\right) \geqslant \frac{|\mathcal{D}|}{4}(1-\mathbb{P}_1) \geqslant \frac{|\mathcal{D}|}{4\alpha}. \quad (1)$$

For simplicity, assume an idealized scenario where no pair in $\mathcal{D}_x$ ever becomes poorly separated with respect to y_j, for any $j \in [\alpha]$. On the one hand, this means $Y_{P,F,y_j} = 0$ and thus the estimate in (1) states that at least $|\mathcal{D}|/4\alpha$ pairs become well separated in one iteration. On the other hand, the only way to remain passive w.r.t. y_j is to *not* become well separated. line 3 and 5 of the Derandomize-Subtree procedure then ensure that $\mathcal{D} = \mathcal{D}_x \setminus \bigcup_{k=1}^{j-1} \mathcal{D}_{y_k}$ holds in the j-th iteration. Since the $\mathcal{D}_{y_k}$ are pairwise disjoint, this yields $|\mathcal{D}_{y_j}| \geqslant \frac{|\mathcal{D}_x| - \sum_{k=1}^{j-1}|\mathcal{D}_{y_k}|}{4\alpha}$. Summing over all iterations gives

$$\sum_{j=1}^{\alpha} |\mathcal{D}_{y_j}| \geqslant \alpha \cdot \frac{|D_x|}{4\alpha} - \sum_{j=1}^{\alpha}\sum_{k=1}^{j-1} \frac{|D_{y_k}|}{4\alpha} = \frac{|D_x|}{4} - \frac{1}{4}\sum_{j=1}^{\alpha}(\alpha - j)\frac{|D_{y_j}|}{\alpha} \geqslant \frac{|D_x|}{4} - \frac{1}{4}\sum_{j=1}^{\alpha}|D_{y_j}|,$$

which is solved by $\sum_{j=1}^{\alpha} |\mathcal{D}_{y_j}| \geqslant |\mathcal{D}_x|/5$.

Now let $\mathcal{P}_j \subseteq \mathcal{D}_x$ be the pairs that become poorly separated w.r.t. y_j. They do not contribute to $\mathcal{D}_{y_j}$ and also do not get included into $\mathcal{D}$ at the beginning of the next iteration. This worsens the lower bound for the individual terms to $|\mathcal{D}_{y_j}| \geqslant \frac{|\mathcal{D}_x| - \sum_{k=1}^{j-1}|\mathcal{D}_{y_k}| - \sum_{k=1}^{j-1}|\mathcal{P}_k|}{4\alpha}$. However, the estimate (1) being non-negative implies that for any two pairs that become poorly separated with respect to y_j, at least one more pair must have become well separated, i.e., $|\mathcal{D}_{y_j}|/2 \geqslant |\mathcal{P}_j|$. The same argument as above gives $\sum_{j=1}^{\alpha} |\mathcal{D}_{y_j}| \geqslant 2|\mathcal{D}_x|/11$. ◀

As a direct corollary of the lemma, we get that the fraction of pairs not yet handled by the trees $T_1, \dots, T_{i-1}$ that get handled by T_i is at least $(2/11)^h$. Slightly abusing notation, let $\mathcal{D}_{T_i} = \bigcup\{\mathcal{D}_x \mid x \text{ is a leaf in } T_i\}$. The number of pairs handled by the K trees is at least

$$\sum_{i=1}^{K} |\mathcal{D}_{T_i}| \geqslant |\mathcal{C}| \cdot \left(\frac{2}{11}\right)^h \sum_{i=1}^{K} \left(1 - \left(\frac{2}{11}\right)^h\right)^{i-1} = |\mathcal{C}| \cdot \left(1 - \left(1 - \left(\frac{2}{11}\right)^h\right)^K\right).$$

Since $\sum_{i=1}^{K} |\mathcal{D}_{T_i}|$ is integral, choosing K large enough so that the lower bound is strictly larger than $|\mathcal{C}| - 1$ is sufficient for the sum to exhaust all $|\mathcal{C}|$ pairs. This condition simplifies to $(1 - (\frac{2}{11})^h)^K < \frac{1}{|\mathcal{C}|}$ and is satisfied, e.g., by $K = 2(\frac{11}{2})^h \ln |\mathcal{C}| = O((\frac{11}{2})^h f \log n)$ since

$$\left(1 - \left(\frac{2}{11}\right)^h\right)^K \leqslant \exp\left(-\left(\frac{2}{11}\right)^h \cdot K\right) = \exp(-2\ln|\mathcal{C}|) = \frac{1}{|\mathcal{C}|^2} < \frac{1}{|\mathcal{C}|}.$$

We can now calculate the covering value and query time depending on f, L, n, and h.

- The size of family $\mathcal{G}$ is $O(K \cdot L^f) = O((\frac{11}{2})^h \cdot fL^f \log n)$.
- The query time to compute $\mathcal{G}_F$ is $O(f\alpha hK) = O((\frac{11}{2})^h h \cdot f^2 L^{f/h} \log n)$.
- For any set F, the size of family $\mathcal{G}_F$ is at most $K = O((\frac{11}{2})^h \cdot f \log n)$.

We finally choose $h = \sqrt{f \log_2 L}$. For $f = o(\log L)$, this ensures $\left(\frac{11}{2}\right)^h = L^{\frac{\log_2\left(\frac{11}{2}\right)}{\log_2 L} \cdot h} = L^{\log_2\left(\frac{11}{2}\right)\sqrt{\frac{f}{\log_2 L}}} = L^{o(1)}$ as well as $L^{\frac{f}{h}} = L^{\sqrt{\frac{f}{\log_2 L}}} = L^{o(1)}$. Finally, we obtain the parameters stated in Theorem 2.

- The covering value is $O((\frac{11}{2})^h \cdot fL^f \log n) = O(fL^{f+o(1)} \log n)$.
- The query time is $O((\frac{11}{2})^h h \cdot f^2 L^{f/h} \log n) = O(f^{5/2} L^{o(1)} \log n)$.
- The size of $\mathcal{G}_F$ is $O((\frac{11}{2})^h \cdot f \log n) = O(fL^{o(1)} \log n)$.

For higher sensitivities up to $f = o(\log n)$ the $L^{o(1)}$ terms turn to $n^{o(1)}$ since $\left(\frac{11}{2}\right)^h = n^{\frac{\log_2\left(\frac{11}{2}\right)}{\log_2 n} \cdot h} = n^{\log_2\left(\frac{11}{2}\right) \cdot \frac{\sqrt{f \log_2 L}}{\log_2 n}} = n^{o(1)}$ and $L^{\frac{f}{h}} = n^{\frac{\log_2 L}{\log_2 n} \cdot \frac{f}{h}} = n^{\frac{\sqrt{f \log_2 L}}{\log_2 n}} = n^{o(1)}$.

4 Randomized Replacement Path Coverings

We now improve the upper bound on the covering value of (randomized) (L, f)-replacement path coverings for the parameter range $f = o(\log L)$. We build on the sampling trees by Bilò et al. [6], reviewed in Subsection 3.1. Like before, each of the K trees has height h and any internal node has exactly α children. We also reuse Algorithm 1 to process queries. The main difference is that we give a tighter analysis of the parameters of the construction which allows us to derive better bounds. In the following, we let $P(s, t, F)$ denote a *replacement path* from s to t, that is, a shortest s-t-path in $G - F$. If there are multiple such paths, we take one with the minimum number of edges. Recall that for any node x in a sampling tree, we use $G_x \subseteq G$ for the spanning subgraph associated with x.

▶ **Lemma 13** (Lemma 7 in [6]). *Let $i \in [K]$ be an index, $F \subseteq E$ a set of $|F| \leqslant f$ edges, and $s, t \in V$ vertices such that $P(s, t, F)$ contains at most L edges.*

1. *Algorithm 1 reaches a leaf of the sampling tree T_i with probab. at least $((1 - (1 - p^f)^\alpha)^h$.*
2. *If Algorithm 1 reaches a leaf x of the tree T_i, then the probability of the path $P(s, t, F)$ existing in G_x is at least $(1 - p^h)^L$.*

The total number of nodes is $O(K\alpha^h)$ and the query algorithm selects $|\mathcal{G}_F| \leqslant K$ leaf nodes in time $O(fK\alpha h)$. The probability to reach a leaf of a tree is exponentially small in h (Lemma 13.1). However, once a leaf is reached, the probability of the stored graph holding the relevant path $P(s, t, F)$ *grows* with h, depending on p. As before, we need to cover (with high probability) all pairs of vertices $s, t \in V$ for which $P(s, t, F)$ has at most L edges. Let $\delta > 0$ be a sufficiently large constant. Diverging from [6], we choose the parameters

- $h = \sqrt{f \ln(L/f)}$
- $K = \delta e^f (\frac{e}{e-1})^h f \ln n$
- $\alpha = (L/f)^{f/h}$
- $p = (f/L)^{1/h}$

Lemma 6 states that in any leaf x, at depth h, the probability for any edge to be removed (event $[e \in A_x]$) is $p^h = f/L$. We verify next that the query algorithm indeed computes a suitable collection of graphs. The lemma also implies that the graphs stored in the leaves of the trees form an (L, f)-replacement path covering with high probability.

▶ **Lemma 14.** *W.h.p. over all sets $F \subseteq E$ with $|F| \leqslant f$ and $s, t \in V$ with $|E(P(s,t,F))| \leqslant L$, upon termination of Algorithm 1, there exists a graph $G_x \in \mathcal{G}_F$ that retains $P(s,t,F)$.*

Proof. By Lemma 13, the probability to reach a leaf x in some tree T_i whose corresponding graph G_x contains the replacement path $P(s,t,F)$ is at least $((1-(1-p^f)^\alpha)^h \cdot (1-p^h)^L$. We insert the parameters into the first factor and apply a lower bound of the form $(1+\frac{t}{k})^k \geqslant (1-\frac{t^2}{k})e^t$ for $k \geqslant 1$ and $k \geqslant |t|$. It can, for example, be found in the textbook by Motwani and Raghavan [30, p. 435]. Recall that we assume $f = o(\log L)$; for now, it will be sufficient that $2f^2 \leqslant L$. Regarding the second factor, we have $(1-p^h)^L = \left(1-\frac{f}{L}\right)^L \geqslant \left(1-\frac{f^2}{L}\right) \cdot \frac{1}{e^f} \geqslant \frac{1}{2e^f}$ Further, observe that $\left(1-\left(1-p^f\right)^\alpha\right)^h = \left(1-\left(1-\frac{1}{(L/f)^{f/h}}\right)^{(L/f)^{f/h}}\right)^h \geqslant \left(1-\frac{1}{e}\right)^h$. Let C abbreviate $\frac{e}{e-1} \approx 1.582$. We have thus shown that the probability is at least $1/(2e^f C^h)$.

Repeating the query in $K = \delta e^f C^h f \ln n$ independent trees reduces the failure probability for any triple (s,t,F) to $\left(1-\frac{1}{2e^f C^h}\right)^{\delta e^f C^h f \ln n} \leqslant \left(\frac{1}{e}\right)^{\frac{\delta}{2} \ln n} = n^{-\frac{\delta f}{2}}$. There are at most $|V^2 \times \binom{E}{\leqslant f}| = O(n^{2+2f})$ relevant triples (s,t,F), which shows that choosing δ large enough and taking a union bound over all triples ensures a high success probability. ◀

Note that $C^h = C^{\sqrt{f \ln(L/f)}} = \left(\left(\frac{L}{f}\right)^{\frac{\ln C}{\ln(L/f)}}\right)^{\sqrt{f \ln(L/f)}} = \left(\frac{L}{f}\right)^{\ln C \sqrt{\frac{f}{\ln(L/f)}}} = \left(\frac{L}{f}\right)^{o(1)}$. The last estimate uses that $f = o(\log L)$ also gives $f = o(\ln(L/f))$. Similarly, we have

$$\alpha = \left(\frac{L}{f}\right)^{f/h} = \left(\frac{L}{f}\right)^{\sqrt{f/\ln(L/f)}} = \left(\frac{L}{f}\right)^{o(1)}.$$

Our choice of parameters thus shows that the whole data structure stores

$$O(K\alpha^h) = O(fe^f C^h (L/f)^f \ln n) = O(fe^f (L/f)^{f+o(1)} \ln n)$$

graphs and has a query time of $O(fKh\alpha) = O(f^{\frac{5}{2}} e^f (L/f)^{o(1)} \sqrt{\ln(L/f)} \ln n)$.

The size of the subfamily $\mathcal{G}_F$ is $K = O(fe^f (L/f)^{o(1)} \ln n)$ in the worst case. However, with high probability the number of subgraphs relevant to a query F is smaller. In Lemma 14, we showed that in any tree T_i the probability of finding a leaf node x satisfying $F \subseteq A_x$ is $(1-\frac{1}{e})^h = 1/C^h$. Thus, the expected size of $\mathcal{G}_F$ is $K/C^h = O(fe^f \log n)$. Further recall that each T_i adds a graph to $\mathcal{G}_F$ independently of other trees. Applying a standard Chernoff bound shows that also with high probability $\mathcal{G}_F$ contains $O(fe^f \log n)$ graphs.

5 Lower Bound

We now establish the lower bound on the covering value. Slightly abusing notation, we show that there exists families of graphs for which any $(L+1, f)$-replacement path covering must contain at least $\sum_{i=0}^{f-1} \binom{L-1}{i}$ subgraphs to fulfill the requirements of Definition 1. That means, we set denote the cut-off parameter as $L+1$ (instead of L) to ease the notation below. Our construction is assembled from building blocks which we call *inner trees*. These are

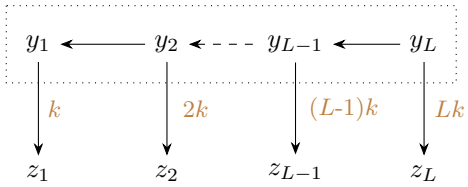

Figure 1 Depiction of an (L, k)-inner tree.

comb-like structures which are connected by iteratively replacing the leafs of one inner tree with new, smaller inner trees. The sizes are arranged in such a way that in the emerging binary tree all leaves have the same distance L from the root.

5.1 Inner Trees

We fix a sensitivity f and cutoff-value $L+1$ for the rest of this subsection. For a positive integer k, an (L, k)-*inner tree* is constructed from a directed path $P = (y_L, \dots, y_2, y_1)$ with $L-1$ edges. The path is rooted at node y_L. For each $i \in [L]$, we add a directed edge from y_i to a new leaf node z_i. The weight of the edge (y_i, z_i) is set to $i \cdot k$. The edges in P all have weight zero. See Figure 1 for an illustration.

We construct a binary tree T in f rounds, numbered from 0 to $f-1$. In the 0-th round, we initialize T as an (L, L^{2f})-inner tree. Its root serves as the root of the whole tree T. In each subsequent round j, we replace each leaf in T at hop-distance d from the root with an $(L{-}d, L^{2(f-j)})$-inner tree. Note that at the end of round $f-1$, every root-to-leaf path in the resulting tree has L edges. See Figure 2.

For any $j \in [f]$, let $N(L, j)$ denote the number of leaf nodes in the tree T at level j. Let further $R(L, j)$ be the number of j-tuples $(r_1, \dots, r_j)$ of positive integers $r_1, \dots, r_j \geqslant 1$ that sum to exactly L. Then, it holds that $R(L, j) = \binom{L-1}{j-1}$. Observe that $N(L, j) \geqslant R(L, j)$. This is because each distinct tuple $(r_1, \dots, r_j)$ maps to a unique leaf of tree T at level j. The mapping is obtained by traversing r_1 hops in the first inner tree, r_2-hop-length in the second inner tree, and so on. Therefore, $N(L, j) \geqslant \binom{L-1}{j-1}$. This shows that the total number of leaf nodes in tree T is lower bounded from below by $\sum_{j=1}^{f} \binom{L-1}{j-1}$.

Let s denote the root of T and X the set of leaves. We extend T to a digraph G by taking a new sink v and adding an edge from each leaf node in X to v of weight 1. We demonstrate that for any $(L{+}1, f)$-replacement path covering for G the size of the family $\mathcal{G}$ is at least $|X|$.

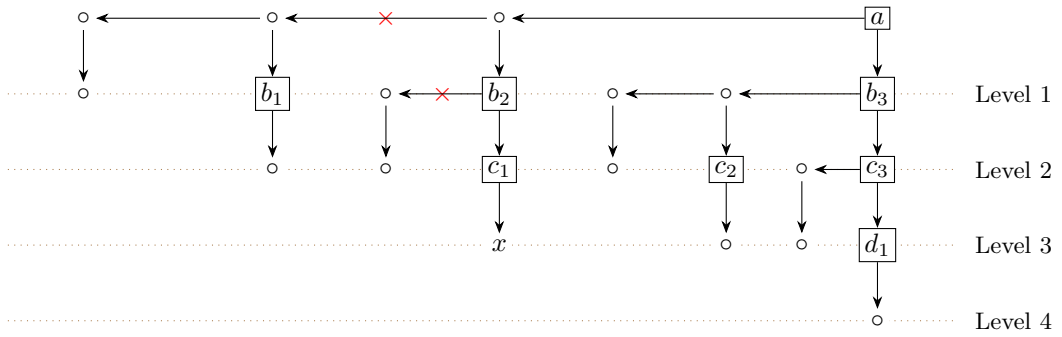

Figure 2 Depiction of the tree T for parameters $L = 4$ and $f = 3$. The edges marked with a cross are the set F_x for the leaf node x at level 3.

Fix some $x \in X$, we define a set $F_x \subseteq E(G)$ of at most f edges as follows. Let $\tau_1, \ldots, \tau_\ell$ be the inner trees that are traversed by the path from s to x in the tree T. For each $i \in [\ell]$, let y_i be the ancestor of x that lies in τ_i and is furthest from s. The set F_x is then formed by taking those out-edges of the nodes $y_1, \ldots, y_\ell$ that do *not* lie on the unique path from s to x in T. Let $T[s,x]$ denote this path. Clearly, we have $|F_x| \leqslant \ell \leqslant f$. Moreover, the edge weights are so that traveling inside one level is free. It is always preferred to descend to the next level as far from the root of the current inner tree as possible. Therefore, after the failures, $T[s,x]$ is the cheapest way to reach a leaf of T, whence the concatenation $T[s,x] \circ (x,v)$ is the unique shortest path from s to v in the graph $G - F_x$. The path has $L+1$ edges.

For each leaf $x \in X$ of T, let $G_x \in \mathcal{G}$ be a graph that avoids the edges in F_x but still contains the replacement path $T[s,x] \circ (x,v)$. For any two distinct $x, x' \in X$, we assert that $G_x \neq G_{x'}$. This is due to the fact that if G_x and $G_{x'}$ were the same, both would include the out-edges of the lowest common ancestor $w = \mathrm{LCA}_T(x,x')$. However, this is a contradiction since at least one of the out-edges of w is included in F_x or $F_{x'}$. From this, we get our lower bound on the covering value $|\mathcal{G}| \geqslant |X| \geqslant \sum_{j=1}^{f} \binom{L-1}{j-1}$, proving Theorem 3.

5.2 Proof of Corollary 4

When adjusting the cut-off value back to L (from $L+1$) and parameterizing the binomial coefficient in terms of $0 \leqslant i \leqslant f-1$ instead of $j \in [f]$, we get the sum $\sum_{i=0}^{f-1} \binom{L-2}{i}$. We give a closed form for it below to prove Corollary 4.

▶ **Corollary 4.** *Let the notation be the same as in Theorem 3.*

1. *If there exists a constant $\varepsilon > 0$ such that $2 \leqslant f \leqslant (1-\varepsilon)\frac{L}{2}$, then any (L,f)-replacement path covering must have covering value $\Omega\Big(\min\Big\{\sqrt{f\,e^f}\,\frac{L^{f-1}}{f^f},\ n\Big\}\Big)$.*
2. *If $f \geqslant \frac{L}{2}$, any (L,f)-replacement path covering must have covering value $\Omega\big(\min\{2^L, n\}\big)$.*

The second clause is immediate from the observation that $f \geqslant \frac{L}{2}$ (that is, $f-1 \geqslant \frac{L-2}{2}$) implies that summing the first $f-1$ binomial coefficients gives at least half the value of the sum over all $L-2$ terms.

$$\sum_{i=0}^{f-1} \binom{L-2}{i} \geqslant \frac{1}{2} \cdot \sum_{i=0}^{L-2} \binom{L-2}{i} = \frac{1}{2} \cdot 2^{L-2} = \Omega(2^L).$$

In the rest of the section, we prove the first clause. If f is bounded away from $L/2$, meaning $f \leqslant (1-\varepsilon)\frac{L}{2}$ for some positive constant $\varepsilon > 0$, the binomial coefficient $\binom{L-2}{f-1}$ is the largest term of the sum. We bound that term using Stirling's approximation. The following lemma can, for example, be found in the textbook by Cover and Thomas [16].

▶ **Lemma 15** (Stirling's approximation). *Let k be a positive integer and $0 < x < 1$ a rational number such that xk is integral. Then, it holds that*

$$\binom{k}{xk} \geqslant \frac{1}{\sqrt{8kx(1-x)}} \left(\frac{1}{x}\right)^{xk} \left(\frac{1}{1-x}\right)^{(1-x)k}.$$

We apply Lemma 15 with $k = L-2$ and $xk = f-1$, where we use that $f \geqslant 2$ and thus $x = (f-1)/(L-2) > 0$. We lower bound the three factors on the right-hand side separately.

$$\left(\frac{1}{x}\right)^{xk} = \left(\frac{L-2}{f-1}\right)^{f-1} = \left(\frac{L-2}{L}\right)^{f-1} \left(\frac{L}{f-1}\right)^{f-1}.$$

It holds that $(\frac{L-2}{L})^{f-1} \geqslant (\frac{L-2}{L})^{\frac{L}{2}} \geqslant (\frac{3}{4})^2$. For the last estimate, we use that $(\frac{L-2}{L})^{\frac{L}{2}}$ is increasing

in L and that $L \geqslant 4$ due to $\frac{L}{2}(1-\varepsilon) \geqslant f \geqslant 2$. This proves $x^{-xk} = \Omega\big((\frac{L}{f-1})^{f-1}\big)$.

Next, observe that for $x < 1$, we have $(1-x)^{-1} \geqslant e^x$, which can be derived from the arguably more common inequality $1 - x \leqslant e^{-x}$. Since $f \leqslant L/2$, we get $1 - x = \frac{L-f-1}{L-2} \geqslant \frac{1}{2}$. Therefore, it holds that $\left(\frac{1}{1-x}\right)^{(1-x)k} \geqslant \exp(x \cdot (1-x)k) \geqslant \exp\left(\frac{f-1}{2}\right) = \Omega\big(\sqrt{e^f}\big)$. Finally, we have $(8kx(1-x))^{-1/2} \geqslant (4xk)^{-1/2} = \Omega\left(\frac{1}{\sqrt{f-1}}\right)$.

Combining the three estimates results in

$$\begin{aligned}\binom{L-2}{f-1} &= \Omega\left(\frac{1}{\sqrt{f-1}}\left(\frac{L}{f-1}\right)^{f-1}\sqrt{e^f}\right)\\ &= \Omega\left(\sqrt{(f-1)\,e^f}\cdot\frac{L^{f-1}}{(f-1)^f}\right) = \Omega\left(\sqrt{fe^f}\cdot\frac{L^{f-1}}{f^f}\right).\end{aligned}$$

References

1. Josh Alman and Dean Hirsch. Parameterized Sensitivity Oracles and Dynamic Algorithms Using Exterior Algebras. In *Proceedings of the 49th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 9:1–9:19, 2022. doi:10.4230/LIPIcs.ICALP.2022.9.
2. Noga Alon, Shiri Chechik, and Sarel Cohen. Deterministic Combinatorial Replacement Paths and Distance Sensitivity Oracles. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming, (ICALP)*, pages 12:1–12:14, 2019. doi:10.4230/LIPIcs.ICALP.2019.12.
3. Davide Bilò, Shiri Chechik, Keerti Choudhary, Sarel Cohen, Tobias Friedrich, Simon Krogmann, and Martin Schirneck. Approximate Distance Sensitivity Oracles in Subquadratic Space. *TheoretiCS*, 3:15:1–15:47, 2024. doi:10.46298/theoretics.24.15.
4. Davide Bilò, Shiri Chechik, Keerti Choudhary, Sarel Cohen, Tobias Friedrich, and Martin Schirneck. Improved Distance (Sensitivity) Oracles with Subquadratic Space. In *Proceedings of the 65th Symposium on Foundations of Computer Science (FOCS)*, pages 1550–1558, 2024. doi:10.1109/FOCS61266.2024.00097.
5. Davide Bilò, Keerti Choudhary, Sarel Cohen, Tobias Friedrich, Simon Krogmann, and Martin Schirneck. Fault-Tolerant ST-Diameter Oracles. In *Proceedings of the 50th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 24:1–24:20, 2023. doi:10.4230/LIPIcs.ICALP.2023.24.
6. Davide Bilò, Keerti Choudhary, Sarel Cohen, Tobias Friedrich, and Martin Schirneck. Efficient Fault-Tolerant Search by Fast Indexing of Sub-Networks. In *Proceedings of the 39th AAAI Conference on Artificial Intelligence (AAAI)*, pages 26463–26471, 2025. doi:10.1609/AAAI.V39I25.34846.
7. Davide Bilò, Katrin Casel, Keerti Choudhary, Sarel Cohen, Tobias Friedrich, J.A. Gregor Lagodzinski, Martin Schirneck, and Simon Wietheger. Fixed-Parameter Sensitivity Oracles. In *Proceedings of the 13th Innovations in Theoretical Computer Science Conference (ITCS)*, pages 23:1–23:18, 2022. doi:10.4230/LIPIcs.ITCS.2022.23.
8. Davide Bilò, Keerti Choudhary, Sarel Cohen, Tobias Friedrich, and Martin Schirneck. Deterministic Sensitivity Oracles for Diameter, Eccentricities and All Pairs Distances. In *Proceedings of the 49th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 22:1–22:19, 2022. doi:10.4230/LIPIcs.ICALP.2022.22.
9. Gilad Braunschvig, Shiri Chechik, David Peleg, and Adam Sealfon. Fault Tolerant Additive and (μ, α)-Spanners. *Theoretical Computer Science*, 580:94–100, 2015. doi:10.1016/J.TCS.2015.02.036.

10 Diptarka Chakraborty and Keerti Choudhary. New Extremal Bounds for Reachability and Strong-Connectivity Preservers Under Failures. In *Proceedings of the 47th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 25:1–25:20, 2020. doi:10.4230/LIPIcs.ICALP.2020.25.

11 Shiri Chechik and Sarel Cohen. Distance Sensitivity Oracles with Subcubic Preprocessing Time and Fast Query Time. In *Proceedings of the 52nd Symposium on Theory of Computing (STOC)*, pages 1375–1388, 2020. doi:10.1145/3357713.3384253.

12 Shiri Chechik, Sarel Cohen, Amos Fiat, and Haim Kaplan. $(1+\varepsilon)$-Approximate f-Sensitive Distance Oracles. In *Proceedings of the 28th Symposium on Discrete Algorithms (SODA)*, pages 1479–1496, 2017. doi:10.1137/1.9781611974782.96.

13 Shiri Chechik, Michael Langberg, David Peleg, and Liam Roditty. f-Sensitivity Distance Oracles and Routing Schemes. *Algorithmica*, 63:861–882, 2012. doi:10.1007/s00453-011-9543-0.

14 Kyungjin Cho, Jihun Shin, and Eunjin Oh. Approximate Distance Oracle for Fault-Tolerant Geometric Spanners. In *Proceedings of the 38th AAAI Conference on Artificial Intelligence (AAAI)*, pages 20087–20095, 2024. doi:10.1609/aaai.v38i18.29987.

15 Julia Chuzhoy, Merav Parter, and Zihan Tan. On Packing Low-Diameter Spanning Trees. In *Proceedings of the 47th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 33:1–33:18, 2020. doi:10.4230/LIPIcs.ICALP.2020.33.

16 Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley Series in Telecommunications and Signal Processing. Wiley-Interscience, New York City, NY, USA, 2nd edition, 2006. doi:10.1002/047174882X.

17 Camil Demetrescu, Mikkel Thorup, Rezaul A. Chowdhury, and Vijaya Ramachandran. Oracles for Distances Avoiding a Failed Node or Link. *SIAM Journal on Computing*, 37:1299–1318, 2008. doi:10.1137/S0097539705429847.

18 Dipan Dey and Manoj Gupta. Nearly Optimal Fault Tolerant Distance Oracle. In *Proceedings of the 56th Symposium on Theory of Computing (STOC)*, pages 944–955, 2024. doi:10.1145/3618260.3649697.

19 Michael Dinitz and Robert Krauthgamer. Fault-Tolerant Spanners: Better and Simpler. In *Proceedings of the 30th Symposium on Principles of Distributed Computing (PODC)*, pages 169–178, 2011. doi:10.1145/1993806.1993830.

20 Michael Dinitz and Caleb Robelle. Efficient and Simple Algorithms for Fault-Tolerant Spanners. In *Proceedings of the 39th Symposium on Principles of Distributed Computing (PODC)*, pages 493–500, 2020. doi:10.1145/3382734.3405735.

21 Ran Duan and Seth Pettie. Dual-Failure Distance and Connectivity Oracles. In *Proceedings of the 20th Symposium on Discrete Algorithms (SODA)*, pages 506–515, 2009. doi:10.1137/1.9781611973068.56.

22 Ran Duan and Seth Pettie. Connectivity Oracles for Failure Prone Graphs. In *Proceedings of the 42nd Symposium on Theory of Computing (STOC)*, pages 465–474, 2010. doi:10.1145/1806689.1806754.

23 Ran Duan and Seth Pettie. Connectivity Oracles for Graphs Subject to Vertex Failures. In *Proceedings of the 28th Symposium on Discrete Algorithms (SODA)*, pages 490–509, 2017. doi:10.1137/17M1146610.

24 Ran Duan and Hanlin Ren. Maintaining Exact Distances under Multiple Edge Failures. In *Proceedings of the 54th Symposium on Theory of Computing (STOC)*, pages 1093–1101, 2022. doi:10.1145/3519935.3520002.

25 Fabrizio Grandoni and Virginia Vassilevska Williams. Faster Replacement Paths and Distance Sensitivity Oracles. *ACM Transaction on Algorithms*, 16:15:1–15:25, 2020. doi:10.1145/3365835.

26 Yong Gu and Hanlin Ren. Constructing a Distance Sensitivity Oracle in $O(n^{2.5794}M)$ Time. In *Proceedings of the 48th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 76:1–76:20, 2021. doi:10.4230/LIPIcs.ICALP.2021.76.

27 Monika Henzinger, Andrea Lincoln, Stefan Neumann, and Virginia Vassilevska Williams. Conditional Hardness for Sensitivity Problems. In *Proceedings of the 8th Conference on Innovations in Theoretical Computer Science (ITCS)*, pages 26:1–26:31, 2017. doi:10.4230/LIPIcs.ITCS.2017.26.

28 Yael Hitron and Merav Parter. Broadcast CONGEST Algorithms against Adversarial Edges. In *Proceedings of the 35th Symposium on Distributed Computing (DISC)*, pages 23:1–23:19, 2021. doi:10.4230/LIPIcs.DISC.2021.23.

29 Karthik C. S. and Merav Parter. Deterministic Replacement Path Covering. *ACM Transactions on Algorithms*, 20:34:1–34:35, 2024. doi:10.1145/3673760.

30 Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, Cambridge, NY, USA, 1995. doi:10.1017/CBO9780511814075.

31 Merav Parter. Small Cuts and Connectivity Certificates: A Fault Tolerant Approach. In *Proceedings of the 33rd Symposium on Distributed Computing (DISC)*, pages 30:1–30:16, 2019. doi:10.4230/LIPIcs.DISC.2019.30.

32 Merav Parter and Eylon Yogev. Low Congestion Cycle Covers and Their Applications. In *Proceedings of the 30th Symposium on Discrete Algorithms (SODA)*, pages 1673–1692, 2019. doi:10.1137/1.9781611975482.101.

33 Merav Parter and Eylon Yogev. Secure Distributed Computing Made (Nearly) Optimal. In *Proceedings of the 38th Symposium on Principles of Distributed Computing (PODC)*, pages 107–116, 2019. doi:10.1145/3293611.3331620.

34 Mihai Patrascu and Mikkel Thorup. Planning for Fast Connectivity Updates. In *Proceedings of the 48th Symposium on Foundations of Computer Science (FOCS)*, pages 263–271, 2007. doi:10.1109/FOCS.2007.59.

35 Oren Weimann and Raphael Yuster. Replacement Paths and Distance Sensitivity Oracles via Fast Matrix Multiplication. *ACM Transactions on Algorithms*, 9:14:1–14:13, 2013. doi:10.1145/2438645.2438646.