Epic Fail: Emulators Can Tolerate Polynomially Many Edge Faults for Free

Greg Bodwin ✉
University of Michigan, Ann Arbor, MI, USA

Michael Dinitz ✉
Johns Hopkins University, Baltimore, MD, USA

Yasamin Nazari ✉
Universität Salzburg, Austria

─── **Abstract** ───

A t-emulator of a graph G is a graph H that approximates its pairwise shortest path distances up to multiplicative t error. We study fault tolerant t-emulators, under the model recently introduced by Bodwin, Dinitz, and Nazari [ITCS 2022] for *vertex failures*. In this paper we consider the version for *edge failures*, and show that they exhibit surprisingly different behavior.

In particular, our main result is that, for $(2k - 1)$-emulators with k odd, we can tolerate a polynomial number of edge faults *for free*. For example: for any n-node input graph, we construct a 5-emulator ($k = 3$) on $O(n^{4/3})$ edges that is robust to $f = O(n^{2/9})$ edge faults. It is well known that $\Omega(n^{4/3})$ edges are necessary even if the 5-emulator does not need to tolerate *any* faults. Thus we pay no extra cost in the size to gain this fault tolerance. We leave open the precise range of free fault tolerance for odd k, and whether a similar phenomenon can be proved for even k.

1 Introduction

A basic question in theoretical computer science is to sparsify an input graph G while approximately preserving its shortest path distances. These sparsifiers have applications in networking, algorithms, and more (see survey [1]). We will discuss two important types of sparsifiers: spanners and emulators.

▶ **Definition 1** (Spanners). *For an input graph $G = (V, E, w)$, a t-spanner is a subgraph H such that for all $u, v \in V$, we have $dist_G(u, v) \le dist_H(u, v) \le t \cdot dist_G(u, v)$. The parameter t is called the stretch of the spanner.*

▶ **Definition 2** (Emulators). *For an input graph $G = (V, E, w)$, a t-emulator is a graph $H = (V, E_H)$ with the following property. If we set the weight of each edge $(u, v) \in E_H$ to be $dist_G(u, v)$, then for all $u, v \in V$, we have $dist_G(u, v) \le dist_H(u, v) \le t \cdot dist_G(u, v)$.*

Elsewhere in the literature, emulators are often defined slightly differently, as arbitrary weighted graphs $H = (V, E_H, w)$ that approximate distances of G. However, once we choose the edge set E_H, the choice of edge weights is clear: for an emulator edge (u, v), we must choose $w(u, v) \ge dist_G(u, v)$ to satisfy the first inequality in Definition 2, and there is no

advantage to choosing $w(u,v)$ strictly larger than $\text{dist}_G(u,v)$.[1] Hence, this optimal choice of edge weights is hardcoded into Definition 2. (This hardcoding will be convenient in our discussion of fault tolerant emulators below.) We also note that, for both spanners and emulators, the first inequality $\text{dist}_G(u,v) \leq \text{dist}_H(u,v)$ is trivially implied by the fact that H is a subgraph of G (for spanners) or the way the weights are set (for emulators).

Spanners were first introduced by [15, 16] in the context of distributed computing, and they have since found many uses throughout algorithms.

Emulators were first introduced in the context of shortest path algorithms [12], but they have close connections to distributed computing as well, e.g. they form a reasonable model of overlay networks [7]. A priori, one could imagine that the additional flexibility of emulators allows for a better size/stretch tradeoff than the one available to spanners. But this possibility was refuted in a classic 1993 work of Althöfer, Das, Dobkin, Joseph, and Soares, which determined the same (up to logs) tradeoff for both spanners and emulators:

▶ **Theorem 3** ([2])**.** *For any positive integer k, every n-node graph has a $(2k-1)$-**spanner** on $O(n^{1+1/k})$ edges. Moreover, assuming the girth conjecture [13], this is essentially best possible in the sense that there are n-node graphs that do not admit $(2k-1)$-**emulators** on $o(n^{1+1/k}/\log n)$ edges.*

1.1   Fault-Tolerance

Distributed computing often deals with networks in which vertices or edges can temporarily fail. Thus, when spanners and emulators are applied in distributed contexts, it is natural to ask for *fault-tolerant* versions of these sparsifiers. The precise definitions are deferred to Section 1.1.1, but at a high level we say that a spanner or emulator H is f-fault tolerant if, for every set of at most f failing parts (edges or vertices), what remains of H is still a spanner/emulator of what remains of G. This leads to a clear question: what is the "price" of fault-tolerance? That is, if we want to build an f-fault-tolerant spanner or emulator with stretch $2k-1$, what factor (depending on f) must we pay in the sparsity bound, above the baseline size of $O(n^{1+1/k})$ determined by Theorem 3 for the non-fault tolerant version?

There has been a fruitful line of work tackling this question for spanners and emulators (e.g., [9, 10, 5, 8, 11, 6, 7, 4, 14]). This has produced optimal bounds on the price of fault tolerance in some settings, distinctions between edge and vertex faults, and a suite of algorithmic techniques that allow for efficient computation of sparse fault tolerant spanners/emulators (including in distributed and parallel models of computation). In particular, following [9, 10], results in [5, 7, 4] have established that for edge/vertex fault tolerant spanners and for vertex fault tolerant emulators, the price of fault tolerance is a sublinear polynomial in f. Such a dependence can still be expensive for the important scenario where f is large, i.e. a (sublinear) polynomial in the number of nodes n.

This paper is the first to investigate the remaining setting of edge fault tolerant emulators, and we show a quantitative size bound that is surprisingly different from the ones for these related objects. Our main result is that there is a parameter regime $f \leq \text{poly}(n)$ in which the price of edge fault tolerance for emulators is *free*. That is, for fixed odd k, we construct $(2k-1)$-emulators that can handle f edge faults and only have $O(n^{1+1/k})$ edges, the same size bound settled by Theorem 3 for the non-faulty setting.

---

[1] We mean here that there is no advantage in terms of quality of distance approximation. One could imagine an advantage to choosing larger weights for other reasons, e.g., perhaps we want to compute the emulator quickly and so we use a lossy approximation algorithm to set weights. But these concerns will not arise in this paper.

1.1.1 Fault-Tolerance Definitions and Previous Results

We now define the objects that we study and reference. For general graph spanners, fault-tolerance was initially studied by Chechik, Langberg, Peleg, and Roditty [9], who introduced the following definition:

▶ **Definition 4** (EFT Spanners). *For an input graph G, an f-edge fault tolerant (EFT) t-spanner is a subgraph H of G such that $dist_{G \setminus F}(u, v) \leq dist_{H \setminus F}(u, v) \leq t \cdot dist_{G \setminus F}(u, v)$ for all $F \subseteq E$ with $|F| \leq f$ and for all $u, v \in V$.*

Equivalently, $H \setminus F$ must be a t-spanner of $G \setminus F$ for all possible fault sets F. Vertex fault-tolerance is defined analogously, with F as a vertex set rather than an edge set.

After significant work following [9] (see in particular [5, 8, 11, 6, 7]), we now have a generally good understanding of both vertex- and edge-fault tolerant spanners. Precise bounds are given in Table 1, but the high-level view is that the price of fault-tolerance is $f^{1-1/k}$ for vertex fault tolerant spanners, and even smaller for edge fault tolerant spanners (approximately $f^{1/2}$, although the exact bounds remain open). With fault-tolerant spanners relatively well understood, fault-tolerant *emulators* were first studied by Bodwin, Dinitz, and Nazari [4]. They studied emulators under vertex failures, but their model under edge failures is the following:

▶ **Definition 5** (EFT Emulators). *For an input graph $G = (V, E)$, we say that $H = (V, E')$ is an f-edge fault tolerant (EFT) t-emulator H if, for all $F \subseteq E$ with $|F| \leq f$, H is a t-emulator of $G \setminus F$.*

Recall that Definition 2 hardcodes the edge weights of emulators, and thus this definition specifically means that H approximates the distances of $G \setminus F$ *when the edge weights of H are set according to distances in $G \setminus F$*. That is, the following definition of EFT emulators is equivalent to the above. Given a set $F \subseteq E$, let $H^F = (V, E', w_F)$ be the weighted version of H where the weight of each edge $(u, v) \in E'$ is set to $w_F(u, v) = \text{dist}_{G \setminus F}(u, v)$. Then an f-EFT t-emulator must satisfy

$$\text{dist}_{G \setminus F}(u, v) \leq \text{dist}_{H^F}(u, v) \leq t \cdot \text{dist}_{G \setminus F}(u, v)$$

for all $F \subseteq E$ with $|F| \leq f$ and $u, v \in V$.

Intuitively, this means a set of edge failures F does not change the *edge set* of H at all; rather, its effect on H is *updated edge weights* due to increasing distances in $G \setminus F$. Note that even when an edge e is in both the emulator and the fault set, it remains as an edge in H^F; it just has a larger weight than it does in the pre-fault graph $H^\emptyset$.

This model may seem too strong: why shouldn't failures in G affect the edges left in H, and why should the edges of H automatically get reweighted to the shortest path in $G \setminus F$?

It turns out that this weight-updating behavior arises naturally in computer networking and distributed computing, particularly in overlay networks, which use virtual edges that inherit their weights from the routing topology of the underlying graph. When pieces of the underlying graph fail, the virtual edges remain intact, but the underlying routing algorithm will automatically update to avoid failures and so the inherited lengths of the virtual edges will reconverge to the new shortest paths. We refer the interested reader to [4] for further practical and theoretical justification for this definition, and we also overview this discussion ourselves in the full version [3].

■ **Table 1** Price of fault tolerance for edge and vertex fault tolerant spanners and emulators, proved in [5, 8, 4, 7], including the new bounds proved in this paper (lower right). All objects also have a lower bound of $\Omega(fn)$ edges, not reflected in this table. We note that the upper bound of $O(f^{1/2})$ for EFT emulators when $k = 2$ is *not* proved in this paper. Rather, this is inherited from the other objects in this table; e.g., it follows from the definitions that any EFT spanner is also an EFT emulator.

	$(2k-1)$ Spanners	$(2k-1)$ Emulators
Vertex Faults	$\Theta\left(f^{1-\frac{1}{k}}\right)$	$\Theta\left(f^{\frac{1}{2}}\right)$ if $k = 2$, $\widetilde{\Theta}_k\left(f^{\frac{1}{2}-\frac{1}{2k}}\right)$ if k is odd, $\widetilde{O}_k\left(f^{\frac{1}{2}}\right)$ and $\Omega_k\left(f^{\frac{1}{2}-\frac{1}{2k}}\right)$ if $k \geq 4$ is even
Edge Faults	$\Theta\left(f^{\frac{1}{2}}\right)$ if $k = 2$, $\Theta_k\left(f^{\frac{1}{2}-\frac{1}{2k}}\right)$ if k is odd, $O_k\left(f^{\frac{1}{2}}\right)$ and $\Omega\left(f^{\frac{1}{2}-\frac{1}{2k}}\right)$ if $k \geq 4$ is even	$\Theta\left(f^{\frac{1}{2}}\right)$ if $k = 2$, $O\left(1 + f^{\frac{3}{5}}n^{-\frac{2}{15}}\right)$ if $k = 3$, $O_k\left(1 + f^{\frac{k}{k+2}}n^{-\frac{2}{k(k+2)}} + f^{\frac{1}{2}}n^{-\frac{1}{k(k+1)}}\right)$ if k is odd, $\Theta(1)$ if $f = O\left(n^{\frac{2}{9}}\right), k = 3$ or $f = O\left(n^{\frac{2}{k(k+1)}}\right), k$ odd

1.2   Our Results

We show that EFT emulators allow for free fault-tolerance in a polynomial range of f. Our results, together with previous work on fault-tolerant spanners and emulators, are summarized in Table 1. We will first state our main result in the special case $k = 3$, for simplicity and because for technical reasons our bound is slightly better here than for larger k.

▶ **Theorem 6** (Main Result, $k = 3$). *Every n-node graph has an f-EFT 5-emulator H satisfying*

$$|E(H)| = O\left(n^{4/3} + n^{6/5}f^{3/5} + nf\right).$$

▶ **Corollary 7.** *For any n-node graph and parameter $f = O(n^{2/9})$, there is an f-EFT 5-emulator H satisfying $|E(H)| = O(n^{4/3})$.*

Recall that there are graphs that require $\Omega(n^{4/3})$ edges even for a non-fault-tolerant $(f = 0)$ 5-emulator (see Theorem 3). Thus in the parameter range $f = O(n^{2/9})$, the price of edge fault tolerance for 5-emulators is $\Theta(1)$, i.e., *free*. We show a similar but slightly worse bound for general odd k:

▶ **Theorem 8** (Main Result). *For any fixed odd k, every n-node graph has an f-EFT $(2k-1)$-emulator H satisfying*

$$|E(H)| = O_k\left(n^{1+\frac{1}{k}} + n^{1+\frac{1}{k+2}}f^{\frac{k}{k+2}} + n^{1+\frac{1}{k+1}}f^{\frac{1}{2}} + fn\right).$$

Since our theorem assumes k is fixed (a constant independent of n and f) we use O_k notation to hide factors that depend only on k. We do not carefully track or try to optimize these factors, but it is not hard to see that they are at most polynomial in k.

This theorem implies the following range of free fault tolerance:

▶ **Corollary 9.** *For any fixed odd k, n-node graph, and parameter $f = O(n^{2/(k(k+1))})$, there is an f-EFT $(2k-1)$-emulator H satisfying $|E(H)| = O_k(n^{1+1/k})$.*

Recall that, assuming the girth conjecture, there are graphs which require $\Omega(n^{1+1/k})$ edges even for a non-fault tolerant $(2k-1)$-emulator. Thus we can build emulators that are robust to $O(n^{2/(k(k+1))})$ faults for free.

The exact reason why our main result holds only for odd k is rather technical, and it is discussed in more detail at the beginning of Section 6. Very roughly, our strategy to upper bound emulator density is based on counting a certain kind of k-path in the emulator (see our proof overview in Section 2). Our counting strategy produces a better bound when there exists an edge exactly in the middle of these k-paths, which is the case if and only if k is odd. This is a similar technical issue to the one that causes inherently different upper bounds for even/odd k in the settings of edge fault tolerant spanners [7] and vertex fault tolerant emulators [4]; thus, this paper adds to an emerging theme that problems in this area seem harder to understand when k is even.

We prove all our upper bounds constructively, via an algorithm which constructs EFT emulators of the claimed size. While the "main" algorithm we analyze does not run in polynomial time (the obvious implementation of it would take time at least $\Omega(n^f)$), in Section 7 we show a variant that runs in polynomial time, paying an additional $O(k)$ factor in emulator size. This algorithm is based on ideas from [11].

At a technical level, our size analysis requires a change in strategy from prior work. The (near-optimal) size bounds in prior work for edge fault tolerant spanners and for vertex fault tolerant emulators happen to both be achieved by the method of analyzing *alternating paths* [7, 4]; hence, this method has roughly been pushed to its limit. In order to *simultaneously* benefit from the focus on edge rather than vertex faults *and* from the focus on emulators rather than spanners, we need a different approach entirely, which we overview in Section 2.

On the lower bounds side, we prove two limitations on the quality of EFT emulators. Both bounds are based on constructions that have been repeatedly used in prior work [9, 5, 7], but which need to be reanalyzed in the setting of EFT emulators.

▶ **Theorem 10** (Tight Bounds for $k = 2$). *Every n-node graph has an f-EFT 3-emulator H satisfying $|E(H)| = O(n^{3/2}f^{1/2})$. This is best possible, in the sense that there exist graphs for which any f-EFT 3-emulator H has $|E(H)| = \Omega(n^{3/2}f^{1/2})$*

The upper bound in Theorem 10 is inherited directly from prior work [4], while the lower bound analysis is new in this paper. A consequence of this theorem is that EFT 3-emulators do *not* confer free fault tolerance in any nontrivial range of parameters. We also have:

▶ **Theorem 11.** *For any k, there are n-node graphs for which any f-EFT $(2k-1)$-emulator has $\Omega(fn)$ edges.*

This lower bound is simple to prove and follows by considering any $f/2$-regular graph. It implies that, for any k, the maximum conceivable range of free fault tolerance is $f = O(n^{1/k})$.

1.3 Open Problems

The main open question left by this work is to pin down the range of f in which EFT emulators exhibit free fault tolerance. There are two main sub-questions. The first, and perhaps most pressing, is to determine whether free fault tolerance exists for even $k \geq 4$:

▶ **Open Question 12** (Is There Free Fault Tolerance for Even k?). *Is there any even constant $k \geq 4$ and constant $c > 0$ such that, for every n-node graph and parameter $f = O(n^c)$, there is an f-EFT $(2k-1)$-emulator H satisfying $|E(H)| = O(n^{1+1/k})$?*

Our current arguments do not confer *any* advantage for even k, relative to the size bounds inherited from EFT spanners and VFT emulators. The technical barrier to improved bounds for even k is discussed briefly in Section 6. While it seems entirely plausible to apply our techniques more carefully and obtain improved edge bounds, we think that a significant new idea will be needed to prove (or refute) *free* fault tolerance for any even k in any polynomial range of f.

The second question is to determine the *range* in which free fault tolerance exists for odd k:

▶ **Open Question 13** (What is the Range of Free Fault Tolerance for Odd k?)**.** *For odd k, what is the largest constant c_k such that, for every n-node graph and parameter $f = O(n^{c_k})$, there is an f-EFT $(2k-1)$-emulator H satisfying $|E(H)| = O(n^{1+1/k})$?*

For this second question, we show that $\Omega\left(\frac{1}{k^2}\right) \leq c_k \leq \frac{1}{k}$. It would be interesting to asymptotically improve the lower bound on c_k, improve the upper bound on c_k at all, or settle the value of c_3 (we show that $2/9 \leq c_3 \leq 1/3$). It would also be interesting to obtain "nearly-free" fault tolerance, e.g., subpolynomial price of fault tolerance in a polynomial range of f.

2   Technical Overview and Outline

We begin with a high-level overview of our algorithm and proof strategy for constructing f-EFT $(2k-1)$-emulators.

2.1   Algorithm

Algorithmically, we will do something simple that may nevertheless be surprising: we will run the obvious fault-tolerant variant of the greedy algorithm of Althöfer et al. [2]. That is, we will consider the edges of G in nondecreasing weight order, and when considering an edge (u,v) we will add it to our emulator if it is "necessary". In the traditional non-fault tolerant setting, "necessary" means that $\text{dist}_H(u,v) > (2k-1) \cdot w(u,v)$, i.e., the current spanner/emulator does not adequately preserve the distance between the endpoints u,v. In the fault-tolerant spanner setting, where the greedy algorithm has also been used and analyzed [5, 8, 7], "necessary" means that there is some appropriate fault set F such that $\text{dist}_{H \setminus F}(u,v) > (2k-1) \cdot w(u,v)$ (i.e., the current solution does not adequately preserve the distance between the endpoints under fault set F). Analogously, in our algorithm, "necessary" will mean that there is some fault set $F \subseteq E$ with $|F| \leq f$ (and not containing (u,v)) such that $\text{dist}_{H^F}(u,v) > (2k-1) \cdot w(u,v)$.

Although this fault-tolerant greedy algorithm is standard in prior work, the reason it is surprising in our setting is that the emulator H that we construct is actually a subgraph of the input graph G! We note this does *not* mean that H is a fault tolerant spanner of G.

In particular, consider what happens if there is some edge $e = (u,v) \in E$ which we include in our emulator H, and then e fails (is part of a fault set F). In a fault-tolerant spanner, this would mean that e also fails in the spanner: we would need $H \setminus F$ to be a spanner of $G \setminus F$, and so e would be removed from both G and H. But in a fault-tolerant emulator, e still exists in H^F: it just gets reweighted to $\text{dist}_{G \setminus F}(u,v)$, and hence becomes more costly to use. In other words, the relevant distance comparisons occur between H^F and $G \setminus F$, and H^F is not typically a subgraph of $G \setminus F$. Thus even though our emulator H consists only of edges that are also in G, we strongly use the fact that we are considering emulators rather than spanners.

2.2   Analysis

The classical analysis of the non-fault tolerant greedy spanner algorithm works as follows.

- First, one argues that H has *high girth*: in particular, all cycles in H will have $> 2k$ edges. This is a fairly straightforward consequence of the greedy algorithm.
- The focus of the proof then shifts from the particular output graph of the greedy algorithm to the more general class of graphs with high girth. One applies a counting argument from extremal combinatorics called the *Moore bounds*, stating that any n-node graph H of girth $> 2k$ can only have $O(n^{1+1/k})$ edges. The Moore bounds are a counting argument over the *non-backtracking k-paths* of H, and they are proved in two steps:
  - (Counting Lemma) Letting d be the average degree in H, and assuming d is at least a large enough constant, we prove that H has $n \cdot \Omega(d)^k$ non-backtracking k-paths.
  - (Dispersion Lemma) Say that a *path meet* is a pair of distinct k-paths with the same endpoints. We then prove that any n-node graph of girth $> 2k$ cannot have any meets between two non-backtracking k-paths.

The Moore bounds now follow roughly by arguing that the average degree d must be only $O(n^{1/k})$, or else H will have so many non-backtracking k-paths that (by the pigeonhole principle) two of them would share endpoints and thus form a meet.

In fault-tolerant spanners and emulators, it is unfortunately not true that the output graph H has high girth. However, one might intuitively believe that H "almost" has high girth, and that this might be enough to prove some type of sparsity bound. This proof strategy was first formalized by [8], who introduced a notion of a "blocking set": informally, a collection of *pairs* (either (edge, edge) pairs or (node,edge) pairs) that hits all cycles of length at most $2k$. Intuitively, a graph that has a small blocking set "almost" has girth larger than $2k$, and so it is sometimes possible to prove similar sparsity bounds through appropriate generalizations of the counting lemma and the dispersion lemma.

These generalized lemmas often analyze more complicated paths: not just non-backtracking, but also satisfying a number of other technical conditions. The precise type of paths that are considered is closely related to the precise version of blocking sets that are used.

We follow this high-level approach. Our first technical innovation is a new kind of blocking set, which we call a *double-blocking set*. The precise definition is given later in Definition 15, but the idea is to require not just that cycles of length at most $2k$ are blocked, but also that cycles of length at most $k+1$ are blocked *twice*. We can show that the greedy emulator H has a small double-blocking set (unlike fault-tolerant spanners). This is slightly more difficult than the analogous part of previous work on spanners and VFT emulators, but is still the more straightforward part of our proof.

The goal is then to show that any graph with this stronger kind of blocking set is particularly sparse. Following the outline for the Moore bounds, we need to define some type of path and prove a counting lemma and dispersion lemma. We introduce a new type of path, which we call SCUM paths (Simple, Chain Unblocked, Middle-Heavy), and prove the associated counting and dispersion lemmas (see Definition 19 for the precise definition). The chain unblocked property is our main new technical idea, as it differs substantially from the technical conditions on paths used in prior work. Once we have the definition of these paths, the counting lemma is reasonably straightforward, but the dispersion lemma is more technically involved.

2.3   Outline

We begin in Section 3 by formally stating our algorithm, proving that the graph H it returns is an f-EFT $(2k - 1)$-emulator, and then proving that H has a double-blocking set of size at most $f|E(H)|$. Then in Section 4 we begin the proof that *any* graph with a small double-blocking set is sparse enough to satisfy our main theorem statement. We do this by proving a counting lemma for SCUM paths in Section 4.2, stating but not proving dispersion lemmas for $k = 3$ and for general odd k in Section 4.3, and then showing in Section 4.4 how to combine the counting and dispersion lemmas to prove that H is sparse. We then prove our dispersion lemmas for $k = 3$ in Section 5, and for general odd k in Section 6. In Section 7 we show how to modify the algorithm to run in polynomial time without significantly affecting the sparsity bounds, and in Section 8 we construct our lower bounds.

3   Algorithm and Double-Blocking

We begin by formally giving our algorithm, proving that it does output an EFT emulator, and then proving that the EFT emulator it outputs has a small double-blocking set.

3.1   Algorithm and Correctness

We construct our emulators using the following greedy algorithm. Recall that H^F is the version of the graph H with weights updated according to distances in $G \setminus F$. So H^F is defined with respect to an ambient graph G, which will typically be clear from context.

■ **Algorithm 1** Algorithm for f-EFT $(2k-1)$-emulators.

Input: Graph $G = (V, E, w)$, positive integers f, k;
Let $H \leftarrow (V, \emptyset)$ be the initially-empty emulator;
foreach *edge $(u, v) \in E$ in order of nondecreasing weight $w(u, v)$* **do**
    if *there is $F \subseteq E$ of size $|F| \leq f$ with $dist_{H^F}(u, v) > (2k - 1) \cdot w(u, v)$* **then**
        Add (u, v) to H;
return H;

If two edges in the input graph have the same weight, then the algorithm considers them in arbitrary order.

Correctness of the output emulator is a standard argument:

▶ **Theorem 14** (Correctness of Algorithm 1). *The graph H returned by Algorithm 1 is an f-EFT $(2k - 1)$ emulator of the input graph G.*

Proof. Let F be an arbitrary set of $|F| \leq f$ failing edges. Our goal is to prove that H^F is a $(2k - 1)$-emulator of $G \setminus F$. By a standard observation, it suffices to prove that $dist_{H^F}(u, v) \leq (2k - 1) \cdot w(u, v)$ for all edges $(u, v) \in G \setminus F$ (by considering shortest paths, this implies the desired stretch bound for all pairs). There are two cases. First, if $(u, v) \in H$, then we immediately have $dist_{H^F}(u, v) = dist_{G \setminus F}(u, v) \leq w(u, v)$, due to the reweighting of (u, v) in H^F. Second, suppose $(u, v) \notin H$. We must have considered the edge (u, v) at some point in the algorithm, and then elected not to add it to H. By the algorithm conditional, this implies that for every F we have $dist_{H^F}(u, v) \leq (2k - 1) \cdot w(u, v)$, completing the proof. ◀
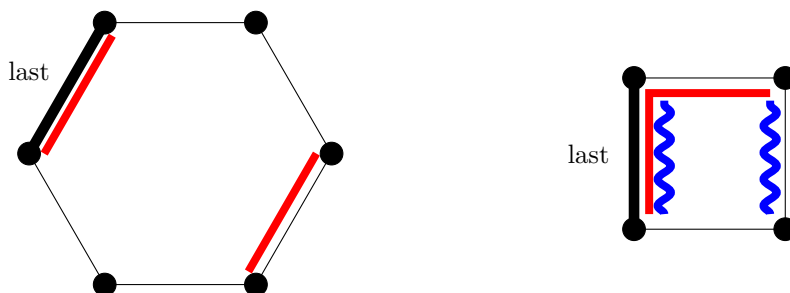
3.2 Double-Blocking Sets

In the following analysis, it will frequently be important to consider a subset of edges within H (typically a cycle or path) and consider the last among these edges to be processed by the main loop of our algorithm. We will use the phrase "the latest (or last) edge (in the ordering of H)" to identify this edge. Note that if all weights are distinct then this will be the highest-weight edge in the subset, but if some edges have the same weight then it specifically refers to the highest-weight edge under the same tiebreaking used by the algorithm.

We begin with our new definition of double-blocking sets.

▶ **Definition 15** (Double-Blocking Sets (see Figure 1 for $k = 3$)). *Let $H = (V, E)$ be a graph with a total ordering on its edges, let k be a positive integer, and let $B \subseteq \binom{E}{2}$. We say that a cycle C in H is* blocked *by $\{e_1, e_2\} \in B$ if we have $e_1, e_2 \in C$ and e_1 is the last edge in C in the ordering. The set B is called a $2k$ double-blocking set for G if:*
- *Every cycle C on $|C| \leq 2k$ edges is blocked by some $b \in B$, and*
- *Every cycle C on $|C| \leq k + 1$ edges is blocked by two distinct $b_1, b_2 \in B$.*

The edge pairs in a blocking set are called blocks.

■ **Figure 1** (Left) For any cycle C with ≤ 6 edges, a 6 double-blocking set B blocks C at least once, e.g., by the pair of red edges. (Right) Additionally, for any cycle C with ≤ 4 edges, B blocks C at least twice, e.g., by the pair of red edges and also the pair of wavy blue edges.

We note that the "short" cycles C of length $k+1$ have blocks of the form $\{e_1, e_2\}, \{e_1, e_3\}$; that is, both blocks must contain the last edge in C and thus they intersect on an edge. We now want to show that the output emulator H of the greedy algorithm has a small double-blocking set.

▶ **Lemma 16.** *The emulator H returned by Algorithm 1 has a $2k$ double-blocking set B of size $|B| \leq f|E(H)|$ (with respect to the order that the edges are added to H in Algorithm 1).*

Proof. Each time we add an edge (u, v) to H, we do so because of an edge subset $F_{(u,v)}$ with $\mathrm{dist}_{H^{F_{(u,v)}}}(u, v) > (2k - 1) \cdot w(u, v)$. Note that $F_{(u,v)}$ contains only edges earlier in the ordering that (u, v), and hence in particular does not contain (u, v). Define

$$B := \{\{e, f\} \mid e \in E(H), f \in F_e\}.$$

Since $|F_e| \leq f$ for all e, we have $|B| \leq f|E(H)|$. We now show that B is a $2k$ double-blocking set.
- Let C be a cycle in H of length $\leq 2k$, and let $(u, v) \in C$ be its last edge. When we add (u, v) to H, the other $2k - 1$ edges in C form a $u \rightsquigarrow v$ path of length $\leq (2k - 1) \cdot w(u, v)$. Thus, we must include one of these edges $f \in C \setminus \{(u, v)\}$ in $F_{(u,v)}$, to fault this short $u \rightsquigarrow v$ path. We thus have $\{(u, v), f\} \in B$, which blocks C as required.

- (See Figure 2) Now let C be a cycle in H of length $\leq k + 1$, and let $(u, v) \in C$ be its last edge. By the previous item, there exists at least one edge $(x, y) \in F_{(u,v)} \cap C$. Suppose towards contradiction that (x, y) is the *only* edge in $F_{(u,v)} \cap C$. Then:
    - In $G \setminus F_{(u,v)}$, we have $\text{dist}(x, y) \leq k \cdot w(u, v)$. This is due to the $x \rightsquigarrow y$ path that uses the other $\leq k$ edges in C, each of which has weight $\leq w(u, v)$. Note that this crucially uses the "automatic edge updating" in our model.
    - We have

        $$\text{dist}_{H^{F_{(u,v)}}}(u, v) \leq (2k - 1) \cdot w(u, v).$$

        This is due to the $u \rightsquigarrow v$ path that uses the other $\leq k$ edges in C. The $\leq k - 1$ edges besides (x, y) each have weight $\leq w(u, v)$, and by the previous item, the edge (x, y) has weight $\leq k \cdot w(u, v)$ in $H^{F_{(u,v)}}$.

    The latter point implies that the algorithm conditional is violated and the fault set $F_{(u,v)}$ would *not* cause us to add (u, v) to H, completing the contradiction. Hence there are at least two distinct edges $f_1, f_2 \in F_{(u,v)} \cap C$. Since $(u, v) \notin F_{(u,v)}$, this means that both $\{(u, v), f_1\}$ and $\{(u, v), f_2\}$ are in B by definition, so C is double-blocked as required. ◄
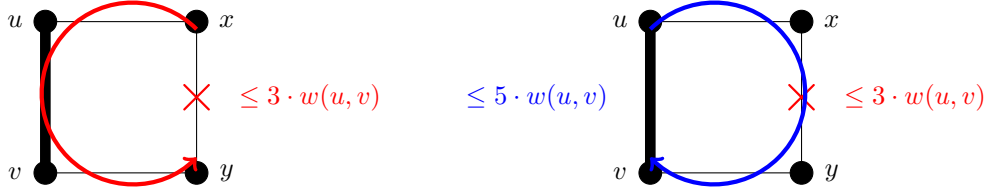
■ **Figure 2** In the proof of Lemma 16, for the case $k = 3$, we argue double-blocking of 4 cycles as follows. Suppose our greedy algorithm is currently considering whether to add (u, v), the heaviest edge in the cycle, to the emulator. (Left) If there is only a single edge (x, y) on the cycle included in the set of failing edges, then $w(x, y)$ increases to at most $3 \cdot w(u, v)$. (Right) This implies $\text{dist}(u, v) \leq 5 \cdot w(u, v)$, so we would not choose to add (u, v) to the emulator in the first place.

4   Main Size Analysis (assuming dispersion)

We now shift perspective. Instead of talking about the greedy algorithm or fault tolerance, our goal in the rest of the proof is simply to limit the maximum number of edges that can appear in an n-node graph H with a $2k$ double-blocking set of size $|B| \leq f|E(H)|$. In particular, we will prove the following theorems:

▶ **Theorem 17** (Sparsity for $k = 3$). *Let H be an n-node graph with an ordering on its edges which has a 6 double-blocking set of size at most $f|E(H)|$. Then $|E(H)| = O(n^{4/3} + n^{6/5}f^{3/5} + fn)$.*

▶ **Theorem 18** (Sparsity for general odd k). *Let k be a positive odd integer and let H be an n-node graph with an ordering on its edges which has a $2k$ double-blocking set of size at most $f|E(H)|$. Then*

$$|E(H)| = O_k \left(n^{1 + \frac{1}{k}} + n^{1 + \frac{1}{k+2}} f^{\frac{k}{k+2}} + n^{1 + \frac{1}{k+1}} f^{\frac{1}{2}} + fn \right).$$

Theorem 17 and Lemma 16 imply Theorem 6 (our main bound for $k = 3$), and Theorem 18 and Lemma 16 imply Theorem 8 (our main bound for general odd k). So it just remains to prove these two theorems.

4.1 Graph Cleaning

We first use some standard combinatorial arguments to make some additional assumptions on H without loss of generality. In particular, since our goal is to prove an upper bound on $|E(H)|$ based on the fact that it has $2k$ double-blocking set of size $|B| \leq f|E(H)|$, we may assume without loss of generality that the following are true:

- Every edge in H participates in at most f pairs in B.
- If d is the average degree of a node in H, then every node has degree $\Theta(d)$.
- $f \leq d/(Ck)$ for some large constant C. Note that we are assuming k is a constant, so this is equivalent to saying that $f \leq d/C'$ for a large constant C'.

The proof that we can assume these without loss of generality is relatively standard, so we defer it to the full version [3]. From now on, we will make these assumptions on H. We also carry forward the notation d for the average degree in H.

4.2 SCUM Path Counting

As discussed in Section 2, we introduce a new type of path to analyze, called SCUM paths.

▶ **Definition 19** (SCUM k-paths). *Let H be a graph with an ordering of its edges, and let $B \subseteq \binom{E(H)}{2}$ be a $2k$ double-blocking set. A SCUM k-path is a path π with k edges that satisfies all of the following properties. Let $(e_1 = (v_0, v_1), \ldots, e_k = (v_{k-1}, v_k))$ be the edge sequence of π, and let $m := (k+1)/2$,[2] such that e_m is the middle edge of π.*
- *(Simple) π does not repeat nodes,*
- *(Chain Unblocked) Let us say that an oriented path q is* chain unblocked *(with respect to B) if, for any edge $e \in q$ and node $v \in q$ where v strictly follows e (in particular $v \notin e$), there is no block of the form $\{e, e'\} \in B$ where e' is incident on v. We require:*
    - *The oriented subpath of π with edge sequence $(e_m, e_{m-1}, \ldots, e_1)$ is chain-unblocked, and*
    - *The oriented subpath of π with edge sequence $(e_m, e_{m+1}, \ldots, e_k)$ is chain-unblocked.*
- *(Middle-Heavy) the latest edge of π (in the edge ordering of H) is e_m.*

The main new ingredient in this definition is the chain unblocked property. Prior work has often focused on *unblocked paths* instead, e.g., where there is no $\{e, e'\} \in B$ with $e, e' \in \pi$ [7, 4].

The chain-unblocked property is qualitatively different and not directly comparable: it is possible for a path to be chain-unblocked and yet blocked in the traditional sense, and it is also possible for a path to be chain-blocked and yet unblocked in the traditional sense.

We now want to prove a counting lemma for our SCUM k-paths. The first technical step is to classify the edges $e \in E(H)$ into *core edges* and *non-core edges*. At the moment an edge $e \in E(H)$ is added to H, we consider the induced subgraph $H_e \subseteq H$ obtained by iteratively deleting any remaining nodes of degree $\leq d/4$. If e survives in H_e, then it is a *core edge*, otherwise it is a *non-core edge*. We have:

▷ **Claim 20.** At least half the edges of H are core edges.

Proof. Let H_{nc} be the subgraph that contains exactly the non-core edges of H. Notice that, by definition of non-core edges, if we iteratively delete nodes in H_{nc} of degree $\leq d/4$ then every edge will eventually be deleted from H_{nc}. By unioning over the n nodes in H_{nc}, we delete at most $n \cdot d/4 \leq |E(H)|/2$ edges in this process. Thus, $\leq |E(H)|/2$ of the edges in H are non-core edges, and the claim follows.   ◁

---

[2] We will focus on odd k, where this is integral. For even k, one would round up or down arbitrarily.

▶ **Lemma 21** (SCUM k-path Counting Lemma). *There are $n \cdot \Omega(d)^k$ SCUM k-paths in H.*

Proof. We may generate a SCUM k-path π as follows:

- Choose a core edge $e_m = (u, v)$ to act as the middle edge of π. Since π will be a k-path and k is odd, this means that $m = (k+1)/2$. Using Claim 20, there are $\Omega(nd)$ choices for e_m. Let $H_{e_m} \subseteq H$ be the subgraph witnessing that e_m is a core edge: that is, consider the subgraph of H that contains only e_m and preceding edges in the ordering, and then H_{e_m} is the induced subgraph obtained by iteratively deleting nodes of degree $\leq d/4$.
- Sequentially choose $m - 1$ edges $e_{m-1}, e_{m-2}, \ldots, e_1 \in E(H_{e_m})$ that extend e_m into a path of length m. If $e_i = (x, y)$, the next edge e_{i-1} may be selected among the $\Omega(d)$ edges incident to y in H_{e_m}, disallowing (1) the $\leq k$ edges that connect y to a node that already appears in π (to enforce simplicity), and (2) the $\leq fk$ edges that connect y to nodes z, such that an edge already in π is blocked with an edge incident to z (to enforce that π is chain unblocked). We have $fk \leq d/C$ for a large constant C, since we assume $f \leq d/(Ck)$ (see Section 4.1). So there are $\Omega(d)$ choices for the next edge at each step, for a total of $\Omega(d)^{m-1}$ choices.
- Repeat the above process to sequentially choose edges in the suffix $e_{m+1}, \ldots, e_k$. By identical logic, there are $\Omega(d)$ choices at each step, for a total of $\Omega(d)^{m-1}$ choices.

The SCUM properties follow from the construction. Since e_m is the middle edge of a path of length k, and k is odd, we have $m = (k+1)/2$. Thus we count

$$\Omega(nd) \cdot \Omega(d)^{m-1} \cdot \Omega(d)^{m-1} = n \cdot \Omega(d)^k$$

ways to generate a SCUM k-path by this process. ◀

4.3 Limiting SCUM Path Meets (Dispersion)

Now that we have a *lower* bound on the number of SCUM paths, we want to find an *upper* bound on a related but distinct object: SCUM k-path *meets*. Combining the two will lead to the desired sparsity bounds.

▶ **Definition 22** (SCUM Path Meets). *A SCUM k-path meet is a pair of distinct SCUM k-paths $\{\pi_1, \pi_2\}$ that use the same endpoints s, t.*

Proving an upper bound on SCUM k-path meets is the most technically complex part of our argument, so we defer the proofs to Section 5 for $k = 3$ and Section 6 for general odd k. We separate them since the bound for $k = 3$ is both stronger and easier to prove than the bound for general k. For now, we simply state the lemmas, and then in Section 4.4 we will show how to use them to prove our main theorems.

▶ **Lemma 23** (Dispersion lemma for $k = 3$). *When $k = 3$, there are $O(ndf^3)$ SCUM 3-path meets in H.*

▶ **Lemma 24** (Dispersion lemma for odd k). *When k is odd, the number of SCUM k-path meets in H is*

$$O_k \left(nf^k d^{k-2} + nf^{(k+1)/2} d^{k-1} \right).$$

4.4 Proof Wrapup

The rest of the proof is a matter of comparing our lower bound on the number of SCUM k-paths to the upper bound on the number of SCUM k-path meets, and algebraically rearranging our expressions into a bound on d. We return to general k for the following lemma.

▶ **Lemma 25.** *For any parameter $\alpha \geq 0$, if H has $\leq \alpha n^2$ SCUM k-path meets, then $d \leq O\left(n^{1/k} + n^{1/k}\alpha^{1/(2k)}\right)$.*

Proof. Let $x_{s,t}$ be the number of $s \rightsquigarrow t$ SCUM k-paths. We have:

$$
\sum_{(s,t)} x_{(s,t)} = \left|\{(s,t) \mid x_{(s,t)} = 1\}\right| + \left(\sum_{(s,t)|x_{(s,t)}\geq 2} x_{(s,t)}\right)
$$

$$
\leq n^2 + O\left(n^2 \cdot \sum_{(s,t)} \binom{x_{(s,t)}}{2}\right)^{1/2} \qquad \text{Cauchy-Schwarz}
$$

$$
\leq n^2 + O\left(n^2 \cdot \alpha n^2\right)^{1/2} = O\left(n^2 + n^2\alpha^{1/2}\right).
$$

We note that the second-to-last inequality is because, by definition, the number of $s \rightsquigarrow t$ SCUM k-path meets is exactly $\binom{x_{(s,t)}}{2}$. Thus, summing $\binom{x_{(s,t)}}{2}$ over all pairs (s,t) counts the number of SCUM k-path meets, which is at most αn^2.

By Lemma 21, we also have the lower bound $n \cdot \Omega(d)^k$ on the total number of SCUM k-paths. Thus,

$$
n \cdot \Omega(d)^k \leq O\left(n^2 + n^2\alpha^{1/2}\right).
$$

If the former term on the right-hand side dominates, then $d = O(n^{1/k})$. If the latter term on the right-hand side dominates, then $d = O(n^{1/k}\alpha^{1/(2k)})$. ◀

4.4.1 $k = 3$

We now again restrict to $k = 3$, and we wrap up the proof of Theorem 17. Plugging in the bound in Lemma 23 to the previous lemma, we have $d \leq O\left(n^{1/3} + n^{1/3}\left(df^3/n\right)^{1/6}\right)$. If the former term dominates, then $d = O(n^{1/3})$. If the latter term dominates, then $d^{5/6} \leq O\left(n^{1/6}f^{1/2}\right)$ and so $d \leq O\left(n^{1/5}f^{3/5}\right)$ which completes the proof of Theorem 17.

4.4.2 General Odd k

We now complete the proof of Theorem 18 for general k. By Lemma 24, the number of SCUM k-path meets is $O_k\left(nf^k d^{k-2} + nf^{(k+1)/2}d^{k-1}\right)$.

Suppose the first term dominates. Plugging into Lemma 25, we get $d = O(n^{1/k}) + n^{1/k} \cdot O_k\left(\frac{f^k d^{k-2}}{n}\right)^{\frac{1}{2k}}$. If the former term dominates, we get $d = O(n^{1/k})$. If the latter term dominates, we get $d^{k+2} = n \cdot O_k\left(f^k\right)$ and so $d = O_k\left(f^{\frac{k}{k+2}}n^{\frac{1}{k+2}}\right)$. Either of these bounds on d suffices to prove Theorem 18.

On the other hand, returning to our bound on the number of SCUM k-path meets, suppose the second term dominates. Again plugging into Lemma 25, we get $d = O(n^{1/k}) + n^{1/k} \cdot O_k\left(\frac{f^{(k+1)/2}d^{k-1}}{n}\right)^{\frac{1}{2k}}$. If the first term dominates we again get $d = O(n^{1/k})$. If the latter term dominates, we get $d^{k+1} = O_k\left(n \cdot f^{(k+1)/2}\right)$ and so $d = O_k\left(f^{1/2}n^{\frac{1}{k+1}}\right)$. Again, either of these bounds suffices to prove Theorem 18.

5   Limiting SCUM Path Meets ($k = 3$)

In this section we prove Lemma 23, which as discussed implies Theorem 6.

5.1   Technical Lemmas on Path Structure

Our focus in this part is on $k = 3$ specifically (i.e., graphs with a 6 double-blocking set). In this setting, the necessary technical lemmas about the structure of our SCUM paths become simpler to state and use. The most important simplification is reflected by Lemma 27, which shows that in any SCUM 3-path meet, the two paths of the meet are internally vertex-disjoint. This lemma does *not* generalize to larger k, which introduces technical complexities and makes our quantitative bounds a bit worse. Analogous structural lemmas for general k are given in Section 6.1.

Our goal is to prepare for a dispersion lemma limiting the number of SCUM 3-path meets. We note that, when considering a graph with a $2k$ blocking set, one generally focuses on meets between k-paths (not $2k$-paths). This is analogous to the Moore bounds, and the reason is roughly because two k-paths in a meet create a cycle on $\leq 2k$ edges, and so our blocking set is indeed helpful in analyzing the structure of this meet.

The following three lemmas hold only for $k \geq 3$ because they rely on the set of blocks in B (either explicitly or by discussion of SCUM paths), and we will use that B is a (≥ 6) double blocking set in their proofs.

▶ **Lemma 26.** *Suppose $k \geq 3$, let $\{\pi^\ell, \pi^r\}$ be a SCUM 3-path meet, let e^* be the middle edge of π^ℓ, and suppose e^* is the latest edge in the ordering in $\pi^\ell \cup \pi^r$. Let F^* be the set of edges that appear in a block with e^* in B. Then the middle edge of π^r is in F^*.*

Proof. Notice that $e^* \notin \pi^r$, since this would imply $\pi^\ell = \pi^r$ but π^ℓ, π^r must be distinct in a meet. Thus, the paths $\pi^\ell \cup \pi^r$ form a cycle C of length ≤ 6, and this cycle contains e^* as its heaviest edge. We therefore have a block of the form $\{e^*, e\} \in B$ with $e \in C$. Since π^ℓ is chain unblocked, e cannot be incident on the first or last node of π^ℓ. Thus, the only possible placement of e in $\pi^\ell \cup \pi^r$ is as the middle edge of π^r.   ◀

▶ **Lemma 27.** *Suppose $k \geq 3$. In any SCUM 3-path meet $\{\pi^\ell, \pi^r\}$, the paths π^ℓ, π^r are internally vertex disjoint. (That is, they intersect only on their endpoints, and otherwise do not share nodes.)*

Proof. Let s, t be the endpoints of π^ℓ, π^r, let $e^* = (u^*, v^*)$ be the latest edge in the ordering in $\pi^\ell \cup \pi^r$, and suppose without loss of generality that e^* is the middle edge of π^ℓ. Notice that e^* cannot be the middle edge e' of π^r, since this would imply $\pi^\ell = \pi^r$.

Seeking contradiction, suppose that π^ℓ, π^r intersect on their first edge (s, u^*). Then the last two edges of π^ℓ, π^r form a 4-cycle C with nodes (u^*, v^*, t, x), which must be double-blocked in B. It is possible that the middle edges $\{e^* = (u^*, v^*), (u^*, x)\}$ appear together in B, which would block C once. But, since π^ℓ is chain unblocked, e^* cannot appear in B with either of the other edges $\{(v^*, t), (t, x)\}$ which completes the contradiction.

The argument that π^ℓ, π^r cannot intersect on their last edges (v^*, t) is symmetric.   ◀

▶ **Lemma 28.** *When $k \geq 3$, for any nodes s, t, there are at most $f + 1$ simple $s \rightsquigarrow t$ paths in H of length 2.*

Proof. Let $\Pi_{s,t}$ be the set of $s \leadsto t$ paths of length 2. Notice that any two paths in $\Pi_{s,t}$ form a 4-cycle, which must be double-blocked. Let e^* be the latest edge in the ordering contained in any path in $\Pi_{s,t}$. Then for any path $\pi \in \Pi_{s,t}$ with $e^* \notin \pi$, there exists a block of the form $\{e^*, e\}$ with $e \in \pi$. Since we may assume without loss of generality that e^* is only blocked with at most f other edges (see Section 4.1), it follows that $|\Pi_{s,t}| \leq f + 1$.   ◀

5.2   Proof of Lemma 23

We can generate all SCUM 3-path meets $\{\pi^\ell, \pi^r\}$ by the following process (see Figure 3).

- Choose an edge $e^* = (u^*, v^*)$ to act as the latest edge in $\pi^\ell \cup \pi^r$ in the ordering. We will assume without loss of generality that e^* is the middle edge of π^ℓ. There are $O(nd)$ possible choices of e^*.

- Let F^* be the edges that are paired with e^* in B. By Lemma 26, π^r uses an edge $e' = (x, y) \in F^*$ as its middle edge. There are $O(f)$ possible choices of e'.

- By Lemma 27, π^ℓ, π^r are internally vertex disjoint, so $u^* \neq x$ and $v^* \neq y$. Thus, we may complete the meet by choosing a simple $u^* \leadsto x$ 2-path and a simple $v^* \leadsto y$ 2-path. By Lemma 28, there are $O(f)$ ways to choose each 2-path, and thus $O(f^2)$ ways to complete the meet in total.

Putting it together, we have $O(nd \cdot f \cdot f^2)$ meets.

$$O(f) \; v^* \leadsto y \text{ 2-paths}$$

$$O(f) \; u^* \leadsto x \text{ 2-paths}$$
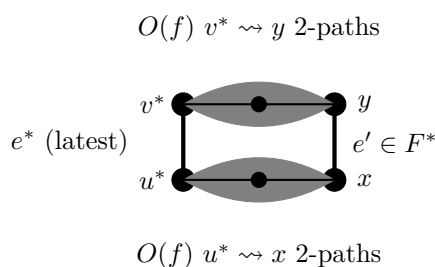
Figure 3 The four choices in our generation of SCUM 3-path meets lead to a bound of $O(ndf^3)$ on the total number of meets. The common endpoints of the two paths in the meet occur as the middle nodes of the $v^* \leadsto y$ and $u^* \leadsto x$ endpoints that we select.

6   Limiting SCUM Path Meets for General Odd k

In this section we prove Lemma 24, which as discussed implies Theorem 8.

We follow the same rough strategy as for the $k = 3$ case. To give intuition, let us briefly sketch counting of meets between *internally vertex-disjoint* SCUM k-paths $\{\pi^\ell, \pi^r\}$. We can generate such a meet by the following process (see Figure 4):

1. First choose an edge $e^* = (u^*, v^*)$ to act as the heaviest edge in the meet. Suppose $e^* \in \pi^\ell$.

2. Similar to the $k = 3$ case, we can argue that there exists an edge $e' \in \pi^r$ that is blocked with e^*. For $k = 3$ we specifically had that e' occurs in the *middle* position of π^r; for general k, we can only say that it occurs in position 2 through $k - 1$. Choose this edge e' and its position in π^r.

3. Extend e' into a subpath of π^r that contains all but the first and last edge. Let x, y be the endpoints of this $(k - 2)$-length subpath of π^r.

4. Complete the meet by choosing paths of length $(k+1)/2$ from u^* to x and from v^* to y. Limiting the number of such paths requires a technical lemma, but we can prove a reasonably good upper bound by exploiting the structure of the double-blocking set.
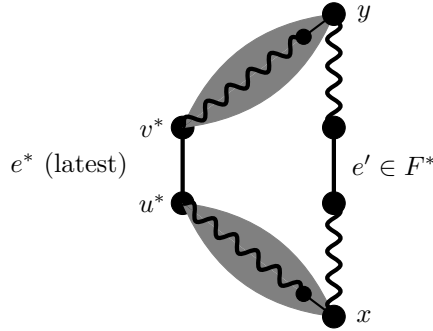
■ **Figure 4** A sketch of an argument that counts the *internally vertex-disjoint* SCUM k-path meets in H.

Challenges in extension to general k.   This sketched argument is the natural analog of our argument for $k = 3$. In fact, *if it were true that all meets are internally vertex disjoint*, this argument would provide a range of free fault tolerance for both odd and even $k > 3$. Unfortunately, we cannot make such an assumption when $k > 3$, and we also need to count meets that nontrivially coincide. Counting meets that share prefixes/suffixes introduces a new term in our dispersion lemma, which harms our bounds for general odd k relative to $k = 3$, and which spoils our ability to prove free fault tolerance for any even k.

6.1   Technical Lemmas on Path Structure

In the following, we say that a path π in H is *blocked* if there is an edge pair $\{e_1, e_2\} \in B$ with $e_1, e_2 \in \pi$, and π is *double-blocked* if there are two such edge pairs in B. Note that this differs slightly from the notion of a cycle being blocked, as we do not require that e_1 is the latest edge in π.

▶ **Lemma 29** (Extends Lemma 26). *Let π_1, π_2 be simple paths with the same endpoints s, t. Suppose $e^* \in \pi_1$ is the last edge in $\pi_1 \cup \pi_2$ in the ordering, and let F^* be the set of edges that are in a block with e^* in B. Then there exists an edge $e' \in F^* \cup \{e^*\}$ with $e' \in \pi_2$ if either of the following conditions hold:*
- π_1, π_2 *are both unblocked and have length* $\leq k$*, or*
- π_1, π_2 *are both un-double-blocked and have length* $\leq (k+1)/2$*.*

Proof. Assume π_1, π_2 are unblocked and have length $\leq k$, and suppose $e^* \notin \pi_2$. Then there is a subpath of π_1 and a subpath of π_2 that form a cycle C with $|C| \leq 2k$ edges, and latest edge e^*. This cycle must be blocked by B; since e^* is its latest edge, there is a block of the form $\{e^*, e'\} \in B$ with $e' \in C$. Since π_1 is unblocked, we have $e' \notin \pi_1$, so $e' \in \pi_2$, completing the proof.

The case where π_1, π_2 are un-double-blocked and have length $\leq (k+1)/2$ is essentially identical, noting that their subpaths form a cycle of length $\leq k+1$ which is thus double-blocked by B.   ◀

▶ **Lemma 30** (Extends Lemma 28). *For any two nodes s, t and any $1 \leq L \leq (k+1)/2$, there are at most $O\left(f^{L-1}\right)$ simple $s \rightsquigarrow t$ paths of length L that are not double-blocked.*

Proof. We prove this lemma by strong induction on L. The base case $L = 1$ is trivial, since there is at most 1 edge between any two nodes.

For the inductive step: let $\Pi_{s,t}$ be the set of all simple $s \rightsquigarrow t$ paths of length L that are not double-blocked, let e^* be the latest edge in the ordering contained in any path in $\Pi_{s,t}$, and let F^* be the edges that appear with e^* in a block in B. By the previous lemma, every path $\pi \in \Pi_{s,t}$ contains an edge in $F^* \cup \{e^*\}$.

Fix $(u, v) \in F^* \cup \{e^*\}$ and a position i, and let us count the number of paths in $\Pi_{s,t}$ that use (u, v) in position i.

- If $i \in \{2, \ldots, L-1\}$, then we may apply the inductive hypothesis twice to argue that we have $O(f^{i-2})$ $s \rightsquigarrow u$ paths of length $i-1$, and $O(f^{L-i-1})$ $v \rightsquigarrow t$ paths of length $L-i$, which gives $O(f^{L-3})$ paths that use (u, v) as their i^{th} edge.
- If $i = 1$ (and so $s = u$), then we may apply the inductive hypothesis once to argue that we have $O(f^{L-2})$ $v \rightsquigarrow t$ paths of length $L-1$.
- If $i = L$ (and so $v = t$), then we may apply the inductive hypothesis once to argue that we have $O(f^{L-2})$ $s \rightsquigarrow u$ paths of length $L-1$.

Unioning over the $\leq k(f+1)$ choices of (u, v) and i, we count $O(f^{L-1})$ paths in $\Pi_{s,t}$.   ◀

The following lemma is new to general k and does not extend any previous lemmas.

▶ **Lemma 31.** *Let s, t be nodes, let $0 \leq L_1 \leq (k-1)/2$, and let $L_2 \geq 1$. Then there are $O(f^{L_1} d^{L_2 - 1})$ pairs of simple paths (π_1, π_2) in H with the following properties:*

- *π_1 has length L_1 and start point s. π_2 has length L_2 and start point t. π_1, π_2 have the same endpoint node (that is: we are counting pairs of $s \rightsquigarrow x, t \rightsquigarrow x$ paths, over all possible choices of node x).*
- *π_1 is chain-unblocked.*
- *π_2 is not a subpath of π_1.*

Proof. First, we acknowledge that it is possible that the paths in the meet π_1, π_2 coincide on a nontrivial suffix. Let c be the number of edges in the longest common suffix of π_1, π_2. We begin the proof by assuming $c = 0$; we will revisit the general case at the end. A pair of paths (π_1, π_2) as described in the lemma may be generated by the following process.

- Starting from t, choose any simple path of length $L_2 - 1$. We know from Section 4.1 that without loss of generality we may assume that there are $\Theta(d)$ edges incident to each node, and so there are $O(d)^{L_2 - 1}$ possible choices for this path.
- Since we have assumed $c = 0$, we notice that π_1 plus the last edge of π_2 forms a simple path q of length $L_1 + 1 \leq (k+1)/2$. The last pair of edges on q (i.e., the last edge of π_1 and the last edge of π_2) could possibly appear as a block in B. However, since π_1 is chain unblocked, no other pair of edges on q can appear as a block. Thus, q is not double-blocked. Applying Lemma 30, there are $O(f^{L_1})$ ways to complete the rest of the meet.

We count $O(d)^{L_2 - 1} \cdot O(f^{L_1})$ ways to generate this meet, which gives our claimed bound for $c = 0$. We then consider general c. Notice that a pair of paths (π_1, π_2) as described in the lemma, which coincide on their last c edges, may be identified by (1) a pair of paths as described in the lemma of lengths $L_1 - c, L_2 - c$, which do not share a nontrivial suffix, and (2) an additional path of length c from their shared endpoint, representing the common suffix. Since we assume that π_2 is not a subpath of π_1, we note that $L_2 - c \geq 1$. So we may apply the first part of the argument, and conclude that there are $O(f^{L_1 - c} d^{L_2 - c - 1})$ ways to

choose the path pair of lengths $L_1 - c, L_2 - c$, and then there are $O(d)^c$ ways to choose the common suffix. This gives a stronger bound of $O(f^{L_1-c}d^{L_2-1})$ on the number of path pairs in the setting of general c. ◄

6.2   Proof of Lemma 24

We count SCUM k-path meets $\{\pi^\ell, \pi^r\}$ by the following process.

- Choose an edge $e^* = (u^*, v^*)$ to act as the latest edge in $\pi^\ell \cup \pi^r$ in the ordering. The latest edge in the ordering must be the middle $(m = (k+1)/2)$ edge of either π^ℓ or π^r; we will assume without loss of generality that e^* is the middle edge of π^ℓ. There are $O(nd)$ possible choices of e^*.
- Let F^* be the edges that are paired with e^* in B. By Lemma 29, π^r contains an edge $e' = (u', v') \in F^* \cup \{e^*\}$. Moreover, since π^ℓ is chain unblocked, e' cannot be the first or last edge in π^r; rather, it occurs in position 2 through $k-1$. Select one of these positions for e' along π^r. There are $\leq (f+1)(k-2)$ possible choices of e' and its position in π^r.
- We next select prefixes for π^ℓ, π^r; that is, we determine the parts of the paths respectively preceding nodes u^*, x. We can view the prefix of π^ℓ as an oriented SCU path of length $m - 1$, and the prefix of π^r as a simple path of length $i - 1 \geq 1$. There are two cases:
  - If the suffix of π^r is not a subpath of the suffix of π^ℓ, then the conditions of Lemma 31 hold. Applying this lemma, we have $O(f^{m-1}d^{i-2})$ choices of prefix pairs.
  - If the prefix of π^r is a subpath of the prefix of π^ℓ, then it is fully determined by a choice of prefix of π^ℓ. There are $O(d)^{m-1}$ possible choices of prefix.
- Finally, we select suffixes for π^ℓ, π^r. Exactly like the prefixes, we can view the suffix of π^ℓ as an oriented SCU path of length $m - 1$, and the suffix of π^r as a simple path of length $k - i \geq 1$. We have the same two cases in our counting:
  - In the case where the suffix of π^r is not a subpath of the suffix of π^ℓ, then by Lemma 31 we have $O(f^{m-1}d^{k-i-1})$ choices of suffix pairs.
  - In the case where the suffix of π^r is a subpath of the suffix of π^ℓ, then we have $O(d)^{m-1}$ choices of suffix.

We now put the parts together. There are $O(ndf)$ ways to make choices in the first two steps. If neither the prefix nor suffix of π^r is a subpath of the prefix/suffix of π^ℓ, then the number of ways to choose prefixes and suffixes is

$$O\left(f^{m-1}d^{i-2}\right) \cdot O\left(f^{m-1}d^{k-i-1}\right) = O\left(f^{2m-2}d^{k-3}\right).$$

If the prefix of π^r is a subpath of the prefix of π^ℓ, but the suffix is not, then the number of choices is

$$O\left(d^{m-1}\right) \cdot O\left(f^{m-1}d^{k-i-1}\right) = O\left(f^{m-1}d^{k-(i-m)-2}\right) = O\left(f^{m-1}d^{k-2}\right),$$

where the last equality is because we must have $i \leq m$ since the prefix of π^r is a subpath of the prefix of π^ℓ. The counting is symmetric in the case where the suffix of π^r is a subpath of the suffix of π^ℓ, and leads to the same bound. Finally, we do not need to consider the case where both the prefix and suffix of π^r is a subpath of the prefix/suffix of π^ℓ, as this would imply $\pi^\ell = \pi^r$ but the paths in a meet must be distinct.

The claimed bound on meets in this lemma now follows by multiplying our prefix/suffix bounds in each case by ndf, and substituting $m = (k+1)/2$.

7 Polynomial-Time Version

Algorithm 1 is actually slightly underspecified, since we do not describe how to determine whether there is some $F \subseteq E$ with $|F| \leq f$ such that $\text{dist}_{H^F}(u,v) > (2k-1) \cdot w(u,v)$. The obvious approach simply tries all possible F. This is correct, but takes time at least $\Omega\left(\binom{m}{f}\right)$, which is not polynomial when f is not constant. Hence while Algorithm 1 is sufficient to prove the structural results we claim, it is not actually *efficient* as an algorithm.

This parallels the situation with fault-tolerant spanners, where the first truly tight bounds on the size used a similar greedy algorithm that did not run in polynomial time [8]. To overcome this, Dinitz and Robelle [11] showed that slightly suboptimal bounds (by a factor of k) could be achieved by *approximating* the "if" condition rather than actually solving it. We now show that a similar result can be obtained for edge-fault-tolerant emulators: we can modify Algorithm 1 to run in polynomial time at the cost of only $O(k)$ in the achieved size.

Due to space constraints, we refer the interested reader to the full version of this paper [3].

8 Lower Bounds

8.1 General Lower Bound

▶ **Theorem 32.** *For all k, f, there are n-node graphs G for which any f-EFT k-emulator H has $|E(H)| = \Omega(nf)$.*

Proof. Let G be any $f/2$-regular n-node graph, and let H be an f-EFT k-emulator of G. Consider an edge $(u,v) \in E(G)$, and seeking contradiction, suppose $(u,v) \notin E(H)$. Consider the fault set F that contains all edges incident to u and all edges incident to v, except for (u,v) itself. We have $\text{dist}_{G \setminus F}(u,v) = 1$, and also for all nodes $x \notin \{u,v\}$, we have

$$\text{dist}_{G \setminus F}(u,x) = \text{dist}_{G \setminus F}(v,x) = \infty.$$

Thus, any edge incident to u or v in H^F has weight ∞, and so $\text{dist}_{H^F}(u,v) = \infty$. Thus H is not a k-emulator of $G \setminus F$, completing the contradiction. ◀

8.2 Lower Bound for $k = 2$

Here we prove the following lower bound for the setting $k = 2$, i.e., emulators with stretch 3:

▶ **Theorem 33.** *For any f, there are n-node graphs G for which any f-EFT 3-emulator H has $|E(H)| = \Omega(f^{1/2}n^{3/2})$.*

We use a standard construction in prior work [5, 4]. Start with a graph on n/f nodes,[3] $\Omega(n/f)^{3/2}$ edges, and girth > 4. It will be a little convenient later to assume this graph is bipartite (without loss of generality). Then, blow up each node v into a "cloud" C_v, which contains f copies of v. For each edge (u,v) we include an edge between every copy in C_u and every copy in C_v. This graph has $\Omega(f^{1/2}n^{3/2})$ edges. Let H be an $4f$-EFT 3-emulator of G (with a sufficiently large implicit constant hidden in the O). We will prove that $|E(H)| = \Omega(f^{1/2}n^{3/2})$.

Consider an edge $(u,v) \in E(G)$ and define the following fault set F, whose size $|F|$ might be much larger than $4f$. See Figure 5 for a picture.

---

[3] We ignore issues of non-integrality, which affect our bounds only by lower-order terms.

- Fault all edges from u to any node in the cloud C_v, and fault all edges from v to any node in the cloud C_u, except for (u, v) itself. This contributes $2f - 1$ edges to F.
- For each edge $(u, x) \in E(H)$ for which u, x lie in different clouds and there exists a 2-path in G of the form (u, v, x), fault the edge (v, x). Let $Z_u \subseteq F$ be all edges faulted in this way.
- For each edge $(t, v) \in E(H)$ for which there exists a 2-path of the form (t, u, v), fault the edge (t, u). Let $Z_v \subseteq F$ be all edges faulted in this way.
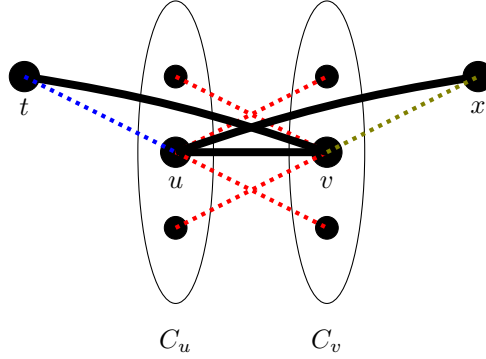
Figure 5 The dotted lines in this diagram are edges in the fault set F associated to (u, v), and the solid black lines are edges in H that are not contained in F. The dotted blue edge on the left is in Z_v, and the dotted yellow edge on the right is in Z_u. For clarity, we have omitted some (unfaulted) edges in this picture going between C_u and C_v.

Recall that H^F is the reweighted version of H under this fault set F. We will describe edges in H^F as "spanner edges" if the edge is also in $G \setminus F$, or "emulator edges" if not.

▷ **Claim 34.** If $(u, v) \notin E(H)$, then $\text{dist}_{H^F}(u, v) > 3$.

Proof. We will prove this claim in three cases. We argue that there is no one-edge $u \rightsquigarrow v$ path in H^F, there is no *short* two-hop $u \rightsquigarrow v$ path in H^F, and there is no *short* three-hop $u \rightsquigarrow v$ path in H^F. Since all edge weights in H^F are at least 1, any path with ≥ 4 hops has length ≥ 4 and may be ignored.

- (One-hop paths) By assumption, $(u, v) \notin E(H)$, so there is no 1-hop path from u to v.
- (Two-hop paths) We next argue that there is no 2-hop path from u to v of length ≤ 3.
  - Since G is bipartite and $(u, v) \in E(G)$, there can be no 2-hop $u \rightsquigarrow v$ path in H^F that specifically uses two *spanner* edges.
  - Each emulator edge in H^F has weight at least 2, so any 2-hop $u \rightsquigarrow v$ path in H^F that uses two *emulator* edges has length ≥ 4.
  - The remaining subcase is to argue that there is no 2-hop $u \rightsquigarrow v$ path of total length ≤ 3 in H^F that uses exactly one spanner edge and one emulator edge. Suppose towards contradiction that such a path π exists, and note that its emulator edge needs to have weight exactly 2. Since G has girth > 4, the path π cannot correspond to a $u \rightsquigarrow v$ 3-path in G that avoids the edge (u, v). It follows that the emulator edge in π either has the form (u, x) and there is a 2-path of the form (u, v, x) in G, or it has the form (t, v) and there is a 2-path (t, u, v) in G. In either case, the associated spanner edge (v, x) or (t, u) is included in Z_u or Z_v (respectively). Thus the remaining edge in π would not be a spanner edge, completing the contradiction.

- (Three-hop paths) Finally, we argue that there is no 3-edge path from u to v of length ≤ 3. Such a path would need to use all spanner edges, since emulator edges have weight ≥ 2. Since $(u, v) \notin E(H^F)$ and the original graph has girth > 4, no such 3-path may exist. ◄

▷ **Claim 35.** Either $(u, v) \in E(H)$, $|Z_u| > f$, or $|Z_v| > f$.

Proof. Suppose $(u, v) \notin E(H)$. From the previous claim, we have $\mathrm{dist}_{H \setminus F}(u, v) > 3$. Since H is a $4f$-EFT 3-emulator, this is possible only if $|F| > 4f$, which is possible only if $|Z_u| > f$ or $|Z_v| > f$. ◁

Now we count edges in $E(H)$. For each pair of adjacent clouds C_u, C_v, there are two cases:

- Suppose that at least half of the edges in $C_u \times C_v$ are present in $E(H)$. In this case, this cloud pair contributes $\Omega(f^2)$ edges to $E(H)$.
- Otherwise, suppose that at least half of the edges in $C_u \times C_v$ are missing in $E(H)$. Let (u, v) be one such missing edge. By our previous claim, we have $|Z_u| > f$ or $|Z_v| > f$. There are two cases:
  - ($|Z_u| > f$) In this case, we have f edges in $E(H)$ of the form (t, v) with a $t \rightsquigarrow v$ 2-path in G. Notice that all such 2-paths have the form (t, u', v) with $u' \in C_u$. We may charge all $> f$ of these edges to the pair (C_u, v).
  - ($|Z_v| > f$) In this case, following identical logic to the previous case, we may charge $> f$ edges to the pair (u, C_v).

  To summarize, at least half of the edges in $C_u \times C_v$ are missing in $E(H)$, and each missing edge goes between two nodes (u, v) where we have charged either (u, C_v) or (C_u, v) for $> f$ emulator edges. It follows that $\Omega(f^2)$ total emulator edges are charged in total when considering $C_u \times C_v$. (We remark that each emulator edge is charged twice in total; e.g., once from the cloud pair $C_u \times C_v$ and again from the cloud pair $C_t \times C_u$, but this factor of two may be ignored.)

Putting it together: for each of the $\Omega((n/f)^{3/2})$ adjacent cloud pairs in G, we charge $\Omega(f^2)$ edges in H (due to one case or the other). Thus we have

$$|E(H)| = \Omega\left(f^2 \cdot (n/f)^{3/2}\right) = \Omega\left(f^{1/2} n^{3/2}\right).$$

—— **References** ——

1 Reyan Ahmed, Greg Bodwin, Faryad Darabi Sahneh, Keaton Hamm, Mohammad Javad Latifi Jebelli, Stephen Kobourov, and Richard Spence. Graph spanners: A tutorial review. *Computer Science Review*, 37:100253, 2020.

2 Ingo Althöfer, Gautam Das, David P. Dobkin, Deborah Joseph, and José Soares. On sparse spanners of weighted graphs. *Discrete & Computational Geometry*, 9:81–100, 1993.

3 Greg Bodwin, Michael Dinitz, and Yasamin Nazari. Epic fail: Emulators can tolerate polynomially many edge faults for free. *CoRR*, abs/2209.03675, 2022. `doi:10.48550/arXiv.2209.03675`.

4 Greg Bodwin, Michael Dinitz, and Yasamin Nazari. Vertex Fault-Tolerant Emulators. In Mark Braverman, editor, *13th Innovations in Theoretical Computer Science Conference (ITCS 2022)*, volume 215 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 25:1–25:22, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.ITCS.2022.25`.

5   Greg Bodwin, Michael Dinitz, Merav Parter, and Virginia Vassilevska Williams. Optimal vertex fault tolerant spanners (for fixed stretch). In Artur Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 1884–1900. SIAM, 2018.

6   Greg Bodwin, Michael Dinitz, and Caleb Robelle. Optimal vertex fault-tolerant spanners in polynomial time. In *Proceedings of the Thirty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2021*, 2021.

7   Greg Bodwin, Michael Dinitz, and Caleb Robelle. Partially optimal edge fault-tolerant spanners. In *Proceedings of the Thirty-Thurd Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2022*, 2022.

8   Greg Bodwin and Shyamal Patel. A trivial yet optimal solution to vertex fault tolerant spanners. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, PODC '19, pages 541–543, New York, NY, USA, 2019. Association for Computing Machinery. `doi:10.1145/3293611.3331588`.

9   Shiri Chechik, Michael Langberg, David Peleg, and Liam Roditty. Fault tolerant spanners for general graphs. *SIAM J. Comput.*, 39(7):3403–3423, 2010.

10   Michael Dinitz and Robert Krauthgamer. Fault-tolerant spanners: better and simpler. In *Proceedings of the 30th Annual ACM Symposium on Principles of Distributed Computing, PODC 2011, San Jose, CA, USA, June 6-8, 2011*, pages 169–178, 2011.

11   Michael Dinitz and Caleb Robelle. Efficient and simple algorithms for fault-tolerant spanners. In *Proceedings of the 2020 ACM Symposium on Principles of Distributed Computing*, PODC '20, 2020.

12   Dorit Dor, Shay Halperin, and Uri Zwick. All-pairs almost shortest paths. *SIAM Journal on Computing*, 29(5):1740–1759, 2000.

13   Paul Erdős. Extremal problems in graph theory. In *In Theory of Graphs and its Applications, Proc. Sympos. Smolenice*, 1964.

14   Merav Parter. Nearly optimal vertex fault-tolerant spanners in optimal time: Sequential, distributed, and parallel. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022, pages 1080–1092, New York, NY, USA, 2022. Association for Computing Machinery. `doi:10.1145/3519935.3520047`.

15   David Peleg and Alejandro A. Schäffer. Graph spanners. *Journal of Graph Theory*, 13(1):99–116, 1989.

16   David Peleg and Jeffrey D. Ullman. An optimal synchronizer for the hypercube. *SIAM J. Comput.*, 18(4):740–747, 1989.