Fault-Tolerant Spanners for General Graphs

S. Chechik * M. Langberg † D. Peleg * L. Roditty ‡

Abstract

The paper concerns graph spanners that are resistant to vertex or edge failures. Given a weighted undirected n-vertex graph $G = (V, E)$ and an integer $k \geq 1$, the subgraph $H = (V, E')$, $E' \subseteq E$, is a *spanner* of stretch k (or, a k-spanner) of G if $\delta_H(u, v) \leq k \cdot \delta_G(u, v)$ for every $u, v \in V$, where $\delta_{G'}(u, v)$ denotes the distance between u and v in G'. Graph spanners were extensively studied since their introduction over two decades ago. It is known how to efficiently construct a $(2k-1)$-spanner of size $O(n^{1+1/k})$, and this size-stretch tradeoff is conjectured to be tight.

The notion of *fault tolerant* spanners was introduced a decade ago in the geometric setting [Levcopoulos et al., STOC'98]. A subgraph H is an f-vertex fault tolerant k-spanner of the graph G if for any set $F \subseteq V$ of size at most f and any pair of vertices $u, v \in V \setminus F$, the distances in H satisfy $\delta_{H \setminus F}(u, v) \leq k \cdot \delta_{G \setminus F}(u, v)$. Levcopoulos et al. presented an efficient algorithm that constructs an f-vertex fault tolerant *geometric* $(1+\epsilon)$-spanner, that is, given a set S of n points in $\mathbb{R}^d$, their algorithm finds a sparse graph H such that for every set $F \subseteq S$ of size f and any pair of points $u, v \in S \setminus F$ it satisfies that $\delta_{H \setminus F}(u, v) \leq (1+\epsilon)|uv|$, where $|uv|$ is the Euclidean distance between u and v. A fault tolerant geometric spanner with optimal maximum degree and total weight was presented in [Czumaj and Zhao, SoCG'03]. This paper also raised as an open problem the question whether it is possible to obtain a fault tolerant spanner for an arbitrary undirected weighted graph.

The current paper answers this question in the affirmative, presenting an f-vertex fault tolerant $(2k-1)$-spanner whose size is $O(f^3 k^{f+1} \cdot n^{1+1/k} \log^{1-1/k} n)$. Interestingly, the stretch of the spanner remains unchanged while the size of the spanner only increases by a factor that depends on the stretch k, on the number of potential faults f, and on logarithmic terms in n. In addition, we consider the simpler setting of f-*edge* fault tolerant spanners (defined analogously). We present an f-edge fault tolerant $2k-1$ spanner with edge set of size $O(f \cdot n^{1+1/k})$ (only f times larger than standard spanners). For both edge and vertex faults, our results, are shown to hold when the given graph G is weighted.

*Department of Computer Science and Applied Mathematics, The Weizmann Institute of Science, Rehovot 76100, Israel, email: {shiri.chechik,david.peleg}@weizmann.ac.il

†Computer Science Division, Open University of Israel, 108 Ravutski St., Raanana 43107, Israel, email: mikel@openu.ac.il

‡Department of Computer Science, Bar-Ilan University, Ramat-Gan 52900, Israel, email: liamr@macs.biu.ac.il

1 Introduction

Graph Spanners Graph spanners are fundamental graph structures, generalizing the concept of spanning trees. A graph spanner can be thought of intuitively as a skeleton structure that allows us to faithfully represent the underlying network using few edges, in the sense that for any two nodes of the network, the distance in the spanner is stretched by only a small factor. More formally, consider a weighted undirected graph $G = (V, E)$ with $|V| = n$ and $|E| = m$ and let $k \geq 1$ be an integer. Let $\delta_G(u, v)$ denote the distance between u and v in G. A graph $H = (V, E')$, where $E' \subseteq E$, is a *spanner* of stretch k (or, a k-spanner) of G if and only if $\delta_H(u, v) \leq k \cdot \delta_G(u, v)$ for every $u, v \in V$.

The notion of graph spanners was introduced in [24, 23] in the late 80's. It is known how to efficiently construct a $(2k-1)$-spanner of size $O(n^{1+1/k})$ [1], and this size-stretch tradeoff is conjectured to be tight. The interest in graph spanners stems from the fact that spanners are used explicitly or implicitly as key ingredients of various distributed applications, e.g., synchronizers [24], compact routing [25, 31], covers [2], dominating sets [11], distance oracles [3, 32], emulators and distance preservers [7], broadcasting [16], or near-shortest path algorithms [12, 13, 15]. Hence, understanding the properties of graph spanners and providing efficient algorithms for constructing them appear as a fundamental problem in distributed computing. Recent reviews of the literature on spanners can be found in [26, 35].

In this work we study the notion of *fault tolerant* spanners. A graph H is an f-vertex (resp. edge) fault tolerant k-spanner of G if for any set $F \subseteq V$ (resp. $F \subseteq E$) of size at most f and any pair of vertices $u, v \in V \setminus F$ (resp. $u, v \in V$) it satisfies that $\delta_{H \setminus F}(u, v) \leq k \cdot \delta_{G \setminus F}(u, v)$. (Here, and throughout the paper, $G \setminus F$ denotes the subgraph of G obtained by removal of the faults F.) For vertex faults, we present an f-vertex fault tolerant $(2k-1)$-spanner whose size is $O(f^3 k^{f+1} \cdot n^{1+1/k} \log^{1-1/k} n)$ (only slightly larger than the best known standard spanners). For edge faults, we present an f-edge fault tolerant $2k-1$ spanner with edge set of size $O(f \cdot n^{1+1/k})$ (only f times larger than standard spanners). Our results open many research directions on fault tolerant constructions and applications. Specifically, any of the above applications in which spanners were used in the past can now be considered in failure-prone settings, in which fault-tolerant solutions could be sought.

Background and Previous work Recently, the question of maintaining spanners in dynamic settings attracted much attention. Baswana and Sarkar [4] presented an algorithm for maintaining a graph spanner that supports both insertions and deletions of edges in poly-logarithmic amortized update time. Elkin [13, 14] presented a fully dynamic spanner for the distributed and the streaming models. In the geometric setting, where the vertices of the graph G are assumed to lie in Euclidean space, a $(1+\epsilon)$-spanner of size $O(n/\epsilon^d)$ can be constructed in $O(n \log n)$ time [34, 30, 17, 27]. For dynamic spanners in the geometric setting, Gao, Guibas and Nguyen [17] presented an algorithm that supports both insertions and deletions of points in $O(\log \Delta)$ time, where Δ is the aspect ratio of the point set (i.e., the ratio between the farthest pair of points to the closest pair of points). Roditty [27] showed how to obtain an update time that does not depend on the aspect ratio using a variation of the algorithm of [17]. Most recently, Roditty and Gottlieb [18, 19] presented two algorithms with $O(\text{polylog } n)$ update time.

The traditional fully dynamic model in which graph spanners were studied so far may be too pessimistic with respect to real world networks, where changes are fairly limited and the core of the network does not change frequently. For example, in a road network the possible changes to the network are rather limited. Some roads may be closed for short periods or a major junction may be temporarily blocked, but the basic structure of the network remains the same. In a standard computer network, some links may occasionally fail and even some routers may be temporarily out of service, but again the basic structure of the network remains unchanged.

The focus of this work is the study of fault tolerant spanners. The "fault tolerance" model lends itself naturally to the scenarios described above. In this model, the input is preprocessed so that after any f

failures, a fast recovery of the network information will be possible. For example, if f roads or f junctions are temporarily closed, we would still like to have a valid spanner of the current network. The idea, as in dynamic algorithms, is to preprocess the original data (namely, the input graph G) so that a fast recovery of information is possible. There are three important parameters when considering the fault tolerant model. The first is the running time of the preprocessing algorithm, the second is the size of the created data structure, and the third is the time that is needed to update the data structure once failures occur.

Fault tolerance aspects of various problems have attracted considerable attention lately. Pătraşcu and Thorup [22] considered the connectivity problem. They showed that it is possible to preprocess a graph in polynomial time and to obtain a linear size data structure that allows responding to connectivity queries in $O(f \cdot \text{polylog}(n))$ time after the failure of f arbitrary edges. In the context of the *all-pairs shortest paths* problem, Demetrescu, Thorup, Chowdhury and Ramachandran [9] showed that it is possible to preprocess a graph into a data structure that is capable of answering distance queries after a single vertex or edge failure. Bernstein and Karger [6] improved the running time of [9]. Very recently, Duan and Pettie [10] presented a data structure that is capable of answering distance queries after two vertex or edge failures.

Fault tolerant spanners were only studied in the context of geometric spanners. A decade ago, Levcopoulos, Narasimhan and Smid [20] introduced the notion of fault tolerant spanners. They presented an efficient algorithm for constructing an f-vertex fault tolerant *geometric* $(1+\epsilon)$-spanner, that is, given a set S of n points in $\mathbb{R}^d$, their algorithm finds a sparse graph H such that for every set $F \subseteq S$ of size f and any pair of points $u, v \in S \setminus F$, the distances in H satisfy $\delta_{H \setminus F}(u,v) \leq (1+\epsilon)|uv|$, where $|uv|$ is the Euclidean distance between u and v. A fault tolerant geometric spanner of improved size was presented by Lukovszki [21]. Finally, Czumaj and Zhao [8] presented a fault tolerant geometric spanner with optimal maximum degree and total weight. In [8] they raised as an open problem the question whether it is possible to obtain a fault tolerant spanner for an arbitrary weighted undirected graph.

In this paper we provide a positive answer to this question. Not only does such a fault tolerant spanner exist for general graphs, but as we show, its properties are almost identical to those of a standard spanner (i.e., one that does not tolerate any faults at all).

Our results This work addresses the design of both vertex and edge fault tolerant spanners. The main result of this paper is an efficient algorithm that constructs an f-vertex fault tolerant $(2k-1)$-spanner for a weighted undirected graph. The size of our spanner is $O(f^3 k^{f+2} n^{1+1/k} \log^{1-1/k} n)$. Our result is especially appealing in comparison to standard spanners as the stretch of our fault tolerant spanner is the same while its size is increased only by a factor of $f^3 k^{f+1}$ (ignoring logarithmic factors).

A natural approach to constructing an f-vertex fault tolerant spanner would be to construct, for every $F \subset V$ of size at most f, a spanner of $G \setminus F$ (*e.g.*, via some known algorithm for spanner construction), and to define the final spanner as their union. This approach may appear to be overly naive, as it would seem to cause an explosion in the size of the resulting spanner. Surprisingly, an algorithm that follows this spirit is exactly what we present. Our construction is based on the distance oracle construction of Thorup and Zwick [32]. An approximate distance oracle is a data structure of size $O(kn^{1+1/k})$ that answers approximate distance queries in $O(k)$ time. It approximates the distances up to a $2k-1$ multiplicative error. In our work, we weaken the construction of Thorup and Zwick, in the sense that it no longer remains a distance oracle, rather, it only holds properties of a standard spanner. The restricted construction combined with some other new ideas and a careful analysis yield our result.

The simple but clever algorithm presented by Thorup and Zwick [32] lies at the foundations of many important results. First, Thorup and Zwick [31] presented optimal routing schemes based on it. Roditty and Zwick [29] used it to obtain a dynamic algorithm to approximate all-pairs shortest paths. Later on Roditty, Thorup and Zwick [28] presented an efficient deterministic construction. Baswana and Sen [5] and Baswana and Kavitha [3] improved the running time of the construction algorithm in a variety of settings. In [33] Thorup and Zwick analyzed their construction in the context of additive spanners, concluding that

```
algorithm clusters(G(V,E), {A_0, ..., A_k}, k)

for every v ∈ V
    for i ← 0 to k − 1
        let δ(A_i, v) ← min{ δ(w, v) | w ∈ A_i }
        let p_i(v) ∈ A_i be such that δ(p_i(v), v) = δ(A_i, v)
    δ(A_k, v) ← ∞
for i ← 0 to k − 1
    for each w ∈ A_i \ A_{i+1}
        [ C(w) ← {v | δ(v, w) < δ(v, p_{i+1}(v))} ]
        C(w) ← {v | δ(v, w) < δ(v, p_{i+1}(v)) ∧ δ(v, w) ≤ k}
let C ← ⋃_w C(w)
return C
```

```
algorithm initialize(V, k)

A_0 ← V ; A_k ← φ
for i ← 1 to k − 1
    A_i ← sample(A_{i−1}, (n / log n)^{−1/k})
let A ← {A_0, ..., A_k}
return A
```

```
algorithm spanner(G(V,E), k)

A ← initialize(V, k)
C ← clusters(G(V,E), A, k)
return C
```

Figure 1: The algorithm of [32]. The first cluster definition (in parentheses [·]) is the definition of [32] and the second is our trimmed cluster definition.

their distance oracles provide also good additive spanners. Our result can be viewed as another unexpected application of the core ideas in [32].

We also present an f-**edge** fault tolerant $(2k-1)$-spanner with edge set of size is $O(f \cdot n^{1+1/k})$ (only f times larger than the standard lower bounds). As in the case of vertex faults, our result holds when the given graph is weighted.

The rest of this paper is organized as follows. In Section 2 we present our main result, namely, the construction of vertex fault tolerant spanners. To simplify our presentation, in Section 2 we consider only unweighed graphs G. We then turn to extend our results to the more involved case of weighted graphs and analyze the running time of our algorithm. Due to space limitations, the latter two steps are presented in Sections B and C of the Appendix. In Section 3 we present our algorithm for edge fault tolerant spanners (for weighted graphs). Finally in Section 4 we present a few concluding remarks.

2 Vertex fault tolerant spanner

In this section we present the main result of this paper, an algorithm for constructing an f-vertex fault tolerant spanner. One of the ingredients of our algorithm is a non-standard usage of the distance oracle construction of Thorup and Zwick [32]. That paper presents an algorithm that creates an *approximate distance oracle*, which is a data structure of size $O(kn^{1+1/k})$ that answers approximate distance queries in $O(k)$ time. It approximates the distances up to a $2k-1$ multiplicative error. The main ingredient of this data structure is a clever tree cover (which is also a spanner) for the graph. Hence the distance oracle is in particular a spanner. In the first part of this section we review the construction of [32]. Our presentation is biased towards our specific usage later on. We then present our algorithm and its analysis.

Let $G(V,E)$ be an unweighted undirected graph. (In Section B of the Appendix, we show how to extend our result for weighted graphs.) For each vertex $w \in V$, let $T(w)$ be a certain *shortest path* spanning tree of G rooted at w. Roughly speaking, the spanner of [32] consists of n *clusters*, each indexed by a vertex $w \in V$ and denoted by $C(w)$. Each such cluster $C(w)$ consists of a tree rooted at w that spans the set of vertices that are in $C(w)$. In [32] it is shown that the tree $C(w)$ is always a subtree of $T(w)$, and thus to simplify notation, we denote both the tree rooted at w and the subset of vertices it spans by $C(w)$. Finally, the edge set $\mathcal{C} = \cup_{w \in V} C(w)$ is defined to be the desired spanner. The algorithm of [32] is given in Figure 1.

Theorem 2.1 [32] *Algorithm* **spanner**$(G(V,E),k)$ *given in Figure 1 (with the clusters defined as in [32]) returns, with high probability, a* $(2k-1)$*-spanner of* $G(V,E)$ *with* $O(n^{1+1/k}\log^{1-1/k} n)$ *edges.*

The analysis of Thorup and Zwick in fact proved a stronger result, namely, that with high probability, the number of clusters in which every vertex participates is at most $O(n^{1/k}\log^{1-1/k} n)$. This property is very important to our construction as we will see later on.

We now describe the first change we make in the spanner algorithm of Thorup and Zwick, which is crucial for our algorithm. Specifically, we change the definition of $C(w)$. Our clusters $C(w)$ differ from those in [32] in the sense that our clusters are *trimmed* at depth k. Formally, our clusters are defined as $C(w) = \{v \mid \delta(v,w) < \delta(v,p_{i+1}(v)) \wedge \delta(v,w) \leq k\}$. In contrast, the original definition of [32] is $C(w) = \{v \mid \delta(v,w) < \delta(v,p_{i+1}(v))\}$. The vertex $p_{i+1}(v)$ is defined to be the closest vertex to v among the vertices of A_{i+1}, where ties are broken by the order of the sampling, that is, the vertex that survived more steps in the sampling process is chosen. This definition is the same as that of [32]. Notice that the new cluster definition can only affect the stretch of the spanner, as its size can only decrease as a result of the change.

Theorem 2.2 *Algorithm* **spanner**$(G(V,E),k)$ *given in Figure 1 (with the new cluster definition) returns, with high probability, a* $(2k-1)$*-spanner of* $G(V,E)$ *with* $O(n^{1+1/k}\log^{1-1/k} n)$ *edges.*

Proof: For every vertex $w \in V$, let $C_{\text{TRIM}}(w)$ be the trimmed cluster obtained from $C(w)$ by removing any vertex of it whose distance from w is more than k. It is shown in [32] (Lemma 3.3) that for every pair of vertices u and v there exists a vertex w such that (i) $u,v \in C(w)$ and (ii) the paths from u to w and from v to w satisfy that one is of length at most k times the distance between u and v and the other is of length at most $k-1$ times the distance between u and v. In particular, it must be that for every edge $(u,v) \in E$ there exists a vertex w whose cluster $C(w)$ contains both u and v such that the paths from u to w and from v to w satisfy that one is of length at most k and the other one is of length at most $k-1$. This implies not only that $u,v \in C(w)$ but also that $u,v \in C_{\text{TRIM}}(w)$. Since any edge is approximated with a path of length at most $2k-1$, the graph $\cup_{w\in V} C_{\text{TRIM}}(w)$ is a $(2k-1)$-spanner. ∎

Hereafter, for every $w \in V$, we denote its trimmed cluster by $C(w)$. We now present our algorithm for f-vertex fault tolerant spanners. By our definitions, an f-fault tolerant $(2k-1)$ spanner $\mathcal{C}$ for a graph $G(V,E)$ must contain, for every subset $F \subset V$ of size at most f, a $(2k-1)$ spanner for the graph $G \setminus F$.

A naive approach to solve this problem is to construct for every $F \subset V$ of size at most f, a spanner of $G \setminus F$ and to define the final spanner as their union. However, in such a solution, even for a single vertex fault, the spanner may contain all the edges of the graph and of course will be useless.

Surprisingly, following the spirt of this naive approach using the variation to the spanner of [32] discussed above (combined with some other new ideas presented later), we obtain an f-fault tolerant $(2k-1)$-spanner with only $O(f^3 k^{f+1} n^{1+1/k}\log^{1-1/k} n)$ edges. The crux of this approach lies in its analysis which is possible due to our trimmed cluster definition.

A high-level description of our algorithm is given in Figure 2. It receives as an input three parameters; a graph, an integer k for the desired stretch-space tradeoff and an integer f for the desired number of faults. Let $F \subset V$ be a set of size at most f. The algorithm constructs a $(2k-1)$-spanner $\mathcal{C}^F$ of $G \setminus F$, for any such F. We denote by $C^F(w)$ the cluster corresponding to w in $\mathcal{C}^F$. In order to ensure that the final f-vertex fault tolerant spanner $\mathcal{C}$ (which our algorithm outputs) is sparse, we design $C^F(w)$ to satisfy the following property.

Property 2.3 *For any* $F' \subset F$ *and any vertex* $v \in V$*, if the path* P *connecting* v *to* w *in* $C^{F'}(w)$ *does not include any vertices from the set* $F \setminus F'$*, then* P *appears in* $C^F(w)$ *as well.*

```
algorithm ft-spanner(G(V,E), k, f)

A ← initialize(V, k)
C^φ ← clusters(G(V,E), A, k)
C ← C^φ
for t = 1 to f
    for every F ⊆ V of size at most t
        C^F ← clusters(G \ F, A, k)
        C ← C ⋃ C^F
return C
```

Figure 2: Our algorithm for constructing an f-vertex fault tolerant $(2k-1)$-spanner

To ensure this property, when constructing $C^F(w)$ for some vertex $w \in V$ and a set F, we enforce the following rule. Let $V = \{v_1, .., v_n\}$. Assume the cluster is already constructed up to depth $r-1$, that is, there is a path to every vertex at distance of at most $r-1$ from w that belongs to the cluster. We now construct level r of the cluster. Let x be a vertex of level r. Let i be the smallest index such that v_i belongs to level $r-1$ and there is an edge between v_i and x in $G \setminus F$. We set v_i to be the parent of x.

We now show that if this rule is applied then Property 2.3 is satisfied. We stress that, as before, throughout we denote by $C^F(w)$ both the set of vertices in the cluster and its corresponding spanning tree obtained by this procedure.

Lemma 2.4 *If the clusters of* **ft-spanner**$(G(V,E), k, f)$ *are constructed using the rule described above then they satisfy Property 2.3.*

Proof: Let $F' \subset F$, let $v \in C^{F'}(w)$ and let P be the shortest path that connects v to w in $C^{F'}(w)$. Assume that P does not include vertices from the set $F \setminus F'$. Thus, the path P still exists in $G \setminus F$ and is a shortest path between v and w. Let Q be some other shortest path from v to w in $G \setminus F$. Since P is a shortest path in $G \setminus F'$, Q is also a shortest path in $G \setminus F'$.

It follows directly from the definition of our clusters that if a vertex v is in a cluster $C(w)$ then every vertex on any shortest path from w to v is also in $C(w)$. Applying this in our context yields that all the vertices on the paths P and Q appear in both $C^{F'}(w)$ and $C^F(w)$.

Let $P = (x_0 = w, x_1, \ldots, x_r, v)$ and $Q = (y_0 = w, y_1, \ldots, y_r, v)$, and let i be the largest index such that $x_i \neq y_i$. Let $x_i = v_j$ and $y_i = v_\ell$. For the sake of contradiction, assume that Q is the path that was chosen by the algorithm to connect v to w in $C^F(w)$. This means that when the algorithm chose a parent for y_{i+1}, it chose v_ℓ, and since $y_{i+1} = x_{i+1}$, it follows that v_ℓ was chosen over v_j and hence $\ell < j$ by our rule. However, this leads to contradiction since P is the path in $C^{F'}(w)$ that was constructed by following the same rule and it must be that $\ell > j$. □

Before we turn to the analysis of our algorithm, we discuss the second change that needs to be applied to the algorithm of [32]. In our construction we would like that, with high probability, for every v and every subset $F \subset V$ of size at most f, the number of vertices $w \in V$ such that $v \in C^F(w)$ be bounded by $O(fn^{1/k} \log^{1-1/k} n)$. This can be guaranteed using the exact same analysis of [32] when one slightly increasing the sample probability in **initialize**. Namely, we prove the following (in the Appendix).

Proposition 2.5 *Increasing the sample probability in* **initialize** *from* $(n/\ln n)^{-1/k}$ *to* $(f+3)^{1/(k-1)}(\frac{n}{\ln n})^{-1/k}$ *ensures that with probability at least* $1 - 1/n$*, for every* $v \in V$*, and every* $F \subset V$ *of size at most* f*, the number of clusters* $C^F(w)$ *that contain* v *is bounded by* $O(f \cdot n^{1/k} \ln^{1-1/k} n)$*.*

Proposition 2.5 implies that, w.h.p, the spanners $\mathcal{C}^F$ we construct in **ft-spanner**$(G(V,E),k,f)$ are each of size $O(fn^{1+1/k}\log^{1-1/k} n)$. We can now turn to show that the union of all these $\Omega(n^f)$ spanners is not much larger than the size of a single one. We do that in two steps. First, as a warm-up that demonstrates our ideas in the simplest possible setting, we analyze the case of a single fault, and then we turn to the general case. We stress that in both cases, our modified definition for $C(w)$, which involves "trimming at depth k," plays a major role in our analysis.

2.1 Warmup: 1-vertex fault tolerant spanners

As a warm-up we analyze the algorithm for 1 fault, that is **ft-spanner**$(G(V,E),k,1)$. The ideas and proof techniques used in this section will be extended to deal with f-faults when we analyze the algorithm **ft-spanner**$(G(V,E),k,f)$ in Section 2.2.

Theorem 2.6 *Algorithm* **ft-spanner**$(G(V,E),k,1)$ *given in Figure 2 returns, with high probability, a* 1*-fault tolerant spanner of* $G(V,E)$ *with stretch* $2k-1$ *and* $O(k^2n^{1+1/k}\log^{1-1/k} n)$ *edges.*

Proof: Let $\mathcal{C}^\phi$ be the spanner returned by the execution of **clusters**$(G,\mathcal{A},k)$. Here the superscript 'ϕ' refers to the set of vertex faults considered (which is currently empty). Let $\delta^\phi(u,v)$ denote the length of the shortest path between u and v in G. Let $\delta^\phi(A_i,v)=\min\{\,\delta^\phi(w,v)\mid w\in A_i\}$. Let $p_i^\phi(v)\in A_i$ be such that $\delta^\phi(p_i^\phi(v),v)=\delta^\phi(A_i,v)$. Let $C^\phi(w)=\{v\mid \delta^\phi(v,w)<\delta^\phi(v,p_{i+1}^\phi(v))\wedge\delta^\phi(v,w)\le k\}$. For, $x\in V$, consider the execution of **clusters**$(G\setminus\{x\},\mathcal{A},k)$ preformed while running **ft-spanner**$(G(V,E),k,1)$. Let $\delta^x(u,v)$ denote the length of the shortest path between u and v in $G\setminus\{x\}$. Let $\delta^x(A_i,v)=\min\{\,\delta^x(w,v)\mid w\in A_i\}$. Let $p_i^x(v)\in A_i$ be such that $\delta^x(p_i^x(v),v)=\delta^x(A_i,v)$. Let $C^x(w)=\{v\mid \delta^x(v,w)<\delta^x(v,p_{i+1}^x(v))\wedge\delta^x(v,w)\le k\}$. Finally, let $\mathcal{C}^x=\bigcup_w C^x(w)$.

We first bound the number of edges in the spanner $\mathcal{C}$ returned by algorithm **ft-spanner**$(G(V,E),k,1)$. Notice that the spanner $\mathcal{C}$ includes the union of the spanner $\mathcal{C}^\phi$ and the additional spanners $\mathcal{C}^x$ (for $x\in V$). By our preliminary discussion, each such spanner in itself has at most $O(n^{1+1/k}\log^{1-1/k} n)$ edges (recall that in this section $f=1$). In what follows we show that the size of the union of these spanners is not much larger than that.

To this end, we analyze the number of edges in the edge set of $\mathcal{C}\setminus\mathcal{C}^\phi$, namely, the number of edges added to the initial spanner $\mathcal{C}^\phi$ during the execution of algorithm **ft-spanner**$(G(V,E),k,1)$. We use the following definition. For a vertex x, let $C^x_{\text{NEW}}(w)\subseteq C^x(w)$ be the set of vertices v for which the path connecting v to w in $C^x(w)$ does not appear in $C^\phi(w)$. To bound the number of edges in $\mathcal{C}$, it suffices to bound the number of vertices v in $\bigcup_{w\in V}\bigcup_{x\in V}C^x_{\text{NEW}}(w)$. This follows from Property 2.3, namely, from the fact that only such vertices v add an edge to $\mathcal{C}\setminus\mathcal{C}^\phi$, i.e., the edge connecting v to its parent in the corresponding cluster $C^x(w)$.

Call a tuple (v,w,x) *costly* iff $v\in C^x_{\text{NEW}}(w)$. The number of edges in $\mathcal{C}$ may be bounded by the size of $\mathcal{C}^\phi$ plus the number of costly tuples. We show that the latter is bounded by $O(k^2n^{1+1/k}\log^{1-1/k} n)$.

Let v be any vertex in V. Let i be an integer between 1 and k. In what follows we consider only costly tuples (v,w,x) for which $w\in A_i\setminus A_{i+1}$. Later, our bound can be multiplied by kn to obtain our assertion (a multiplicative factor of k for each of the sets $A_0\setminus A_1,\ldots,A_{k-1}\setminus A_k$, and a factor of n for each $v\in V$).

We consider two cases. In the first case, consider tuples (v,w,x) for which $v\in C^\phi(w)$. We claim that in this case the vertex x must lie on the path P_{in} connecting v and w in $C^\phi(w)$, as otherwise, by Property 2.3, the path P_{in} will appear identically in $C^x(w)$, which in turn will imply that $v\notin C^x_{\text{NEW}}(w)$.

By Proposition 2.5, the number of vertices w for which $v\in C^\phi(w)$ is bounded by $O(n^{1/k}\log^{1-1/k} n)$. In addition, for every such w there are at most k vertices x on the path between v and w in $C^\phi(w)$. The latter

follows by our definition of $C(w)$, which guarantees that $\delta(v,w) \leq k$ for every $v \in C(w)$. We conclude that a total of at most $O(kn^{1/k}\log^{1-1/k} n)$ costly tuples are accounted for in this case.

We now turn to the case in which $v \notin C^{\phi}(w)$ and show that in any costly tuple (v,w,x), the vertex x must be on the path P_{out} that connects v to $p^{\phi}_{i+1}(v)$ in G. Recall that $C^{\phi}(w) = \{v \mid \delta^{\phi}(v,w) < \delta^{\phi}(v,p^{\phi}_{i+1}(v)) \wedge \delta^{\phi}(v,w) \leq k\}$. Thus, either $\delta^{\phi}(v,w) \geq \delta^{\phi}(v,p^{\phi}_{i+1}(v))$ or $\delta^{\phi}(v,w) > k$. We are assuming that (v,w,x) is a costly tuple, *i.e.*, $v \in C^{x}_{\text{NEW}}(w) \subseteq C^{x}(w) = \{v \mid \delta^{x}(v,w) < \delta^{x}(v,p^{x}_{i+1}(v)) \wedge \delta^{x}(v,w) \leq k\}$, which implies $\delta^{x}(v,w) \leq k$ and in turn $\delta^{\phi}(v,w) \leq \delta^{x}(v,w) \leq k$. Here we use $\delta^{\phi}(v,w) \leq \delta^{x}(v,w)$, which follows from the fact that distances in $G \setminus \{x\}$ are at least as large as those in G. It remains to consider the case $\delta^{\phi}(v,p^{\phi}_{i+1}(v)) \leq \delta^{\phi}(v,w) \leq \delta^{x}(v,w) < \delta^{x}(v,p^{x}_{i+1}(v))$. This, in turn, implies that $\delta^{\phi}(v,p^{\phi}_{i+1}(v))$ is strictly less than $\delta^{x}(v,p^{x}_{i+1}(v))$, which can only happen if x is on the path P_{out} specified above. Namely, x must be one of at most k vertices in the path P_{out} (note that the discussion above implies that the length of P_{out} is indeed bounded by k).

To complete our proof, we recall that for each such x, by Proposition 2.5, there are at most $O(n^{1/k}\log^{1-1/k} n)$ vertices w for which v is in $C^{x}(w)$, and hence at most $O(n^{1/k}\log^{1-1/k} n)$ vertices w for which v is in $C^{x}_{\text{NEW}}(w)$. Thus, all in all, the number of costly tuples accounted for in this case is bounded again by $O(kn^{1/k}\log^{1-1/k} n)$.

This completes our proof bounding the number of edges in $\mathcal{C}$. The bound on the stretch of $\mathcal{C}$ follows directly from the properties of the spanners $\mathcal{C}^{\phi}$ and $\mathcal{C}^{x}$ outlined in Theorem 2.2. □

2.2 The general case: f-vertex fault tolerant spanners

Theorem 2.7 *Algorithm* **ft-spanner**$(G(V,E),k,f)$ *given in Figure 2 returns, with high probability, an f-fault tolerant spanner of $G(V,E)$ with stretch $2k-1$ and $O(f^3k^{f+1}n^{1+1/k}\log^{1-1/k} n)$ edges.*

Proof: Let $\mathcal{C}^{\phi}$ be the spanner returned by the execution of **clusters**$(G,\mathcal{A},k)$. Let $F \subseteq V$. Consider the execution **clusters**$(G \setminus F,\mathcal{A},k)$ preformed inside the main loop of **ft-spanner**$(G(V,E),k,f)$. Let $\delta^{F}(u,v)$ denote the length of the shortest path between u and v in $G \setminus F$. Let $\delta^{F}(A_i,v) = \min\{\delta^{F}(w,v) \mid w \in A_i\}$. Let $p^{F}_{i}(v) \in A_i$ be such that $\delta^{F}(p^{F}_{i}(v),v) = \delta^{F}(A_i,v)$. Let $C^{F}(w) = \{v \mid \delta^{F}(v,w) < \delta^{F}(v,p^{F}_{i+1}(v)) \wedge \delta^{F}(v,w) \leq k\}$. Finally, let $\mathcal{C}^{F} = \bigcup_{w} C^{F}(w)$.

Recall that the algorithm **clusters**$(G \setminus F,\mathcal{A},k)$ satisfies Property 2.3, that is, for a vertex $v \in C^{F'}(w)$ if the path P that connects v to w in $C^{F'}(w)$ does not include any vertex from the set $F \setminus F'$ then P also connects v to w in $C^{F}(w)$.

Thus, in order to analyze the exact upper bound on the size of our spanner, we only need to count the new connections that are formed at each stage. To this end, we present the following central definition, to be used throughout the rest of the proof.

Definition 2.8 *For a subset F of faults, let $C^{F}_{NEW}(w) \subseteq C^{F}(w)$ be the set of vertices v for which the path connecting v to w in $C^{F}(w)$ does not appear in $C^{F\setminus\{x\}}$ for any $x \in F$.*

Let $\mathcal{C}$ be the subgraph returned by algorithm **ft-spanner**$(G(V,E),k,f)$. To bound the number of edges in $\mathcal{C}$, it suffices to bound the number of vertices v in

$$\bigcup_{w \in V} \bigcup_{(F \subseteq V \text{ and } |F| \leq f)} C^{F}_{\text{NEW}}(w) \ .$$

Hence, it suffices to bound the number of tuples (v,w,F) that satisfy $|F| \leq f$ and $v \in C^{F}_{\text{NEW}}(w)$. This is exactly what we do next. Specifically, for any vertex $v \in V$ and any vertex $w \in A_i \setminus A_{i+1}$ for which

$v \in C^F_{\text{NEW}}(w)$ and $|F| = f$, we show that F has a very restricted structure and must be one of few different subsets of V. We then proceed to show how this claim will conclude our proof.

Throughout the discussion below we only consider triplets (v, w, F) for a specific vertex v, a specific value of i (which will imply that $w \in A_i \setminus A_{i+1}$) and a set F of size f. Thus, to obtain the final bound we will have to multiply the obtained bound by n (so as to count the cost of all the vertices), by k (for all the sets $A_0 \setminus A_1, \ldots, A_{k-1} \setminus A_k$) and by f (for all the possible set sizes).

Claim 2.9 *Let $F_t \subseteq V$ such that $|F_t| = t$. If $v \in C^{F_t}(w)$ for $w \in A_i \setminus A_{i+1}$, then the number of tuples (v, w, F) for which $v \in C^F_{NEW}(w)$ and $F_t \subseteq F$ is bounded by k^{f-t}.*

Proof: Let w be as defined in the claim. Let $F = F_t \cup \{u_1, \ldots, u_{f-t}\}$. Assume that $v \in C^F_{\text{NEW}}(w)$. We now show that there is a small number of extensions $\{u_1, \ldots, u_{f-t}\}$ that may be added to F_t to obtain F.

We first claim that as $v \in C^{F_t}(w)$ it must be the case that F includes a vertex on the path P_{in} between v and w in $C^{F_t}(w)$. If this is not the case then it follows from Property 2.3 that for every F' that satisfies $F_t \subseteq F' \subset F$ the path P_{in} is in $C^{F'}(w)$ and in particular there exists $x \in F$ such that both $v \in C^{F \setminus \{x\}}(w)$ and x is not in P_{in}. This implies that $v \notin C^F_{\text{NEW}}(w)$ which yields a contradiction. Thus, we can conclude that there exists a vertex of F in P_{in}. Assume, w.l.o.g, that $u_1 \in F$ is this vertex and let $F_{t+1} = F_t \cup \{u_1\}$.

We now consider the cluster $C^{F_{t+1}}(w)$. There are two possible scenarios, the first is that as before $v \in C^{F_{t+1}}(w)$ and the second is that $v \notin C^{F_{t+1}}(w)$. If v is in $C^{F_{t+1}}(w)$ then from the same arguments as before it must be that F includes a vertex from the path connecting v to w in $C^{F_{t+1}}(w)$. As in the proof of Theorem 2.6 , in the scenario that $v \notin C^{F_{t+1}}(w)$ it holds that $\delta^{F_{t+1}}(v, p^{F_{t+1}}_{i+1}(v)) \leq \delta^{F_{t+1}}(v, w) \leq \delta^F(v, w) < \delta^F(v, p^F_{i+1}(v))$ and $\delta^F(v, w) \leq k$. Namely, it must be that there is a vertex from F on the path that connects v to $p^{F_{t+1}}_{i+1}(v)$ in $G \setminus F_{t+1}$. Otherwise, the path that connects v to $p^{F_{t+1}}_{i+1}(v)$ is not affected by the deletion of the vertices of the set $F \setminus F_{t+1}$ and is still valid in the graph $G \setminus F$. The distance between v and w in $G \setminus F$ can only get larger with respect to the distance between v and w in $G \setminus F_{t+1}$ (recall that $F_{t+1} \subseteq F$). On the other hand, the distance between v and $p^{F_{t+1}}_{i+1}(v)$ would remain the same. Since $v \notin C^{F_{t+1}}(w)$, by definition, this would imply that $v \notin C^F(w)$ and obviously $v \notin C^F_{\text{NEW}}(w)$ which yields a contradiction.

We thus conclude that there must be a vertex of F that is either on the path that connects v to w if $v \in C^{F_{t+1}}(w)$ or on the path that connects v to $p^{F_{t+1}}_{i+1}(v)$ if $v \notin C^{F_{t+1}}(w)$. In each of these two possible scenarios there are at most k vertices that can be chosen. Assume, w.l.o.g, that $u_2 \in F$ is this vertex and let $F_{t+2} = F_{t+1} \cup \{u_2\}$.

We continue in a similar manner and define F_{t+3} which is an extension of F_{t+2} by one of k vertices defined by v, w and F_{t+2}; and in general we define F_{t+j} which is an extension of F_{t+j-1} by one of $2k$ vertices defined by v, w and F_{t+j-1}. In each iteration, the number of possible subsets F_{t+j} increases by a multiplicative factor of k. All in all, we conclude that the number of possible subsets F is bounded by the expression given in the claim. □

Let $v \in V$. Consider a triplet (v, w, F) for which $v \in C^F_{\text{NEW}}(w)$. We now bound the number of such triplets when w is assumed to be in $A_i \setminus A_{i+1}$ and F is assumed to have size f.

We start with the cluster set $\mathcal{C}^\phi$. We consider the case that $v \in C^\phi(w)$ and the case that $v \notin C^\phi(w)$. Consider the vertices w such that $v \in C^\phi(w)$. It follows from Proposition 2.5 that there are only $O(fn^{1/k} \log^{1-1/k} n)$ such vertices w from $A_i \setminus A_{i+1}$ for which v is in their cluster. Here, and throughout the proof we assume that Proposition 2.5 indeed holds (which happens with high probability over the sets A_i). For each one of these vertices it follows from Claim 2.9 that there are at most k^f possible sets F that satisfy $v \in C^F_{\text{NEW}}(w)$.

We now consider the vertices w for which $v \notin C^\phi(w)$. As in the proof of Claim 2.9, since $v \in C^F_{\text{NEW}}(w)$ it must be that the set F includes one of the k vertices on the path between v and $p^\phi_{i+1}(v)$ in G, as otherwise,

the path that connects v to $p^{\phi}_{i+1}(v)$ is not affected by the deletion of the set F and is valid in $G \setminus F$. The distance between v and w in $G \setminus F$ can only get larger with respect to the distance between v and w in G. Since $v \notin C^{\phi}(w)$, by definition, it cannot be that $v \in C^{F}(w)$ and obviously it cannot be that $v \in C^{F}_{\text{NEW}}(w)$ which yields a contradiction. We conclude that F must include one of the k vertices on the path that connects v to $p^{\phi}_{i+1}(v)$. Let u_1 be one such vertex, and consider the set $F_1 = \{u_1\}$.

As before, we consider two cases. First consider the vertices w such that $v \in C^{F_1}(w)$. Again, it follows from Proposition 2.5 that there are only $O(fn^{1/k}\log^{1-1/k} n)$ such vertices w from $A_i \setminus A_{i+1}$ for which v is in their cluster. For each one of these vertices, by Claim 2.9, there are at most k^{f-1} tuples (v, w, F), where $F_1 \subseteq F$, for which $v \in C^{F}_{\text{NEW}}(w)$. There are k possible values for F_1. Summing over all possible values for F_1 results in at most k^f tuples (v, w, F) for which $v \in C^{F}_{\text{NEW}}(w)$.

Consider any other vertex w for which $v \notin C^{F_1}(w)$. As before, we now claim that in any tuple (v, w, F), where $F_1 \subseteq F$, for which $v \in C^{F}_{\text{NEW}}(w)$ it must be the case that F includes one of the k vertices on the path that connects between v and $p^{F_1}_{i+1}(v)$ in $G \setminus F_1$, as otherwise, the path that connects v to $p^{F_1}_{i+1}(v)$ is not affected by the deletion of the set F and is valid in $G \setminus F$. The distance between v and w in $G \setminus F$ can only get larger with respect to the distance between v and w in $G \setminus F_1$ and since $v \notin C^{F_1}(w)$ it cannot be that $v \in C^{F}(w)$ and obviously it cannot be that $v \in C^{F}_{\text{NEW}}(w)$ which yields a contradiction. We conclude that F must include one of the k vertices on the path that connects v to $p^{F_1}_{i+1}(v)$. Let u_2 be one such vertex, we can now set $F_2 = F_1 \cup \{u_2\}$.

For a general iteration j we have the case that $v \in C^{F_{j-1}}(w)$ and the case that $v \notin C^{F_{j-1}}(w)$.

For the case that $v \in C^{F_{j-1}}(w)$ it follows from Proposition 2.5 that there are only $O(fn^{1/k}\log^{1-1/k} n)$ such vertices w from $A_i \setminus A_{i+1}$ that v is in their clusters. Using Claim 2.9, it follows that there are k^{f-j+1} tuples (v, w, F), where $F_{j-1} \subseteq F$, for which $v \in C^{F}_{\text{NEW}}(w)$. There are k^{j-1} possible values for F_{j-1}. Summing over all possible values for F_{j-1} results in at most k^f tuples (v, w, F) for which $v \in C^{F}_{\text{NEW}}(w)$. For the second case we define the set F_j to be an extension of F_{j-1} by one of at most k vertices on the path that connects between v and $p^{F_{j-1}}_{i+1}(v)$ in $G \setminus F_{j-1}$.

In our last step, once F_{f-1} has been defined, our first case yields an addition of k^f tuples. For our second case, we notice that F_f may have k^f different values. For each possible value F, it follows from Proposition 2.5 that there are at most $O(fn^{1/k}\log^{1-1/k})$ corresponding vertices w such that $w \in A_i \setminus A_{i+1}$ and $v \in C^{F}(w)$ (and in particular $v \in C^{F}_{\text{NEW}}(w)$). Thus any tuple (v, w, F) for which $v \in C^{F}_{\text{NEW}}(w)$ that has not been counter for so far must be one of $O(fk^f n^{1/k}\log^{1-1/k})$ corresponding tuples.

All in all, we have counted $\left(\sum_{i=1}^{f} k^i k^{f-i} + k^f\right) O(fn^{1/k}\log^{1-1/k} n) = O(f^2 k^f n^{1/k}\log^{1-1/k} n)$ tuples (v, w, F) for which $v \in C^{F}_{\text{NEW}}(w)$. Multiplying this by nfk as discussed in the beginning of the proof yields our assertion. □

3 Edge fault-tolerant Spanners

In this section we describe our algorithm for creating an f-edge fault tolerant spanner. Our algorithm presented here is for weighted undirected graphs G. As mentioned before, it is possible to efficiently construct a $(2k-1)$-spanner of size $O(n^{1+1/k})$. In our construction of f-edge fault tolerant spanners we may use *any* such spanner construction. Other than the size and stretch of the resulting spanner, we do not rely on any other of its properties. Therefore, we can use any construction of spanners that guarantees a resulting spanner with stretch $(2k-1)$ and size $O(n^{1+1/k})$.

The algorithm is given in Figure 3. The algorithm consists of $f+1$ iterations. Let E^{SP} be the set of edges added to the spanner so far. At the beginning of the algorithm initialize it to be empty. In each iteration, we build a $(2k-1)$-spanner for the graph $G \setminus E^{\text{SP}}$ via the procedure **spanner**. At the end of each iteration

```
algorithm edge-ft-spanner(G(V,E), k, f)

E^SP ← ∅
for i = 1 to f + 1 do:
    (V, E_i^SP) = spanner(G\E^SP, k)
    E^SP = E^SP ∪ E_i^SP
return H ← (V, E^SP)
```

Figure 3: Our algorithm for constructing an f-edge fault tolerant $(2k-1)$-spanner

we add the edges of the current $(2k-1)$-spanner to E^{SP}. After the last iteration, we return $H(V, E^{\text{SP}})$ which is the required f-edge fault tolerant spanner.

As mentioned above, the resulting subgraph in each invocation of procedure **spanner** returns a $(2k-1)$-spanner of size $O(n^{1+1/k})$. As we invoke procedure **spanner** $f+1$ times, the total number of edges in the resulting spanner is $O(fn^{1+1/k})$. We now show that H is indeed an f-edge fault tolerant $2k-1$ spanner.

Lemma 3.1 *For every subset $E' \subseteq E$, where $|E'| \leq f$, the subgraph $H' = (V, E^{SP} \setminus E')$ is a $(2k-1)$-spanner of the graph $G' = (V, E \setminus E')$.*

Proof: Consider a subset $E' \subseteq E$, where $|E'| \leq f$. Let $H = (V, E^{\text{SP}})$ be the spanner returned by the algorithm. Consider an edge $e \in E \setminus E'$ that is not included in the spanner H. It suffices to show that H' contains an alternative path whose *length* is at most $(2k-1)$ times e's weight. Here, the length of a path is the sum of its edge weights. Let H_i be the $(2k-1)$ spanner added during the i'th iteration. Notice that the edges of the $(2k-1)$ spanner H_i are disjoint for $1 \leq i \leq f+1$. The edge e was not included in each iteration i for $1 \leq i \leq f+1$. Therefore, each H_i contains an alternate path whose length is at most $(2k-1)$ times e's weight. Hence, there are $f+1$ disjoint alternative paths of length at most $(2k-1)$ times e's weight in H. As $|E'| \leq f$, there must be at least one alternative path left in H' of length at most $(2k-1)$ times e's weight. ∎

We thus conclude:

Theorem 3.2 *For every f, k, and weighted graph $G(V,E)$ where $|V| = n$, one can efficiently construct an f-edge fault tolerant $(2k-1)$ spanner with $O(fn^{1+1/k})$ edges.*

4 Concluding remarks

In this paper we study the construction of both vertex and edge fault tolerant spanners. We present fault tolerant $(2k-1)$ spanners of size only slightly larger than that of the best known standard $(2k-1)$ spanners. The many applications of spanners as a key ingredient in the design of distributed algorithms, naturally raise the question if such applications still hold in the failure-prone setting. Being such fundamental graph structures, our study of spanners in the context of fault tolerance opens the door to several intriguing questions that now seem to be within reach.

References

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

[2] B. Awerbuch, B. Berger, L. Cowen, and David Peleg. Near-linear cost sequential and distributed constructions of sparse neighborhood covers. In *34th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 638–647, 1993.

[3] S. Baswana and T. Kavitha. Faster algorithms for approximate distance oracles and all-pairs small stretch paths. In *FOCS*, pages 591–602. IEEE Computer Society, 2006.

[4] S. Baswana and S. Sarkar. Fully dynamic algorithms for graph spanners with poly-logaritmic update time. In *SODA*, pages 672–681. ACM and SIAM, 2008.

[5] S. Baswana and S. Sen. Approximate distance oracles for unweighted graphs in expected $O(n^2)$ time. *ACM Transactions on Algorithms*, 2(4):557–577, October 2006.

[6] A. Bernstein and D. Karger. Improved distance sensitivity oracles via random sampling. In *SODA '08: Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 34–43, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics.

[7] B. Bollobás, D. Coppersmith, and M. Elkin. Sparse distance preservers and additive spanners. In *14th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 414–423, 2003.

[8] A. Czumaj and H. Zhao. Fault-tolerant geometric spanners. *Discrete & Computational Geometry*, 32:2004, 2003.

[9] C. Demetrescu, M. Thorup, R. Alam Chowdhury, and V. Ramachandran. Oracles for distances avoiding a failed node or link. *SIAM J. Comput*, 37(5):1299–1318, 2008.

[10] R. Duan and S. Pettie. Dual-failure distance and connectivity oracles. In *SODA*, 2009.

[11] D. Dubhashi, A. Mei, A. Panconesi, J. Radhakrishnan, and A. Srinivasan. Fast distributed algorithms for (weakly) connected dominating sets and linear-size skeletons. *J. Computer and System Sciences*, 71:467–479, 2005.

[12] M. Elkin. Computing almost shortest paths. *ACM Trans. Algorithms*, 1(2):283–323, 2005.

[13] M. Elkin. A near-optimal distributed fully dynamic algorithm for maintaining sparse spanners. In Indranil Gupta and Roger Wattenhofer, editors, *Proceedings of the Twenty-Sixth Annual ACM Symposium on Principles of Distributed Computing (26th PODC'07)*, pages 185–194, Portland, Oregon, USA, August 2007. ACM SIGACT/SIGOPS.

[14] M. Elkin. Streaming and fully dynamic centralized algorithms for constructing and maintaining sparse spanners. In Lars Arge, Christian Cachin, Tomasz Jurdzinski, and Andrzej Tarlecki, editors, *Automata, Languages and Programming, 34th International Colloquium, ICALP 2007, Wroclaw, Poland, July 9-13, 2007, Proceedings*, volume 4596 of *Lecture Notes in Computer Science*, pages 716–727. Springer, 2007.

[15] M. Elkin and J. Zhang. Efficient algorithms for constructing $(1+\epsilon, \beta)$-spanners in the distributed and streaming models. In *23rd ACM Symp. on Principles of Distributed Computing (PODC)*, pages 160–168, 2004.

[16] A. M. Farley, A. Proskurowski, D. Zappala, and K. Windisch. Spanners and message distribution in networks. *Discrete Applied Mathematics*, 137(2):159–171, 2004.

[17] J. Gao, L. Guibas, and A. Nguyen. Deformable spanners and applications. In *ACM Symposium on Computational Geometry*, 2004.

[18] L. Gottlieb and L. Roditty. Improved algorithms for fully dynamic geometric spanners and geometric routing. In *Proc. of 19th SODA*, pages 591–600, 2008.

[19] L. Gottlieb and L. Roditty. An optimal dynamic spanner for doubling metric spaces. In *Proc. of 16th ESA*, pages 478–489, 2008.

[20] C. Levcopoulos, G. Narasimhan, and M. Smid. Efficient algorithms for constructing fault-tolerant geometric spanners. In *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 186–195, New York, NY, USA, 1998. ACM.

[21] T. Lukovszki. New results of fault tolerant geometric spanners. In *WADS '99: Proceedings of the 6th International Workshop on Algorithms and Data Structures*, pages 193–204, London, UK, 1999. Springer-Verlag.

[22] M. Pătraşcu and M. Thorup. Planning for fast connectivity updates. In *Proc. 48th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 263–271, 2007.

[23] D. Peleg and A. A. Schäffer. Graph spanners. *J. Graph Theory*, pages 99–116, 1989.

[24] D. Peleg and J. D. Ullman. An optimal synchronizer for the hypercube. *SIAM J. Computing*, 18(4):740–747, 1989.

[25] D. Peleg and E. Upfal. A trade-off between space and efficiency for routing tables. *J. ACM*, 36(3):510—530, 1989.

[26] S. Pettie. Low distortion spanners. In *34th Int. Colloq. on Automata, Languages and Programming (ICALP)*, pages 78–89, 2007.

[27] L. Roditty. Fully dynamic geometric spanners. In *ACM Symposium on Computational Geometry*, 2007.

[28] L. Roditty, M. Thorup, and U. Zwick. Deterministic constructions of approximate distance oracles and spanners. In *Proc. of 32th ICALP*, pages 261–272, 2005.

[29] L. Roditty and U. Zwick. On dynamic shortest paths problems. In *ESA: Annual European Symposium on Algorithms*, 2004.

[30] J. S. Salowe. Constructing multidimensional spanner graphs. *Int. J. Comput. Geometry Appl*, 1(2):99–107, 1991.

[31] M. Thorup and U. Zwick. Compact routing schemes. In *SPAA '01: Proceedings of the thirteenth annual ACM symposium on Parallel algorithms and architectures*, pages 1–10, New York, NY, USA, 2001. ACM.

[32] M. Thorup and U. Zwick. Approximate distance oracles. *Journal of the ACM*, 52(1):1–24, 2005.

[33] M. Thorup and U. Zwick. Spanners and emulators with sublinear distance errors. In *SODA '06: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 802–809, New York, NY, USA, 2006. ACM.

[34] P. M. Vaidya. A sparse graph almost as good as the complete graph on points in K dimensions. *Discrete & Computational Geometry*, 6:369–381, 1991.

[35] D. P. Woodruff. Lower bounds for additive spanners, emulators, and more. In *47th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 389–398, 2006.

APPENDIX

A Proof of Proposition 2.5

Proof: Our proof is strongly based on Lemma 3.5 from [32]. We set the sampling probability to be $p = (f+3)^{1/(k-1)}(\frac{n}{\ln n})^{-1/k}$. Consider a node $v \in V$ and consider a set $F \subset V$ such that $|F| \leq f$. Let $n' = |V \setminus F|$. We assume that $f \leq n/2$, therefore $n' > n/2$.

As discussed in [32], the number of vertices $w \in V$ such that $v \in C^F(w)$ is stochastically dominated by the sum $\sum_{i=1}^{k-1} X_i$, where X_i for $0 \leq i \leq k-1$ are k random variables distributed as follows. The variable X_{k-1} is binomially distributed with parameters n' and p^{k-1}. The rest of the variables X_i for $0 \leq i \leq k-2$ are geometric random variables with parameter p. Moreover, these k random variables are independent.

Let X be a binomial random variable such that $E[X] = \mu$. Chernoff's bounds say the following:

$$Pr[X > (1+\delta)\mu] < \left[\frac{\exp(\delta)}{(1+\delta)^{1+\delta}}\right]^{\mu},$$

$$Pr[X < (1-\delta)\mu] < \exp(-\mu\delta^2/2).$$

Notice that, $E[X_{k-1}] = n'p^{k-1} = n'(f+3)n^{1/k-1}\ln^{1-1/k} n$.

We now use the first Chernoff inequality, with $\delta = 3$ and $\mu = n'(f+3)n^{1/k-1}\ln^{1-1/k} n$,

$$\begin{aligned} Pr[X_{k-1} > 4(f+3)n^{1/k}\ln^{1-1/k} n] &< Pr[X_{k-1} > 4n'(f+3)n^{1/k-1}\ln^{1-1/k} n] \\ &< (\frac{e^3}{4^4})^{\frac{n}{2}(f+3)n^{1/k-1}\ln^{1-1/k} n} = (\frac{e^3}{4^4})^{\frac{1}{2}(f+3)n^{1/k}\ln^{1-1/k} n} \\ &< \exp(-(f+3)n^{1/k}\ln^{1-1/k} n) < \exp(-(f+3)\ln n) \\ &= \frac{1}{n^{f+3}} < \frac{1}{2n^{f+2}}. \end{aligned}$$

Recall that $X_0, .., X_{k-2}$ are independent geometric random variables with parameter p, therefore for every s, $Pr[\sum_{i=0}^{k-2} X_i > s] = Pr[B(s,p) < k]$, where $B(s,p)$ is a binomial random variable with parameters s and p.

We get that,

$$Pr[\sum_{i=0}^{k-2} X_i > 16(f+3)\cdot n^{1/k}\ln^{1-1/k} n] = Pr[B(16(f+3)\cdot n^{1/k}\ln^{1-1/k} n, (f+3)^{1/(k-1)}(\frac{n}{\ln n})^{-1/k}) < k].$$

Note that $\mu = E[B(16(f+3)\cdot n^{1/k}\ln^{1-1/k} n, (f+3)^{1/(k-1)}(\frac{n}{\ln n})^{-1/k})] = 16(f+3)^{1+1/(k-1)}\ln n$. We assume that $k \leq \ln n$, therefore $k < \mu/2$. We now use the second Chernoff inequality, with $\delta = 1/2$,

$$Pr[\sum_{i=0}^{k-2} X_i > 16(f+3)\cdot n^{1/k}\ln^{1-1/k} n] < exp(-\mu/8) < \frac{1}{n^{2(f+3)}} < \frac{1}{2n^{f+2}}.$$

The probability that there is a node $v \in V$ and a subset $F \subset V$ of size at most f where the number of vertices $w \in V$ such that $v \in C^F(w)$ is greater than $20(f+3) \cdot n^{1/k} \ln^{1-1/k} n$ is at most $2n \cdot n^f \frac{1}{2n^{f+2}} = 1/n$. ∎

B Vertex fault tolerant spanners for weighted graphs

In this section we consider the construction of vertex fault tolerant spanners for graphs $G(V, E)$ with edge weights $\omega : E \to R^+$. We show that a slightly modified version of the algorithm for unweighted graphs presented in Section 2 yields a similar result for weighted graphs. Recall that in the case of unweighted graphs our crucial observation is that we only need to consider the clusters $C(w)$ defined in the algorithm of [32] up to depth k in order to get a $(2k-1)$-spanner (Theorem 2.2). In the case of weighted graphs this observation no longer holds. Indeed, on input edge (u, v) it could be the case that the algorithm of [32] returns an estimated path of length at most $2k-1$ times $\omega(u, v)$ but with more than $2k$ edges.

As in the unweighted case, we would like to guarantee for each edge (u, v) in E a corresponding node w such that (i) $u, v \in C(w)$ and (ii) the paths from u to w and from v to w in the spanner both contain at most k edges and are both of weight at most k times $\omega(u, v)$. In what follows we show how to modify the algorithm of [32] in order to get this property.

Let $\delta_i(w, v)$ be the length of the shortest path from v to w with at most i edges (if such a path does not exist then $\delta_i(w, v) = \infty$). Formally, δ_i no longer satisfies the triangle inequality, nevertheless it will suffice for our needs. The definition of $\delta_i(A_j, v)$ is changed accordingly, i.e., $\delta_i(A_j, v) \leftarrow min\{\delta_i(w, v) \mid w \in A_j\}$. The vertex $p_i(v)$ is now set to be the vertex w in A_i with the smallest $\delta_i(w, v)$, i.e., $\delta_i(p_i(v), v) = \delta_i(A_i, v)$. Namely, in the definition of $p_i(v)$, for small values of i we are considering paths with only few edges. Accordingly, we change the definition of $C(w)$ for $w \in A_i \setminus A_{i+1}$ to be $C(w) \leftarrow \{v \mid \delta_{i+1}(v, w) < \delta_{i+1}(v, p_{i+1}(v))\}$. So, for $w \in A_i \setminus A_{i+1}$ the clusters $C(w)$ will include only paths with at most $i+1$ edges. For a fault set F, we also use the analogous definitions for $C^F(w)$, $\delta_i^F(u, v)$ and $p_i^F(v)$ when considering the graph $G \setminus F$ (as done in Section 2). The corresponding modified algorithm **clusters**$(G(V, E), \{A_0, \ldots, A_k\}, k)$ is given in Figure 4.

We now show that the analysis given for the unweighed case in Section 2 can be modified slightly to hold in the weighted case as well. We first note that Proposition 2.5 holds for the definitions above. Namely, with high probability for every $v \in V$ and $F \subseteq V$ of size at most f the number of clusters that contain v is $O(fn^{1/k} \log^{1-1/k} n)$ in the graph $G \setminus F$. Only slight modifications are needed in the proof of Lemma 3.2 in [32] for our analysis to hold. Namely, in Lemma 3.2 of [32], instead of considering the nodes in A_i in nondecreasing order of distance from v, we now consider the nodes in A_i in nondecreasing order of the distance $\delta_{i+1}(w, v)$.

There are two additional differences in procedure **clusters**. The first difference is when we add a node v to a cluster $C(w)$ such that $w \in A_i \setminus A_{i+1}$, we add the entire shortest path from v to w with at most $i+1$ edges to the edge set E^{SP}. This is essential in order to get a spanner. In the unweighted case in Theorem 2.7 when considering the shortest path from a node v to w, all nodes in that path belongs to $C(w)$. In the new definitions this assumption not longer holds. Consider a shortest path from v to w and assume this shortest path is of length exactly $i+1$. Let z be the parent of v in that path. Assume G contains a path from z to $u \in A_{i+1}$ with exactly $i+1$ edges. Assume this path is very light and therefore z does not belongs to $C(w)$. As v is of distance $i+2$ from u, v belongs to $C(w)$. The second difference is that for each v and each i we add the path from v to $p_i(v)$ to the spanner edge set E^{SP}. This is also essential in order to get a spanner. In the unweighted case it holds that $v \in C(p_i(v))$. Again in the new distance definitions this is no longer holds. Consider for example the following case. Assume $p_1(v) \in A_1 \setminus A_2$, it could be that A_2 contains a node closer to v but with a path of two edges and not one, so $v \notin C(p_1(v))$. Notice that in the weighted case we distinguish between the clusters and the edge set E^{SP} as the the edge

set E^{SP} contains more edges, for each node v such that $v \in C(w)$ we might add k additional edges.

```
algorithm clusters(G(V,E), {A_0, ..., A_k}, k)

E^SP ← ∅
for every v ∈ V
    for i ← 0 to k − 1
        let δ_i(A_i, v) ← min{ δ_i(w, v) | w ∈ A_i}
        let p_i(v) ∈ A_i be such that δ_i(p_i(v), v) = δ_i(A_i, v)
        let P be the shortest path from v to p_i(v) with at most i edges.
        add to E^SP the edges of P.
    δ_k(A_k, v) ← ∞
for i ← 0 to k − 1
    for each w ∈ A_i \ A_{i+1}
        let C(w) ← {v | δ_{i+1}(v, w) < δ_{i+1}(v, p_{i+1}(v))}
        for each v ∈ C(w)
            let P be the shortest path from v to w with at most i + 1 edges.
            add to E^SP the edges of P.
let C ← ⋃_w C(w)

return H ← (V, E^SP)
```

Figure 4: The algorithm for weighted graphs

```
algorithm dist(u, v)

w_0 ← u; u_0 ← u; v_0 ← v  i ← 0
while v_i ∉ C(w_i)
    i ← i + 1
    (u_i, v_i) ← (v_{i−1}, u_{i−1})
    w_i ← p_i(u_i)
return δ_i(w, u) + δ_i(w, v)
```

Figure 5: Answering a distance query for an edge (u, v)

Next, we show that using the modified algorithm **clusters**$(G(V, E), \{A_0, \ldots, A_k\}, k)$ of Figure 4 in the framework discussed in Section 2 indeed yields a $2k-1$ spanner. In our analysis we use the query algorithm **dist**(u, v) from [32] (presented in Figure 5).

Lemma B.1 *For a given edge (u, v), there exists a vertex $w \in A_{i+1} \setminus A_i$ for some $0 \leq i \leq k-1$ such that one of the following occurs (i) $w = p_i(u)$, $v \in C(w)$, $\delta_i(w, u) \leq (k-1) \cdot \omega(u, v)$ and $\delta_{i+1}(w, v) \leq k \cdot \omega(u, v)$. (ii) $w = p_i(v)$, $u \in C(w)$, $\delta_i(w, v) \leq (k-1) \cdot \omega(u, v)$ and $\delta_{i+1}(w, u) \leq k \cdot \omega(u, v)$.*

Proof: We follow the proof of Lemma 3.3 in [32]. Denote by Δ the weight of the edge (u, v), i.e., $\Delta = \omega(u, v)$. It is shown in [32] that $\delta(w_i, u_i) \leq \delta(w_{i-1}, u_{i-1}) + \Delta$, if the ith iteration passes the test of the while-loop of Procedure **dist**. We need to show that $\delta_i(w_i, u_i) \leq \delta_{i-1}(w_{i-1}, u_{i-1}) + \Delta$, if the ith iteration passes the test of the while-loop of Procedure **dist**.

```
algorithm ft-spanner(G(V,E), k, f)

A ← initialize(V, k)
C^φ ← clusters(G(V,E), A, k)
C ← C^φ
for t = 1 to f
    for every F ⊆ V of size at most t
        E^F_SP ← clusters(G \ F, A, k)
        E_SP ← E_SP ⋃ E^F_SP
return E_SP
```

Figure 6: Our algorithm for constructing an f-vertex fault tolerant $(2k-1)$-spanner for weighted graph

Assume the ith iteration passes the test of the while-loop of Procedure **dist**, then $v_{i-1} \notin C(w_{i-1})$, so $\delta_i(w_{i-1}, v_{i-1}) \geq \delta_i(A_i, v_{i-1}) = \delta_i(p_i(v_{i-1}), v_{i-1})$. Moreover, $v_{i-1} = u_i$ and $w_i = p_i(u_i)$, so we get

$$\delta_i(w_i, u_i) = \delta_i(p_i(u_i), u_i) = \delta_i(p_i(v_{i-1}), v_{i-1}) \leq \delta_i(w_{i-1}, v_{i-1}) \leq \delta_{i-1}(w_{i-1}, u_{i-1}) + \Delta.$$

Where the last inequality follows from the fact that if there exists a path from w_{i-1} to v_{i-1} of length ℓ and with at most $i-1$ edges then there exists a path from w_{i-1} to u_{i-1} of length $\ell + \Delta$ and with at most i edges.

Assume the algorithm leaves the while-loop at iteration i. From the algorithm, we get that $w_i = p_i(u_i)$. From the analysis above we get that $\delta_i(w_i, u_i) < i\Delta$. As the algorithm does not pass iteration i of the while-loop, $v_i \in C(w_i)$. As $\delta_i(w_i, u_i) < i\Delta$, it must be that $\delta_{i+1}(w_i, v_i) < (i+1)\Delta$. ∎

W.l.o.g assume the first part of Lemma B.1 holds, i.e., $w = p_i(u)$, $v \in C(w)$, $\delta_i(w, u) \leq (k-1) \cdot \omega(u, v)$ and $\delta_{i+1}(w, v) \leq k \cdot \omega(u, v)$. As we add the shortest path from u to $w = p_i(u)$ with at most i edges to the spanner edge set E^{SP} and the shortest path from v to w with at most $i+1$ edges, we get that using the modified algorithm **clusters**$(G(V,E), \{A_0, \ldots, A_k\}, k)$ of Figure 4 in the framework discussed in Section 2 yields a $2k-1$ spanner.

Notice that with the new distance definition δ_i the depth of the produced clusters $C(w)$ is already up to depth k as we consider only paths with at most k edges. Therefore, we do not need to trim the clusters up to depth k as in the unweighted case.

The algorithm for constructing an f-vertex fault tolerant $(2k-1)$-spanner for weighted graph is given in Figure 6.

We now show an analogue property to Property 2.3.

Property B.2 *For any $F' \subset F$ and any vertex $v \in V$: if $v \in C^{F'}(w)$ and the path P connecting v to w added to the spanner edge set E^{SP} in the invocation of procedure* **clusters** *on $G \setminus F'$ does not include any vertices from the set $F \setminus F'$, then P is also added to the spanner edge set E^{SP} in the invocation of procedure* **clusters** *on $G \setminus F$.*

To ensure this property, when constructing $C^F(w)$ for some vertex $w \in V$ and a set F, we enforce the following rule. Let $V = \{v_1, .., v_n\}$. Assume the path from v to w is already constructed up to distance $r-1$ from v and we now add another node at distance r from v. Moreover, let x be the vertex at distance $r-1$. Let i be the smallest index such that there is an edge between v_i to x in $G \setminus F$ and v_i is of minimal distance from w. This implies that v_i is on a shortest path from v to w. We add v_i to the constructed

path from v to w. A similar analysis to Lemma 2.4 show that the modified **clusters** procedure using the rule described above satisfies Property B.2.

As Property 2.3 in the unweighted case, Property B.2 is essential here, in order to claim that triplets (v, w, F) such that $v \notin C^F_{\text{NEW}}(w)$ for nodes v, w and set F such that $|F| \leq f$ do not contribute additional edges to the spanner edge set.

We now turn to the analysis of the size.

Only slight modifications are needed in the proof of Theorem 2.7 for the weighted case.

Theorem B.3 *The Algorithm given in Figure 6 returns, with high probability, an f-fault tolerant spanner of $G(V, E)$ with stretch $2k-1$ and $O(f^3 k^{f+2} n^{1+1/k} \log^{1-1/k} n)$ edges.*

Proof: As mentioned above Proposition 2.5 also holds for the definitions above. In the proof of Theorem 2.7 we bound the number of tuples (v, w, F) for which $v \in C^F_{\text{NEW}}(w)$, $w \in A_i \setminus A_{i+1}$ and $|F| = f$ for given $v \in V$, given i such that $1 \leq i \leq k-1$ and given f. The analysis of the bound is the same for the weighted case. We just need to notice that in Theorem 2.7 when considering the shortest path from a node v to w, all nodes in that path belongs to $C^F(w)$ for some subsets $|F| \leq f$. In the weighted case this assumption not longer holds, instead we consider the path from v to w that was added to the spanner.

This gives us a bound on the number of tuples (v, w, F) for which $v \in C^F_{\text{NEW}}(w)$, $w \in A_i \setminus A_{i+1}$ and $|F| = f$ for given $v \in V$, given i such that $1 \leq i \leq k-1$ and given f.

As before, to obtain the final bound we will need to multiply by n (to count the cost of all the vertices), by k for all the sets $A_0 \setminus A_1, \ldots, A_{k-1} \setminus A_k$ and by f for all the possible sizes of sets. We get that the number of tuples (v, w, F) for which $v \in C^F_{\text{NEW}}(w)$ is bounded by $O(f^3 k^{f+1} n^{1+1/k} \log^{1-1/k} n)$.

We now turn to bound the number of edges added to the spanner edge set E^{SP}. For each such tuple (v, w, F) we may add k additional edges.

Moreover, in addition we also add edges for the paths from v to $p_i(v)$ for all $v \in V$ and $1 \leq i \leq k-1$. Using the same logic as in the analysis of Theorem 2.7 the number of such paths is bounded by nk^f, for each such path we add at most k edges.

Finally, with very high probability we get that the number of edges added to the spanner edge set is bounded by $O(f^3 k^{f+2} n^{1+1/k} \log^{1-1/k} n)$. ∎

C Analyzing the running time

The running time of our algorithm for vertex fault tolerant spanners, presented in Section 2, depends on n^f. This follows from the fact that in algorithm **ft-spanner**$(G(V, E), k, f)$ we are enumerating over all subsets $f \subseteq V$ of size at most f. We now show how to modify algorithm **ft-spanner**$(G(V, E), k, f)$ in order to get a running time of $f \cdot n^2$ times the number of edges in the spanner, namely, a running time of $O(f^4 k^{f+1} \cdot n^{3+1/k} \log^{1-1/k} n)$.

Let m be the number of edges in our spanners, $m = O(f^3 k^{f+1} n^{1+1/k} \log^{1-1/k} n)$. As $m << n^f$, for the vast majority of subsets F such that $|F| < f$ the spanner C^F computed in algorithm **ft-spanner**$(G(V, E), k, f)$ doesn't add any new edges to $\mathcal{C}$. Roughly speaking, we utilize this fact, and instead of invoking the **clusters** procedure for every subset F such that $|F| < f$, we only invoke it for subsets F that might contribute new edges to $\mathcal{C}$.

More precisely, we follows the analysis of Theorem 2.7 and add only the tuples (v, w, F) that satisfy $|F| \leq f$ and $v \in C^F_{\text{NEW}}(w)$.

We perform the following algorithm for every $v \in V$, every $0 \leq i \leq k-1$ and every $f' \leq f$.

Our goal now is to find all tuples (v, w, F) that satisfy $|F| = f'$, $w \in A_i \setminus A_{i+1}$ and $v \in C^F_{\text{NEW}}(w)$.

We handle the two cases mentioned in Theorem 2.7.

The first case is that $v \in C^{\phi}(w)$. The construction for this case follows the analysis of Claim 2.9 in Section 2.2. As in the Theorem 2.7 only subsets F that contains a node in the path P from v to $C^{\phi}(w)$ may satisfies $v \in C^F_{\text{NEW}}(w)$. For each w such that $w \in A_i \setminus A_{i+1}$ and $v \in C^{\phi}(w)$ we do the following. For each node u_1 in the path P, we consider the cluster $C^{\{u_1\}}(w)$. There are two possible scenarios, the first is that as before $v \in C^{\{u_1\}}(w)$ and the second is that $v \notin C^{\{u_1\}}(w)$. If v is in $C^{\{u_1\}}(w)$ then as before we consider all nodes u_2 in the path from v to w in $C^{\{u_1\}}(w)$ and store temporarily the sets $\{u_1, u_2\}$. The second scenario is that $v \notin C^{\{u_1\}}(w)$, here we consider all nodes u_2 in the path from v to $p^{\{u_1\}}_{i+1}(v)$ and store temporarily the sets $\{u_1, u_2\}$. We continue in a similar manner and for general j for all stored tuples F' of size $j-1$ we consider one of $2k$ vertices defined by v, w and F' and store the relevant sets F such that $|F| = j$. We continue with this manner until we reach subsets F of size f'. For each such subset F, we add the tuple (v, w, F) to our list.

The aim of the second case is to handle the vertices w for which $v \notin C^{\phi}(w)$. The construction for the second case follows the analysis that appears after Claim 2.9 in Section 2.2. We consider all nodes u_1 in the path from v to $p^{\phi}_{i+1}(v)$. For each such node u_1, we consider two subcases. First consider the vertices w such that $v \in C^{\{u_1\}}(w)$, for each such u_1 and w we follow the same algorithm as in the case for $v \in C^{\phi}(w)$ (the only difference is that our starting point is not ϕ but the set $\{u_1\}$) and we find all tuples (F, w, P) that satisfy $|F| = f'$, $u_1 \in F$ and $v \in C^F_{\text{NEW}}(w)$. The goal of the second subcase is to handle any other vertex w for which $v \notin C^{\{u_1\}}(w)$. As before, we consider the path that connects between v and $p^{\{u_1\}}_{i+1}(v)$ in $G \setminus \{u_1\}$. For each node u_2 in that path we consider the set $\{u_1, u_2\}$ and again store it temporarily. For a general iteration j we consider the case that $v \in C^{F'}(w)$ and the case that $v \notin C^{F'}(w)$ for all stored F' such that $|F'| = j$. For the first subcase, consider the vertices w such that $v \in C^{F'}(w)$, for each such node w we follow the same algorithm as in the case for $v \in C^{\phi}(w)$ where our starting point is F' and not ϕ and we find all tuples (v, w, F) that satisfy $|F| = f'$, $F' \subseteq F$ and $v \in C^F_{\text{NEW}}(w)$. We now handle any other vertex w for which $v \notin C^{F'}(w)$. We again consider all nodes u in the path between v and $p^{F'}_{i+1}(v)$ in $G \setminus F'$ and store all subsets $F' \cup \{u\}$. Again, we continue with this manner until we reach subsets F of size f'. For each such subset F, we find the vertices w such that $v \in C^F_{\text{NEW}}(w)$ and add the tuple (v, w, F) to our list.

As in the analysis of 2.7 the number of tuples (v, w, F) we consider is $O(f^3 k^{f+1} n^{1+1/k} \log^{1-1/k} n)$. For each such tuple (v, w, F) where $w \in A_i \setminus A_{i+1}$, we do some operations as finding $p^F_{i+1}(v)$ and constructing the cluster $C^F(w)$. This can be done in $O(|E|) = O(n^2)$ time. We consider each (v, w, F) at most f times, for each size of the subsets $f' \leq f$. We conclude that the running time is bounded by $O(f^4 k^{f+1} n^{3+1/k} \log^{1-1/k} n)$.