Diameter Spanner, Eccentricity Spanner, and Approximating Extremal Graph Distances: Static, Dynamic, and Fault Tolerant

Keerti Choudhary, Weizmann Institute of Science, Israel
keerti.choudhary@weizmann.ac.il

Omer Gold, Tel Aviv University, Israel
omergold@post.tau.ac.il

Abstract

The diameter, vertex eccentricities, and the radius of a graph are some of the most fundamental graph parameters, and dictate how fast information spreads in a network. In the past few years, a lot of work has been done in estimating these graph parameters in the static setting.

Roditty and Williams [STOC 2013] gave an $\tilde{O}(m\sqrt{n})$ time algorithm for computing a $1.5$ approximation of graph diameter. In this paper, we present the first non-trivial algorithm for maintaining $\frac{1}{2}$-approximation of graph diameter in dynamic environment. For any graph undergoing edge insertions (or edge deletions), our algorithm takes $O(\epsilon^{-1}n^{1.25})$ amortized update for maintaining a $(1.5 + \epsilon)$ approximation of graph diameter. For graphs whose diameter remains bounded by some large constant, the total update time of our algorithm is $\tilde{O}(\epsilon^{-2}m\sqrt{n})$, which almost matches the best known bound for static $1.5$-approximation of diameter.

Backurs et al. [STOC 2018] gave an $\tilde{O}(m\sqrt{n})$ time algorithm for computing $2$-approximation of eccentricities. They also showed that no $O(n^{2-o(1)})$ time algorithm can achieve an approximation factor better than $2$ for graph eccentricities, unless SETH fails. It was, however, not known if there exists a linear time solution to this problem. In this paper, we present the first $\tilde{O}(m)$ time algorithm for computing $2$-approximation of vertex eccentricities in general directed weighted graphs.

We also present fault tolerant data-structures for maintaining $1.5$-diameter and $2$-eccentricities.

Another important contribution of this paper is that we initiate the study of Extremal Distance Spanners. Since the diameter of a graph is known to have many practical applications in real-world networks, it is a natural question to ask how much can we sparsify a graph and still guarantee that its diameter remains bounded by some value $c$. This property is captured by the notion of extremal distance spanners. Given a graph $G = (V, E)$, a subgraph $H = (V, E_H)$ is defined to be a $t$-diameter-spanner if the diameter of $H$ is at most $t$ times the diameter of $G$.

We show that for any $n$-vertex and $m$-edges directed graph $G$ we can compute a sparse subgraph $H$ which is a $(1.5)$-diameter-spanner of $G$ and contains at most $\tilde{O}(n^{1.5})$ edges. We also show that the stretch factor cannot be improved to $(1.5 - \epsilon)$ by presenting for any pair $(n, D)$ of integers, an $n$-vertex unweighted graph of diameter $D$ whose $(1.5 - \epsilon)$-diameter spanner contains at least $\Omega(n^{1.5+\epsilon})$ edges, whenever $D \leq n^{1/4-\epsilon}$.

For graph whose diameter is bounded by some constant, we show existence of $\frac{5}{2}$-diameter spanner, containing at most $\tilde{O}(n^{2})$ edges. We also show that this bound is tight, by presenting for any integer $n$, an $n$-vertex graph whose $(\frac{5}{2} - \epsilon)$-diameter spanner contains at least $\Omega(n^{1.5})$ edges.

We present several other extremal-distance spanners with various size-stretch trade-offs, including $2$-eccentricity-spanners, and $2$-radius-spanners, that contains only $\tilde{O}(n)$ edges and are computable in $\tilde{O}(m)$ time. Finally, we extensively study these objects in the dynamic and fault-tolerant settings.

A graph in this paper always refer to directed graph.

$\tilde{O}(\cdot)$ hides the poly-logarithmic factors.
1 Introduction

The diameter, vertex eccentricities, and the radius of a graph are core graph parameters, and are useful in many real-world applications. In the context of data and communication networks they are specially important, as they determine how fast information can spread throughout the networks. However, the only known algorithm for computing exact diameter and graph eccentricities of a weighted directed graph is solving the all-pair-shortest-path problem and takes $\Omega(mn)$ time, where $n$ and $m$ respectively denote the number of vertices and edges in a graph. For unweighted directed graphs, the best known algorithm for computing diameter takes $O(\min\{mn, n^2\})$ time, where $\omega$ is constant of matrix multiplication [21]. So, in the past few years, a lot of work has been done in approximating the diameter, vertex eccentricities, and the radius of a graph in the static setting [3, 7, 15, 35, 41]. In this work, we present algorithms for maintaining these extremal distance graph parameters in the dynamic setting. We also improve the computation time of some of the existing extremal distance parameter in the static setting. Last, but not the least, we introduce a new notion called extremal distance spanners and study these objects in the static, dynamic, and fault-tolerant models. We now discuss our results in detail.

Dynamic maintenance of 1.5-approximate Diameter. Aingworth et al. [3] gave the first subcubic algorithm for obtaining a less than 2 approximation of graph diameter. They showed that for any $n$-vertex directed graph, we can compute an almost 1.5-approximation of diameter that in $\tilde{O}(n^2 + mn)$ time. Roditty and Williams [41] improved this result by giving a randomized algorithm that takes $O(m\sqrt{n})$ expected time for computing a 1.5-approximation of graph diameter. They also showed that any $(1.5 - \epsilon)$-approximation algorithm for graph diameter with $\tilde{O}(n^{2-\delta})$ runtime, for any constants $\epsilon, \delta > 0$ would falsify the SETH conjecture [37, 38]. In this paper, we present the first non-trivial algorithm for maintaining a less than 2 approximation of diameter in dynamic graph undergoing edge insertions (or deletions).

**Theorem 1.1.** For any $\epsilon \in [0, 1/2]$, there exists an incremental (and decremental) algorithm that maintains for an $n$-vertex directed graph $G$ a $(1.5 + \epsilon)$-approximation of graph diameter whose expected amortized update time is $O(\epsilon^{-2} D_0 \sqrt{n} \log^{4.5} n)$ for incremental setting and $O(\epsilon^{-1} D_0 \sqrt{n} \log^{1.5} n)$ for decremental setting, where $D_0$ denotes an upper bound on the diameter of the graph throughout the run of the algorithm.

Observe that for graphs whose diameter remains bounded by some large constant $D_0$, the total update time of our incremental and decremental algorithms is $\tilde{O}(\epsilon^{-2} m\sqrt{n})$, which almost matches the best known static bound of [41] for obtaining a $(2 - \epsilon)$-approximation of diameter for directed graphs.

For graphs with large diameter we obtain another result for approximating diameter, which is still better than running the static approximate-diameter algorithm from scratch even for graphs with $\tilde{O}(n)$ edges.

**Theorem 1.2.** For any $\epsilon \in [0, 1/2]$, there exists a Monte-Carlo algorithm for incrementally/decrementally maintaining for an $n$-vertex directed graph $G$ a $(1.5 + \epsilon)$ approximation of diameter. The algorithm outputs a correct approximation with high probability, and its amortized update time is $O(\epsilon^{-1} n^{1.25} \log^{2.75} N)$ for incremental setting, and $O(\epsilon^{-0.5} n^{1.25} \log^{1.25} N)$ for decremental setting.

Near optimal time algorithm for 2-approximation of eccentricities. For the problem of computing exact eccentricities in directed graphs the only known solution is solving the all-pair-shortest-path problem and takes $\Omega(mn)$ time. Backurs et al. [7] showed that for any directed weighted graph there exists an algorithm for computing 2-approximation of eccentricities in $\tilde{O}(m\sqrt{n})$ time. They also showed that for any $\delta > 0$, there exists an $O(m/\delta)$ time algorithm for computing a $2 + \delta$ approximation of graph eccentricities. We improve these results by presenting an $\tilde{O}(m)$ time algorithm for 2-approximation of eccentricities.

**Theorem 1.3.** For any directed weighted graph $G = (V, E)$ with $n$ vertices and $m$ edges, we can compute in $O(m \log^2 n)$ expected time a $2$-approximation of eccentricities of vertices in $G$.
Our result is essentially **tight**. The approximation factor of 2 cannot be improved since Backurs et al. showed in their paper that unless SETH fails no $O(n^{2-o(1)})$ time algorithm can achieve an approximation factor better than 2 for graph eccentricities [7]. Also the computation time of our algorithm is almost optimal as we need $\Omega(m)$ time to even scan the edges of the graph.

**Extremal Distance Spanners** Another important contribution of this paper is that we initiate the study of *Extremal Distance Spanner*. A spanner (also known as distance spanner) of a graph $G = (V,E)$ is a sparse subgraph $H = (V,E_H)$ that approximately preserves pair-wise distances of the underlying graph $G$. Distance spanners are known to have numerous applications in different areas of computer science such as distributed systems, communication networks and efficient routing schemes [19, 20, 39, 40, 42, 32, 5, 31], motion planning [24, 18], approximating shortest paths algorithms [16, 17, 25] and distance oracles [11, 43]. The works of [4, 10] provided efficient constructions of spanners for undirected graphs with $O$ edges and multiplicative stretch $2k - 1$, for any integer $k \geq 1$. It is also widely believed that this size-stretch trade-off is tight.

Unfortunately, the landscape of distance spanners in the directed setting is far less understood, and no universal construction for spanners in directed graphs are known, even when underlying graph is strongly-connected. This brings us to the following question.

**Question 1.1.** Given a graph $G = (V,E)$ and a “stretch factor” $t$, can we construct a sparse subgraph $H$ of $G$ such that the distance between any two vertices in $H$ is bounded by $t$ times the maximum distance in $G$?

We define such graphs as $t$-diameter-spanners as they essentially preserve the diameter up to a multiplicative factor $t$. Such a sparsification of graphs indeed suffices for many of the original applications of spanners, such as communication networks, facility location problem, routing, etc. The motivation of studying diameter spanners stems from the fact that the diameter indicates how quickly information can spread in a network.

We show sparse diameter spanner constructions with various trade-offs between the size (number of edges) of the spanner and its stretch factor $t$, and provide efficient algorithms to construct such spanners.

The following theorem provides efficient construction of 1.5 diameter spanners.

**Theorem 1.4.** For any directed unweighted graph $G$ with $n$ vertices and $m$ edges, we can compute a 1.5-diameter-spanner $H$ of $G$ with at most $O(n^{3/2}\sqrt{\log n})$ edges. The computation time of $H$ is $\tilde{O}(m^{1/2})$ time with high probability. Moreover, if $G$ is edge-weighted, then $H$ satisfies the condition that $\text{diam}(H) \leq 1.5 \text{diam}(G) + W$, where $W$ is an upper bound on the weight of edges in $G$.

The following theorem shows that our 3/2-stretch diameter-spanner construction is essentially tight for graphs whose diameter is bounded by $O(n^{1/4})$.

**Theorem 1.5.** For every $n$ and every $D$, there exists an $n$-vertex unweighted directed graph $G = (V,E)$ with diameter $\Theta(D)$ such that any subgraph $H = (V,E' \subseteq E)$ of $G$ with $\text{diam}(H) \leq 1.5\text{diam}(G) - 2$, contains $\Omega(n^{2/D^2})$ edges.

The next theorem shows construction of 5/3-diameter-spanners that are sparser than the 1.5-diameter-spanners whenever $D = o(\sqrt{n})$.

**Theorem 1.6.** For any directed graph $G = (V,E)$ with diameter $D$, in $\tilde{O}(mn^{1/3}(D + n/D)^{1/3})$ expected time we can compute a subgraph $H = (V,E' \subseteq E)$ satisfying $\text{diam}(H) \leq [5D/3]$ that contains at most $O(n^{4/3}(\log n)^{2/3}D^{1/3})$ edges, where $n$ and $m$ respectively denotes the number of vertices and edges in $G$.

---

1Though the time required for computing $H$ is a function of $D$, the algorithm does not need to apriori know the value $D$. 

---
We also show that our $5/3$-stretch spanner construction is tight, by showing existence of graphs whose $(5/3 - \epsilon)$-diameter-spanner requires at least $\Omega(n^{1.5})$ edges.

**Theorem 1.7.** For every $n$, there exists an $n$-vertex graph $G = (V, E)$ with $O(n^{3/2})$ edges, for which any subgraph $H = (V, E' \subseteq E)$ of $G$ with $\text{diam}(H) \leq (5/3 - \epsilon)\text{diam}(G)$ must have $\Omega(n^{3/2})$ edges.

For graphs with diameter at least $\omega(n^{5/6})$, we show computation of diameter spanners with $o(1)$ stretch that have size almost linear in $n$.

**Theorem 1.8.** For any $n$-vertex directed graph $G = (V, E)$ satisfying $\text{diam}(G) = \omega(n^{5/6})$, we can compute a subgraph $H = (V, E')$ with $O(n \log^2 n)$ edges satisfying $\text{diam}(H) \leq (1 + o(1))\text{diam}(G)$.

Given a graph $G$, we say that a subgraph $H$ of $G$ is $t$-eccentricity-spanner of $G$ if the eccentricity of any vertex $x$ in $H$ is at most $t$ times its eccentricity in $G$. Similarly, $H$ is said to be a $t$-radius spanner, if the radius of $H$ is at most $t$ times the radius of graph $G$. We obtain following constructions for eccentricity-spanners for stretch 2 and 3.

**Theorem 1.9.** There exists an algorithm that for any directed weighted graph $G = (V, E)$ with $n$ vertices and $m$ edges, computes in $O(m \log^2 n)$ expected time

(i) 3-eccentricity-spanner (and a 3-radius-spanner) of $G$ with at most $2n$ edges.

(ii) 2-eccentricity-spanner (and a 2-radius-spanner) of $G$ with at most $O(n \log^2 n)$ edges.

---

| Time complexity                        | Stretch | Size                                    | Diameter                        |
|----------------------------------------|---------|-----------------------------------------|---------------------------------|
| **Static Diameter Spanner**            |         |                                         |                                 |
| $O(m \sqrt{n \log n})$                 | 1.5     | $O(n^{1.5} \sqrt{\log n})$             |                                |
| $O(m^{2/3})$                           | 5/3     | $O(n^{4/3} D^{1/3} \log^{2/3} n)$ diam = $D$ |                                |
| $O(n^{c})$, for some constant $c$      | $n^{5/6}$-additive | $O(n \log^2 n)$                          |                                |
| **Incremental Diameter Spanner**       |         |                                         |                                 |
| $O(\epsilon^{-1} n^{1.25} \log^{2.25} n)$ amortized | $1.5 + \epsilon$ | $O(n^{1.5} \sqrt{\log n})$             | $\leq D_{\text{max}}$ |
| $O(\epsilon^{-2} \sqrt{n D_{\text{max}}} \log^{4.25} n)$ amortized | $1.5 + \epsilon$ | $O(n^{1.5} \sqrt{\log n})$             | $\leq D_{\text{max}}$ |
| $O(\epsilon^{-2} n^{2/3} D_{\text{max}} \log^{5.5} n)$ amortized | $5/3 + \epsilon$ | $O(n^{4/3} D_{\text{max}}^{1/3} \log n)$ | $\leq D_{\text{max}}$ |
| **Decremental Diameter Spanner**       |         |                                         |                                 |
| $O(\epsilon^{-0.5} n^{1.25} \log^{1.25} n)$ amortized | $1.5 + \epsilon$ | $O(n^{1.5} \sqrt{\log n})$             | $\leq D_{\text{max}}$ |
| $O(\epsilon^{-1} \sqrt{n D_{\text{max}}} \log^{1.5} n)$ amortized | $1.5 + \epsilon$ | $O(n^{1.5} \sqrt{\log n})$             | $\leq D_{\text{max}}$ |
| $O(\epsilon^{-2} n^{2/3} D_{\text{max}} \log^{2.5} n)$ amortized | $5/3 + \epsilon$ | $O(n^{4/3} D_{\text{max}}^{1/3} \log n)$ | $\leq D_{\text{max}}$ |
| **Eccentricity-Spanner and Radius-Spanner** |         |                                         |                                 |
| $O(m \log^2 n)$                        | 3       | $2n$                                    |                                |
| $O(m \log^2 n)$                        | 2       | $O(n \log^2 n)$                         |                                |
| **Subset Eccentricity-Spanner and Radius-Spanner** |         |                                         |                                 |
| $O(m \sqrt{|W| \log n})$              | 2       | $O(n \sqrt{|W| \log n})$               |                                |
| **Incremental Eccentricity-Spanner and Radius-Spanner** |         |                                         |                                 |
| $O(\epsilon^{-3} \sqrt{n D_{\text{max}}} \log^{5.5} n)$ amortized | $2 + \epsilon$ | $O(n^{1.5} \sqrt{\log n})$             | $\leq D_{\text{max}}$ |
| **Decremental Eccentricity-Spanner and Radius-Spanner** |         |                                         |                                 |
| $O(\epsilon^{-1} \sqrt{n D_{\text{max}}} \log^{1.5} n)$ amortized | $2 + \epsilon$ | $O(n^{1.5} \sqrt{\log n})$             | $\leq D_{\text{max}}$ |
The Table 1 presents a partial summary of our dynamic algorithms for maintaining diameter-spanners and eccentricity-spanners. For our results on fault-tolerant data-structures, see Section 7.

1.1 Other Related work

The girth conjecture of Erdős \cite{29} implies that there are undirected graphs on \( n \) vertices, for which any \((2k - 1)\)-spanner will require \( \Omega(n^{1+1/k}) \) edges. This conjecture has been proved for \( k = 1, 2, 3, \) and \( 5 \), and is widely believed to be true for any integer \( k \). Thus, assuming the girth conjecture, one can not expected for a better size-stretch trade-offs.

Althöfer et al. \cite{4} were the first to show that any undirected weighted graph with \( n \) vertices has a \((2k - 1)\)-spanner of size \( O(n^{1+1/k}) \). The lower bound mentioned above implies that the \( O(n^{1+1/k}) \) size-bound of this spanner is essentially optimal. Althöfer et al. \cite{4} gave an algorithm to compute such a spanner, and subsequently, a long line of works have studied the question of how fast can we compute such a spanner, until Baswana and Sen \cite{10} gave a linear-time algorithm.

A \( c \)-additive spanner of an undirected graph \( G \) is a subgraph \( H \) that preserves distances up to an additive constant \( c \). That is, for any pair of nodes \( u, v \) in \( G \) it holds that \( d_H(v, u) < d_G(v, u) + c \). This type of spanners were also extensively studied \cite{9, 14, 2, 28}. The sparsest additive spanner known is a \( 6 \)-additive spanner of size \( O(n^{4/3}) \) that was given by Baswana, Kavitha, Mehlhorn, and Pettie \cite{9}. It was only recently that Abboud and Bodwin \cite{11} proved that the \( O(n^{4/3}) \) additive spanner bound is tight, for any additive constant \( c \).

Since for directed graph distance spanners are impossible, the roundtrip distance metric was proposed. The roundtrip-distance between two vertices \( u \) and \( v \) is the distance from \( v \) to \( u \) plus the distance from \( u \) to \( v \). Roditty, Thorup, and Zwick \cite{40} presented the notion of roundtrip spanners for directed graphs. A roundtrip spanner of a directed graph \( G \) is a sparse subgraph \( H \) that approximately preserve the roundtrip distance between each pair of nodes \( v \) and \( u \). They showed that any directed graph has roundtrip spanners, and gave efficient algorithms to construct such spanners.

The question of finding the sparsest spanner of a given graph was shown to be NP-Hard by Peleg and Schäffer \cite{22}, in the same work that graph spanner notion was introduced by Peleg and Schäffer \cite{22}.

Diameter spanners were mentioned by Elkin and Peleg \cite{27, 26}, but in the context of approximation algorithms for finding the sparsest diameter spanner (which is NP-Hard). To the best of our knowledge, there is no work that showed the existence of sparse diameter spanners with stretch less than 2, for directed graphs.

2 Preliminaries

Given a directed graph \( G = (V, E) \) on \( n = |V| \) vertices and \( m = |E| \) edges, the following notations will be used throughout the paper.

- \( \pi_G(v, u) \): the shortest path from vertex \( v \) to vertex \( u \) in graph \( G \).
- \( d_G(v, u) \): the length of the shortest path from vertex \( v \) to vertex \( u \) in graph \( G \). We sometimes denote it by \( d(v, u) \), when the context is clear.
- \( \text{diam}(G) \): the diameter of graph \( G \), that is, \( \max_{u,v \in V} d_G(u, v) \).
- \( \text{OUT-BFS}(s) \): an outgoing breadth-first-search (BFS) tree of \( s \).
- \( \text{IN-BFS}(s) \): an incoming breadth-first-search (BFS) tree of \( s \).
- \( \text{InEcc}(s) \): the depth of tree \( \text{IN-BFS}(s) \).
- \( \text{OutEcc}(s) \): the depth of tree \( \text{OUT-BFS}(s) \).
- \( \text{rad}(G) \): the radius of graph \( G \), that is, \( \min_{v \in V} \text{OutEcc}(v) \).
- \( \text{OutEcc}(x, W) \): \( \max_{w \in W} d(x, w) \).
• rad(G|W): \( \min_{w \in W} \text{OutEcc}(w, W) \).
• OUT-BFS\((s, d)\): the tree obtained from OUT-BFS\((s)\) by truncating it at depth \(d\).
• IN-BFS\((s, d)\): the tree obtained from IN-BFS\((s)\) by truncating it at depth \(d\).
• \(N^{\text{out}}(x, \ell)\): the \(\ell\) closest outgoing vertices of \(x\), where ties are broken arbitrarily.
• \(N^{\text{in}}(x, \ell)\): the \(\ell\) closest incoming vertices of \(x\), where ties are broken arbitrarily.
• Depth\((v, T)\): the depth of vertex \(v\) in the rooted tree \(T\).
• Depth\((T, W)\): \(\max_{w \in W} \text{Depth}(w, T)\).
• \(P(W)\): the power set of \(W\).

Throughout the paper we assume the graph \(G\) is strongly connected, as otherwise the diameter of \(G\) is \(\infty\), and even an empty subgraph of \(G\) preserves its diameter.

We first formally define the notion of the \(t\)-diameter spanners that is used in the paper.

**Definition 2.1.** Given a directed graph \(G = (V, E)\), a subgraph \(H = (V, E' \subseteq E)\) is said to be \(t\)-diameter spanner of \(G\) if \(\text{diam}(H) \leq \lceil t \cdot \text{diam}(G) \rceil\).

We below state few results that will useful in our construction.

**Lemma 2.1.** Let \(G = (V, E)\) be an \(n\)-vertex directed graph. Let \(n_p, n_q \geq 1\) be integers satisfying \(n_p n_q = 8n \log n\), and let \(S\) be a uniformly random subset of \(V\) of size \(n_p\). Then with a high probability, \(S\) has non-empty intersection with \(N^{\text{in}}(v, n_q)\) and \(N^{\text{out}}(v, n_q)\), for each \(v \in V\).

In order to dynamically maintain diameter-spanners, we will use the following result by Even and Shiloach [30] on maintaining single-source shortest-path-trees. Even and Shiloach gave the algorithm for maintaining shortest path tree in the decremental setting, and their algorithm can be easily adapted to work in the incremental setting as well.

**Theorem 2.1 (ES-tree [30]).** There is a decremental (incremental) algorithm for maintaining the first \(k\) levels of a single-source shortest-path-tree, in a directed or undirected graph, whose total running time, over all deletions (insertions), is \(O(km)\), where \(m\) is the initial (final) number of edges in the graph.

### 3 Static and Dynamic maintenance of Diameter Spanners with 1.5 stretch

Our main idea for computing a sparse static 1.5-diameter-spanner is very simple and comes from the recent line of works [41, 7, 15, 36] on approximating diameter in directed or undirected graphs. Let \(S_1\) be a uniformly random subset of \(V\) of size \(\sqrt{n \log n}\). We take \(a\) to be the vertex of the maximum depth in OUT-BFS\((S_1)\). Also, \(S_2\) is set to \(N^{\text{in}}(a, \sqrt{n \log n})\). By Lemma 2.1 with high probability, the set \(N^{\text{in}}(a, \sqrt{n \log n})\) contains a vertex of \(S_1\), if not, we can re-sample \(S_1\), and compute \(a, S_2\) again. For convenience, throughout this paper, we refer to this constructed set-pair as a valid set-pair. The key idea in all the aforementioned papers for estimating graph diameter was that the set \(S_1 \cup S_2\) contain at least one vertex of high eccentricity. Interestingly, we show that the subgraph \(H\) which is just union of the shortest-path-trees rooted at vertices lying in \(S_1 \cup S_2\) is also a 1.5-diameter-spanner of \(G\). The proof of this follows from a nice-structural property of valid-set-pairs which we next discuss.

Let us first introduce the notion of \((h_1, h_2)\)-dominating-set-pair which are a generalization of traditional \(h\)-dominating sets [33, 34].

---

2If the graph \(G\) is undergoing either edge insertions, or edge deletions, then with high probability the relation holds for each of the \(O(n^2)\) instances of \(G\).
In this subsection, we provide efficient algorithms for maintaining a dominating-set-pair.

### 3.1 Dynamic Algorithms for maintaining dominating-sets

For any integers \( n \), to be a uniformly random set of size \( n \), for a chosen \( \langle \lfloor \frac{n}{2} \rfloor \), and either (1) for each \( x \in V \), \( d_G(x, s_1) \leq h_1 \), or (2) for each \( x \in V \), \( d_G(x, s_2) \leq h_2 \).

Here, \( S_1 \) is said to be \( h_1 \)-out-dominating if it satisfies condition 1, and \( S_2 \) is said to be \( h_2 \)-in-dominating if it satisfies condition 2.

We now show that a valid-set-pair is \( \langle \lfloor \frac{n}{2} \rfloor \), and either (1) for each \( x \in V \), \( d_G(x, s_1) \leq h_1 \), or (2) for each \( x \in V \), \( d_G(x, s_2) \leq h_2 \).

### 3.1 Dynamic Algorithms for maintaining dominating-sets

Let \( H \) denote the subgraph of \( G \) which is union of \( \text{IN-BFS}(s) \) and \( \text{OUT-BFS}(s) \), for \( s \in S_1 \cup S_2 \). Observe that computation of \( H \) takes \( O(\sqrt{n} \log n) \) expected time (recall computation of \( (S_1, S_2) \) is randomized).

If \( S_1 \) is \( \lfloor \frac{n}{2} \rfloor \)-out-dominating, then \( H \) is a 1.5-diameter-spanner. Indeed, for any \( x, y \in V \), there is an \( s \in S_1 \) satisfying \( d_H(s, y) = d_G(s, y) \leq \lfloor \frac{n}{2} \rfloor \cdot \sqrt{n} \log n \).

Moreover, if \( G \) is edge-weighted, then \( H \) satisfies the condition that \( \text{diam}(H) \leq 1.5 \text{ diam}(G) + W \), where \( W \) is an upper bound on the maximum edge-weight in \( G \).

In order to maintain a 1.5-diameter-spanner dynamically, a naive approach would be to dynamically maintain the valid-sets. However, we face two challenges, the first being dynamic maintenance of a vertex \( a \) having maximum depth in tree \( \text{OUT-BFS}(S_1) \), and the second is dynamically maintaining the set \( N^{in}(a, n_q) \). So, we resort to dynamic maintenance of dominating-set-pairs, which are \( \langle \lfloor (p + \epsilon)D \rfloor, \lfloor (q + \epsilon)D \rfloor \rangle \)-dominating for a chosen \( \epsilon > 0 \).

### 3.1 Dynamic Algorithms for maintaining dominating-sets

In this subsection, we provide efficient algorithms for maintaining a dominating-set-pair.

Observe that the static construction of \( \langle \lfloor D \rfloor, \lfloor qD \rfloor \rangle \)-dominating-set-pair holds even when \( S_1 \) is chosen to be a uniformly random set of size \( n_p \), and \( S_2 \) is \( N^{in}(a, n_q) \), where \( n_p \) and \( n_q \) are integers satisfying \( n_p n_q = 8n \log n \). The construction of dominating-set-pair can be even further generalized, and will be useful in constructing other diameter-spanners later in the paper (for proof of lemma refer to Appendix).

**Lemma 3.1.** For any integers \( n_p, n_q \geq 1 \) satisfying \( n_p \cdot n_q = 8n \log n \), and any directed graph \( G = (V, E) \) with \( n \) vertices and \( m \) edges, in \( O(m^{1/2}) \) expected time we can compute a set-pair \( (S_1, S_2) \in P(V) \times P(V) \) of size bound \( \langle n_p, n_q \rangle \) which is \( \langle \lfloor p \text{ inEcc}(a) \rfloor, \lfloor q \text{ inEcc}(a) \rfloor \rangle \)-dominating for some vertex \( a \in V \) and any arbitrary fractions \( p, q \) satisfying \( p + q = 1 \).

Our main approach for dynamically maintaining a dominating-set-pair is to use the idea of lazy updates. We formalize this through the following lemma.

**Lemma 3.2.** Let \( G \) be a dynamic graph whose updates are insertions (or deletions) of edges, and \( S_1 \) be a (non-dynamic) subset of \( V \) of size \( n_p \). Let \( t \geq t_0 \) be two time instances, and let \( S_2 = N^{in}_{t_0}(a, n_q) \), for some \( a \in V \). Let \( t_0 = \text{DEPTH}_{t_0}(\text{OUT-BFS}(S_1)) \) and \( \epsilon \in [0, 1/2] \) be such that \( \text{DEPTH}_{t}(\text{OUT-BFS}(S_1)) \) and
DEPTH$_t(a, \text{OUT-BFS}(S_1))$ lies in the range $[\ell_0(1-\epsilon), \ell_0(1+\epsilon)]$. Then for any fractions $p, q > 0$ satisfying $p + q = 1$, set-pair $(S_1, S_2)$ is $\langle([p+2\epsilon) \text{InEcc}_t(a)], ([q+2\epsilon) \text{InEcc}_t(a)]\rangle$ dominating at time $t$ if $S_1 \cap S_2$ is non-empty, and

(i) DEPTH$_{t_0}(a, \text{OUT-BFS}(S_1)) \geq (1 - \epsilon)\ell_0$ when restricted to edge deletions case.

(ii) DEPTH$_t(\text{IN-BFS}(a)) \geq (1 - \epsilon)\text{DEPTH}_{t_0}(\text{IN-BFS}(a))$ when restricted to edge insertions case.

Proof. Let $\delta$ and $\delta_0$ respectively denote the values $\text{InEcc}_t(a)$, and $\text{InEcc}_{t_0}(a)$.

We first analyse the edge deletions case. If depth of $\text{OUT-BFS}(S_1)$ at the time $t$ is bounded by $[(p + \epsilon)\delta]$, then $S_1$ is $[(p + \epsilon)\delta]$-out-dominating. So let us assume that DEPTH$_t(\text{OUT-BFS}(S_1))$ is strictly greater than $(p + \epsilon)\delta$. Then $(p + \epsilon)\delta < \text{DEPTH} t(\text{OUT-BFS}(S_1)) \leq (1 + \epsilon)\ell_0$. So $\ell_0 > \left(\frac{p + \epsilon}{1 + \epsilon}\right)\delta > p\delta$. Note that DEPTH$_{t_0}(a, \text{OUT-BFS}(S_1))$ is bounded below by $(1 - \epsilon)\ell_0 > (1 - \epsilon)p\delta > (p - \epsilon)\delta$, so, at time $t_0$, the truncated tree IN-BFS$_{t_0}(a, (p - \epsilon)\delta)$ must have empty intersection with $S_1$. Since $S_2 = N_{t_0}^{\in}(a, n_q)$ intersects $S_1$, at time $t_0$, IN-BFS$_{t_0}(a, (p - \epsilon)\delta)$ must be contained in $N_{t_0}^{\in}(a, n_q) = S_2$. The crucial point to observe is that in the decremental scenario, the set IN-BFS$_t(a, (p - \epsilon)\delta)$ can only reduce in size with time. Thus IN-BFS$_t(a, (p - \epsilon)\delta) \subseteq \text{IN-BFS}_{t_0}(a, (p - \epsilon)\delta) \subseteq S_2$. Since $S_2$ contains IN-BFS$_t(a, (p - \epsilon)\delta)$, we have DEPTH$_t(\text{IN-BFS}(S_2)) \leq \text{DEPTH} t(\text{IN-BFS}(a)) - (p - \epsilon)\delta = \delta - (p - \epsilon)\delta = [(q + \epsilon)\delta]$. Thus, if $S_1$ is not $(p + \epsilon)\delta$-out-dominating, then $S_2$ is $[(q + \epsilon)\delta]$-in-dominating set.

We next analyse the edge insertions case. If DEPTH$_{t_0}(a, \text{OUT-BFS}(S_1)) \leq p\delta$, then $S_1$ is $(p + 2\epsilon)\delta$-out-dominating at time $t_0$ because DEPTH$_{t_0}(\text{OUT-BFS}(S_1)) = \ell_0 \leq \text{DEPTH} t(a, \text{OUT-BFS}(S_1))/(1 - \epsilon) \leq \text{DEPTH} t_0(a, \text{OUT-BFS}(S_1))/(1 - \epsilon) \leq (1 - \epsilon)p\delta \leq (p + 2\epsilon)\delta$. If DEPTH$_{t_0}(a, \text{OUT-BFS}(S_1)) > p\delta$, then at time $t_0$ the truncated tree IN-BFS$_{t_0}(a, p\delta)$ must have an empty intersection with set $S_1$, however, the set $S_2 = N_{t_0}^{\in}(a, n_q)$ has a non-empty intersection with $S_1$, thus IN-BFS$_{t_0}(a, p\delta) \subseteq N_{t_0}^{\in}(a, n_q) = S_2$. So DEPTH$_{t_0}(\text{IN-BFS}(S_2)) \leq \text{DEPTH} t_0(\text{IN-BFS}(a)) - p\delta \leq \delta/(1 - \epsilon) - p\delta \leq [(q + 2\epsilon)\delta]$. Thus, $(S_1, S_2)$ is $\langle([p + 2\epsilon)\delta], [(q + 2\epsilon)\delta]\rangle$ dominating at time $t_0$. Observe that as edges are added to $G$, the depth of vertices in $\text{OUT-BFS}(S_1)$ and $\text{IN-BFS}(S_2)$ can only decrease with time, so the set-pair $(S_1, S_2)$ must also be $\langle([p + 2\epsilon)\delta], [(q + 2\epsilon)\delta]\rangle$ dominating at time $t$.

We now present algorithms that for a given $\epsilon \in [0, 1/2]$, and integers $n_p, n_q \geq 1$ satisfying $n_p n_q = 8n \log n$, incrementally (and decrementally) maintains for an $n$-vertex graph $G$ a triplet $(S_1, S_2, a) \in P(V) \times P(V) \times V$ such that at any time instance $t$,

(i) $|S_1| = n_p$, and $S_2 = N_{t_0}^{\in}(a, n_q)$ for some $t_0 \leq t$,

(ii) DEPTH$_t(\text{OUT-BFS}(S_1)), \text{DEPTH} t(a, \text{OUT-BFS}(S_1)), \text{DEPTH}_0(a, \text{OUT-BFS}(S_1)) \in [\ell_0(1-\epsilon), \ell_0(1+\epsilon)]$, where $\ell_0 = \text{DEPTH}_{t_0}(\text{OUT-BFS}(S_1))$,

(iii) \text{DEPTH} t(\text{IN-BFS}(a)) \geq (1 - \epsilon)\text{DEPTH}_{t_0}(\text{IN-BFS}(a))$.

**Incremental scenario.** We first discuss the incremental scenario. The main obstacle in this setting is to dynamically maintain a vertex $a$ having large depth in $\text{OUT-BFS}(S_1)$. We initialize $S_1$ to a uniformly random subset of $V$ containing $n_p$ vertices, and store in $\ell_0$ the depth of tree $\text{OUT-BFS}(S_1)$. Next we compute a set FAR that consist of all those vertices whose distance from $S_1$ is at least $(1 - \epsilon)\ell_0$, and set $A$ to be a uniformly random subset of FAR of size $\min[8 \log n, |\text{FAR}|]$. We initialize $a$ to any arbitrary vertex in set $A$, and set $S_2$ to $N_{t_0}^{\in}(a, n_q)$. Throughout the algorithm whenever $S_1 \cap S_2$ is non-empty, then we recompute $S_1, \ell_0, A, a,$ and $S_2$. The probability of such an event is inverse polynomial in $n$.

We use Theorem 2.1 to dynamically maintain DEPTH(OUT-BFS(S_1)) and depth of individual vertices in OUT-BFS(S_1). This takes $O(mD)$ time for any fixed $S_1$. Whenever DEPTH(OUT-BFS(S_1)) falls below the value $(1 - \epsilon)\ell_0$, then we recompute $\ell_0, A, a$ and $S_2$. For any fixed $S_1$, this happens at most $O(\epsilon^{-1} \log n)$
times, and so takes $O(me^{-1} \log n)$ time in total. Whenever depth of a vertex lying in $A$ falls below the value $(1 - \epsilon)\ell_0$, then we remove that vertex from $A$. This step takes $O(m|A|) = O(m \log n)$ time in total. If $\text{DEPTH}(a, \text{OUT-BFS}(S_1))$ falls below the value $(1 - \epsilon)\ell_0$, then we replace $a$ by an arbitrary vertex in $A$, and recompute $S_2$.

If $A$ becomes empty and $\text{DEPTH}(\text{OUT-BFS}(S_1))$ is still greater than $(1 - \epsilon)\ell_0$, then we recompute $\text{FAR}, \text{A}, a$, and $S_2$. Observe that for any fixed $\ell_0$ this happens at most $\log n$ times. This is because if $\text{FAR}_1$ and $\text{FAR}_2$ is a partition of $\text{FAR}$ such that the depth of all vertices in $\text{FAR}_1$ falls below $(1 - \epsilon)\ell_0$ earlier than the vertices in $\text{FAR}_2$, then with high probability $A$ has a non-empty intersection with $\text{FAR}_2$. This holds true as we assume adversarial model in which edge insertions are independent of choice of $A$. Thus with high probability, each time $A$ is recomputed the size of set $\text{FAR}$ decreases by at least half, assuming $\ell_0$ remains fixed. Since $\ell_0$ changes at most $\epsilon^{-1} \log n$ times, the set $A$ is recomputed at most $\epsilon^{-1} \log^2 n$ times, and vertex $a$ can thus change $O(\epsilon^{-1} \log^2 n|A|) = O(\epsilon^{-1} \log^3 n)$ times.

Finally, for vertex $a$ we maintain $\text{DEPTH}(\text{IN-BFS}(a))$ using ES-tree. Since, $a$ changes at most $O(\epsilon^{-1} \log^3 n)$ times, total time for maintaining $\text{DEPTH}(\text{IN-BFS}(a))$ is $O(mD\epsilon^{-1} \log^3 n)$. Whenever $\text{DEPTH}(\text{IN-BFS}(a))$ falls by a factor of $(1 - \epsilon)$, then we re-set $S_2$ to $\text{NIN}(a, n_q)$. For a fixed $a$, $S_2 = \text{NIN}(a, n_q)$ is updated at most $O(\epsilon^{-1} \log n)$ times. So in total $S_2$ changes at most $O(\epsilon^{-2} \log^4 n)$ times, and the total time for maintaining set $S_2$, throughout the edge insertions is $O(\epsilon^{-2} \log^4 n)$. Thus, the total time taken by the algorithm is $O(\epsilon^{-1} D \log^3 n + \epsilon^{-2} \log^4 n)$, where $D$ denotes the maximum diameter of $G$ throughout the sequence of edge updates. Also the expected number of times the triplet $(S_1, S_2, a)$ changes is $O(\epsilon^{-2} \log^4 n)$.

**Decremental Scenario.** We now discuss the simpler scenario of edge deletions. As before, we initialize $S_1$ to be a uniformly random subset of $V$ containing $n_p$ vertices. Next we compute $\text{OUT-BFS}(S_1)$ and set $a$ to be an arbitrary vertex having maximum depth in $\text{OUT-BFS}(S_1)$. Also $S_2$ is set to $\text{NIN}(a, n_q)$. We store in $\ell_0$ the depth of tree $\text{OUT-BFS}(S_1)$, and as in incremental setting use Theorem 2.1 to dynamically maintain the depth of $\text{OUT-BFS}(S_1)$. This takes $O(mD)$ time in total. Whenever $\text{DEPTH}(\text{OUT-BFS}(S_1))$ exceeds the value $(1 + \epsilon)\ell_0$, then we recompute $\ell_0, a$ and $S_2$. For any fixed $S_1$ such an event happens at most $O(\epsilon^{-1} \log n)$ times, and takes in total $O(\epsilon^{-1} \log^2 n)$ time. Also whenever $S_1 \cap S_2$ is non-empty, then we recompute $S_1, S_2, a, \ell_0$, and reinitialize the Even and Shiloach data-structure. The probability of such an event is inverse polynomial in $n$. So the expected amortized update time for edge deletions is $O(D + \epsilon^{-1} \log n)$. Also if $t_0$ is the time when $\ell_0, a, S_2$ were last updated and $t$ is the current time then $\text{DEPTH}(a, \text{OUT-BFS}(S_1)) = \ell_0, \text{DEPTH}(\text{OUT-BFS}(S_1), \text{DEPTH}(a, \text{OUT-BFS}(S_1)) \in [\ell_0, (1 + \epsilon)\ell_0]$.

The following theorem is immediate from the above discussion and Lemma 3.2.

**Theorem 3.2.** For any $\epsilon \in [0, 1/2]$, and any integers $n_p, n_q \geq 1$ satisfying $n_p n_q = 8n \log n$, there exists an algorithm that incrementally/decrementally maintains for an $n$-vertex directed graph a set-pair $(S_1, S_2)$ of size-bound $(n_p, n_q)$ which is $([p + 2\epsilon \ln E(a), ((q + 2\epsilon) \ln E(a))]-$dominating, for some $a \in V$, and any arbitrary fractions $p, q > 0$ satisfying $p + q = 1$.

The expected amortized update time of the algorithm is $O(\epsilon^{-1} D_{\text{max}} \log^3 n + \epsilon^{-2} \log^4 n)$ in incremental setting and $O(D_{\text{max}} + \epsilon^{-1} \log n)$ in decremental setting, where, $D_{\text{max}}$ denotes the maximum diameter of the graph throughout the sequence of edge updates. Also, the algorithm ensures that with high probability the triplet $(S_1, S_2, a)$ changes at most $O(\epsilon^{-2} \log^4 n)$ times in the incremental setting, and at most $O(\epsilon^{-1} \log n)$ times in the decremental setting.
3.2 Dynamic Algorithms for 1.5-Diameter-Spanners

We consider two models for maintaining the diameter spanners, namely, the explicit model and the implicit model. The explicit model maintains at each stage all the edges of a diameter spanner of the current graph. In the model of implicitly maintaining the diameter spanner, the goal is to have a data-structure that efficiently supports the following operations: (i) UPDATE($e$) that adds to or remove from the graph $G$ the edge $e$, and (ii) QUERY($e$) that checks if the diameter-spanner contains edge $e$.

We first consider the explicit maintenance of diameter-spanners.

Let $A$ be an algorithm that uses Theorem 3.2 to incrementally (or decrementally) maintain at any time $t$, a $\langle [(1/2+\epsilon)\text{InEcc}(a)] \rangle$-dominating set-pair $(S_1, S_2)$ of size bound $\langle \sqrt{n \log n}, \sqrt{n \log n} \rangle$, where $a \in V$. We dynamically maintain a subgraph $H$ which is union of $\text{IN-BFS}(s)$ and $\text{OUT-BFS}(s)$, for $s \in S_1 \cup S_2$. This takes in total $O(mD_{\max}|S_1 \cup S_2|) = O(m \cdot D_{\max}\sqrt{n \log n})$ time, where $D_{\max}$ is the maximum diameter of the graph throughout the sequence of edge updates. Observe that similar to Theorem 3.1, it can be shown that at any time instance subgraph $H$ is a $(1/2 + \epsilon)$-diameter-spanner of $G$, and it contains at most $O(n^{n/\log n})$ edges. Let $T$ be the expected amortized update time of $A$ for maintaining $(S_1, S_2)$, and let $C$ be the total number of times the pair $(S_1, S_2)$ changes throughout the algorithm run. Then the total time for maintaining $H$ is $O(C \cdot m \cdot D_{\max}\sqrt{n \log n} + m \cdot T)$. On substituting the values of $C$ and $T$ from Theorem 3.2, we get that the expected amortized update time of $A$ is $O(e^{-1}D_{\max}\sqrt{n \log n^{1.5}})$ for the decremental setting, and $O(e^{-2}D_{\max}\sqrt{n \log n^{4.5}})$ for the incremental setting.

For the scenario when $D_{\max}$ is large we alter our algorithm as follows. Let $D_0$ be some threshold value for diameter. We maintain a $2$-approximation of $\text{diam}(G)$, say $\delta$, by dynamically maintaining for an arbitrarily chosen vertex $z$, the value $\text{DEPTH}(\text{IN-BFS}(z)) + \text{DEPTH}(\text{OUT-BFS}(z))$. This by Theorem 2.1 takes $O(mn)$ time in total. We now explain another algorithm $B$ which will be effective when $\delta \geq 4D_0$. We sample a uniformly random subset $W$ of $V$ containing $(8n \log n/D_0)$ vertices, and maintain at each stage a subgraph $H_B$ which is union of $\text{IN-BFS}(w)$ and $\text{OUT-BFS}(w)$, for $w \in W$. Also we maintain the value $\text{DEPTH}(\text{OUT-BFS}(W))$. If $\delta \geq 4D_0$, but $\text{DEPTH}(\text{OUT-BFS}(W)) \notin D_0$, we re-sample $W$. When $\delta \geq 4D_0$, then with high probability at each time instance, set $W$ intersects $\pi(x, y)$ for every $x, y \in V$ that satisfy $d_G(x, y) \geq D_0$, and thus $\text{DEPTH}(\text{OUT-BFS}(W)) \leq D_0$. This shows that the expected number of re-samplings for $W$ is $O(1)$, and the total expected runtime of $B$ is $O(mW) = O(mn^2 \log n/D_0)$. Since $\text{DEPTH}(\text{OUT-BFS}(W)) \leq D_0 \leq \delta/4 \leq \text{diam}(G)/2$, it follows that in this case the distance between any two vertices in $H_B$ is at most $1.5\text{diam}(G)$. As long as $\delta \leq 4D_0$, we use algorithm $A$ to maintain a $(1.5 + \epsilon)$-diameter-spanner, we denote the corresponding subgraph by notation $H_A$. Thus $A$ takes in total $O(e^{-2}D_{\max}\sqrt{n \log n^{4.5}})$ time for incremental setting, and $O(e^{-1}D_{\max}\sqrt{n \log n^{1.5}})$ time for decremental setting. On optimizing over $D_0$, we get that the amortized update time of the combined algorithm is $O(e^{-1}n^{1.25} \log^{2.75} n)$ for incremental setting, and $O(e^{-0.5}n^{1.25} \log^{1.25} n)$ for decremental setting.

Theorem 3.3. For any $\epsilon \in [0, 1/2]$ and any incrementally/decrementally changing graph on $n$ vertices, there exists an algorithm for maintaining a $(1.5 + \epsilon)$-diameter-spanner containing at most $O(n^{3/2} \sqrt{\log n})$ edges.

The expected amortized update time of the algorithm is $O((1/\epsilon^2)\sqrt{n}D_{\max}\log n^{4.5})$ for incremental setting and $O((1/\epsilon)\sqrt{n}D_{\max}\log n^{1.5})$ for decremental setting, where, $D_{\max}$ denotes the maximum diameter of the graph throughout the run of the algorithm. Moreover, when $D_{\max}$ is large, the algorithm can be altered so that the expected amortized update time is $O(e^{-1}n^{1.25} \log^{2.75} n)$ for the incremental setting, and $O(e^{-0.5}n^{1.25} \log^{1.25} n)$ for the decremental setting.

We now present the algorithm for implicitly maintaining diameter-spanner. Let $A$ be a Monte-Carlo variant of Theorem 3.2 to incrementally/decrementally maintain a $\langle [(1/2+\epsilon)D] \rangle$-dominating set-pair $(S_1, S_2)$ of size bound $\langle \sqrt{n \log n}, \sqrt{n \log n} \rangle$. So $A$ takes in total $O(e^{-1}mn \log^2 n + me^{-2} \log^4 n)$ time for incremental setting, and $O(mn + me^{-1} \log n)$ time for decremental setting. We also maintain a data-structure for dynamic all-pairs shortest-path problem. For edge-insertions only case, Ausiello et al. [10] gave
an $O(n^3 \log n)$ time algorithm that answers any distance query in constant time, and for edge-deletions only case, Baswana et al. [8] gave an $O(n^3 \log^2 n)$ time Monte-Carlo algorithm that again answers any distance query in constant time. Now in order to check whether or not an edge $e = (u, v)$ lies in $H$, it suffices to check whether or not $e$ is present in either IN-BFS$(s)$ or OUT-BFS$(s)$, for some $s \in S$. We can assume that edge weights are slightly perturbed so that no two distances are identical in $G$. Therefore $e = (u, v)$ lies in OUT-BFS$(s)$, for some $s$, if and only if $d_G(s, v) = d_G(s, u) + d_G(u, v)$. Since the distances queries can be answered in $O(1)$ time, in order to check whether or not $e$ lies in $H$, we perform in the worst case $O(|S_1 \cup S_2|) = O(\sqrt{n \log n})$ distance queries.

**Theorem 3.4.** There exists a data-structure that for any incrementally/decrementally changing $n$-vertex directed graph and any $\epsilon \in \left(\frac{\log n}{n}, \frac{1}{2}\right]$, implicitly maintains a $(1.5 + \epsilon)$-diameter-spanner containing at most $O(n^{3/2} \sqrt{\log n})$ edges. The total time taken by UPDATE operations is $O(\epsilon^{-1} n^3 \log^3 (n))$ for incremental setting, and $O(n^3 \log^3 (n))$ for decremental setting. Each QUERY operation takes $O(\sqrt{n \log n})$ time in the worst case, and the answers are correct with high probability (i.e., failure probability is inverse polynomial in $n$).

### 4 Other Sparse Extremal Distance Spanners

In this section, we show several other constructions of extremal-distance-spanners (diameter-spanners, eccentricity-spanners, and radius spanners) for directed graphs, with various size-stretch trade-offs.

We first present construction of 5/3-diameter-spanners that are sparser than the 1.5-diameter-spanners whenever $D = o(\sqrt{n})$.

**Theorem 4.1.** For any directed graph $G = (V, E)$ with diameter $D$, in $\tilde{O}(mn^{1/3}(D + n/D)^{1/3})$ expected time\(^3\) we can compute a subgraph $H = (V, E' \subseteq E)$ satisfying $\text{diam}(H) \leq \lceil 5D/3 \rceil$ that contains at most $O(n^{4/3}(\log n)^2 D^{1/3})$ edges, where $n$ and $m$ respectively denotes the number of vertices and edges in $G$.

**Proof.** Let $\alpha$ be a parameter to be chosen later on. The construction of $H$ is presented in Algorithm 1.

Consider any two vertices $x, y \in V$. If $A_1$ is $\lceil 2D/3 \rceil$-out-dominating set, then $d_G(s, y) \leq \lceil 2D/3 \rceil$ for some $s \in A_1$. Thus $d_H(x, y) \leq d_H(x, s) + d_H(s, y) = d_G(x, s) + d_G(s, y) \leq D + \lceil 2D/3 \rceil = \lceil 5D/3 \rceil$. Similarly, if $B_2$ is $\lceil 2D/3 \rceil$-in-dominating set, then it can be shown that $d_H(x, y)$ is bounded by $\lceil 5D/3 \rceil$.

Let us next suppose that neither $A_1$ is $\lceil 2D/3 \rceil$-out-dominating, nor $B_2$ is $\lceil 2D/3 \rceil$-in-dominating. Then $A_2$ is $D/3$-in-dominating and $B_1$ is $D/3$-out-dominating. So $d_G(x, A_2), d_G(B_1, y) \leq D/3$. Since $H$ contains IN-BFS$(A_2)$ and OUT-BFS$(B_1)$, there must exists $s_x \in A_2$ and $s_y \in B_1$ such that $d_H(x, s_x) = d_G(x, s_x)$ and $d_H(s_y, y) = d_G(s_y, y) = d_G(B_1, y)$. Since $H$ contains the shortest path between each pair of vertices in $A_2 \times B_1$, we obtain that $d_H(s_x, s_y) = d_G(s_x, s_y) \leq D$. Therefore, $d_H(x, y) \leq d_H(x, s_x) + d_H(s_y, y) + d_H(s_x, s_y) \leq 5D/3$. Let us first analyse size of $H$. We have $O(\alpha \log n)$ shortest-path trees that require a total of $O(n \alpha \log n)$ edges. The shortest paths between

---

\(^3\)Though the time required for computing $H$ is a function of $D$, the algorithm does not need to apriori know the value $D$. 

---

**Algorithm 1: 5/3-Diameter-Spanner Construction**

1. $H \leftarrow$ an empty graph;
2. $(A_1, A_2) \leftarrow (\lceil 2D/3 \rceil, D/3)$-dominating-set-pair of size-bound $\langle \alpha \log n, n/\alpha \rangle$;
3. $(B_1, B_2) \leftarrow (D/3, \lceil 2D/3 \rceil)$-dominating-set-pair of size-bound $\langle n/\alpha, \alpha \log n \rangle$;
4. Add to $H$ the trees IN-BFS$(A_2)$ and OUT-BFS$(B_1)$;
5. **foreach** $s \in A_1 \cup B_2$ **do** add to $H$ union of IN-BFS$(s)$ and OUT-BFS$(s)$;
6. **foreach** $(u, v) \in A_2 \times B_1$ **do** add the edges of the shortest path $\pi_G(u, v)$ to $H$;
7. **return** $H$;

---

have $O(\alpha \log n)$ shortest-path trees that require a total of $O(n \alpha \log n)$ edges. The shortest paths between
all pairs in $A_2 \times B_1$ use in total $O(n^2 D/\alpha^2)$ edges. Thus, the total number of edges in $H$ is $O(n \alpha \log n + n^2 D/\alpha^2)$. This is minimized when $\alpha = \Theta((n D/\log n)^{1/3})$. Therefore, the total number of edges in $H$ is $O(n^{4/3} D^{1/3} \log^{2/3} n)$. Observe that in order to compute $\alpha$, it suffices to have an estimate of $D$. We can easily compute a $2$-approximation for the diameter $D$ in $O(m)$ time, since for any arbitrary vertex $w \in V$, $D \leq \text{depth}(\text{in-BFS}(w)) + \text{depth}(\text{out-BFS}(w)) \leq 2D$, and the depth of in-BFS($w$) and out-BFS($w$) are computable in $O(m)$ time. We now analyse the running time of each step in Algorithm 1. Steps 2 and 3: By Lemma 3.1, the time to compute the set-pairs $(A_1, A_2)$ and $(B_1, B_2)$ is $O(m)$ on expectation. Step 4: This step just takes $O(m)$ time. Step 5 and 6: For each vertex $s \in A_1 \cup B_2 \cup A_2 \cup B_1$, the BFS trees in-BFS($s$) and out-BFS($s$) can be computed in $O(m)$ time. So, this step can be performed in $O(m \cdot |A_1 \cup B_2 \cup A_2 \cup B_1|)$ time. Overall, the total expected runtime of the algorithm is $O(m(|A_1 \cup A_2 \cup B_1 \cup B_2|)) = O(m(\alpha \log n + n/\alpha)) = \tilde{O}(mn^{1/3} D^{1/3} + mn^{2/3} D^{1/3}) = \tilde{O}(mn^{1/3}(D^{1/3} + (n/D)^{1/3})) = \tilde{O}(mn^{1/3}(D + n/D)^{1/3})$.

We next show construction of an $\tilde{O}(n)$ size spanner with additive stretch.

**Theorem 4.2.** For any $d > 0$, and any $n$-vertex directed graph $G = (V, E)$, we can compute a subgraph $H = (V, E') \subseteq (V, E)$ with $O(n + d n^{3/2} \cdot \min\{n^2 \log n, d \log^2 n\})$ edges satisfying $\text{diam}(H) \leq \text{diam}(G) + n/d$.

**Proof.** Let $S$ be random set of $8d \log n$ vertices. We first check that $S$ has non-empty intersection with $N^{\text{out}}(w, n/2d)$ and $N^{\text{in}}(w, n/2d)$, for each $w \in V$, if not then re-sample $S$. The expected computation time of $S$ is $O(n)$. Next we initialize $H$ to union of trees in-BFS($S$) and out-BFS($S$).

If $n^2 \log n \leq d \log^2 n$, then we add to $H$ tree out-BFS($s$), for each $s \in S$. So the number of edges in $H$ is $O(n |S|) = O(nd \log n)$. To prove correctness consider any two vertices $x, y \in V$. There must exists $s \in S$ such that, $d_H(x, s) = d_H(x, S) = d_G(x, S) \leq n/d$ (the last inequality holds since $S$ intersects $N^{\text{out}}(x, n/d)$ and $G$ is unweighted). Also $d_H(s, y) \leq \text{diam}(G)$. Thus, $d_H(x, y) \leq d_H(x, s) + d_H(s, y) \leq \text{diam}(G) + n/d$.

Let us next consider the case $d \log^2 n \leq n^2 \log n$. In this case we add to $H$ a pair-wise distance preserver for each pair of nodes in $S \times S$. Bodwin showed that for any set $\mathcal{S} \subseteq V$ in a directed graph, we can compute a sparse subgraph with at most $O(n + n^{2/3} |\mathcal{S} \times \mathcal{S}|)$ edges that preserves distance between each node pair in $\mathcal{S} \times \mathcal{S}$. So the total number of edges in subgraph $H$ is $O(n + n^{2/3} |\mathcal{S} \times \mathcal{S}|) = O(n + n^{2/3} d \log^2 n)$. Now to prove the correctness consider any two vertices $u, v \in V$, let $x_u, x_v \in S$ be such that $x_u \in N^{\text{out}}(u, n/2d)$ and $x_v \in N^{\text{in}}(v, n/2d)$. Then $d_H(u, s_u) \leq n/2d$, $d_H(s_v, v) \leq n/2d$, and $d_H(s_u, s_v) \leq \text{diam}(G)$. Thus, $d_H(x, y) \leq \text{diam}(G) + n/d$.

On substituting $d = n^{1/6}$ in the Theorem 4.2 we obtain the following results on construction of spanners of size almost linear in $n$.

**Theorem 4.3.** For any $n$-vertex directed graph $G = (V, E)$, we can compute a subgraph $H = (V, E') \subseteq (V, E)$ with $O(n \log^2 n)$ edges satisfying $\text{diam}(H) \leq \text{diam}(G) + n^{5/6}$.

**Theorem 4.4.** For any $n$-vertex directed graph $G = (V, E)$ satisfying $\text{diam}(G) - \omega(n^{5/6})$, we can compute a subgraph $H = (V, E') \subseteq (V, E)$ with $O(n \log^2 n)$ edges satisfying $\text{diam}(H) \leq (1 + o(1))\text{diam}(G)$.

**General (low-stretch or low-size)-Diameter-Spanners.** We show that for any directed graph $G$ we can either (i) compute a diameter-spanner with arbitrarily low stretch, or (ii) compute a diameter-spanner with arbitrarily low size.

**Theorem 4.5.** Let $n_p, n_r > 1$ be integers satisfying $n_p n_r = 8n \log n$, and $p, r > 0$ be fractions satisfying $p + r = 1$. For any directed graph $G = (V, E)$ with $n$ vertices and $m$ edges, in $O(m \max\{n_p, n_r\})$ expected time, we can compute a subgraph $H = (V, E_0 \subseteq E)$ satisfying at least one of the following conditions:

(i) $|E_0| = O(n n_p)$ and $\text{diam}(H) \leq [(1 + p) \text{diam}(G)]$.

(ii) $|E_0| = O(n n_r)$ and $\text{diam}(H) \leq [(1 + r) \text{diam}(G)]$. 

11
Proof. Let $D$ denote the diameter of $G$. Let $(S_1, S_2)$ be a $\langle \lfloor p \ln\text{Ecc}_G(a) \rfloor, \lceil r \ln\text{Ecc}_G(a) \rceil \rangle$-dominating-set-pair of size bound $\langle n_p, n_r \rangle$ obtained from Lemma 3.1 for some $a \in V$ and some integers $n_p, n_r > 1$ satisfying $n_p n_r = 8n \log n$. Let $H_1$ (respectively $H_2$) be the union of the trees IN-BFS$(s)$ and OUT-BFS$(s)$, for each $s \in S_1$ (respectively $S_2$). The time for computing $H_1$ and $H_2$ is derived from $|S_1 \cup S_2|$ BFS computations, plus the time for finding the dominating set-pair $(S_1, S_2)$, which in total is $O(m(n_p + n_r))$ on expectation. Note that the graph $H_1$ contains $O(nn_p)$ edges and $H_2$ contains $O(nn_r)$ edges. Now consider any two vertices $x, y \in V$. If $S_1$ is $\lfloor p \ln\text{Ecc}_G(a) \rfloor$-out-dominating, then there exists $s \in S_1$ such that $d_G(s, y) \leq p \ln\text{Ecc}_G(a) \leq pD$. Since $d_{H_1}(x, s) = d_G(x, s) \leq D$, and $d_{H_1}(s, y) = d_G(s, y) \leq pD$, we have $d_{H_1}(x, y) \leq (1 + p)D$. Similarly, if $S_2$ is $\lceil r \ln\text{Ecc}_G(a) \rceil$-in-dominating, then $d_{H_2}(x, y) \leq \lceil (1 + r)D \rceil$. Thus the claim follows.

As a corollary we obtain following result.

**Corollary 4.1.** Let $\epsilon$ be fraction in range $[0, 1]$. For any directed graph $G = (V, E)$ with $n$ vertices and $m$ edges, in $\tilde{O}(mn^{1-\epsilon})$ expected time, we can compute at least one of the following two graphs.
(i) a subgraph $H_1 = (V, E_1 \subseteq E)$ satisfying $|E_1| = O(n^2 - \sqrt{n \log n})$ and $\text{diam}(H_1) \leq \lceil (1.5 - \epsilon) \text{diam}(G) \rceil$.
(ii) a subgraph $H_2 = (V, E_2 \subseteq E)$ satisfying $|E_2| = O(n^{1+\epsilon} \sqrt{n \log n})$ and $\text{diam}(H_2) \leq \lceil (1.5+\epsilon) \text{diam}(G) \rceil$.

**Remark.** As in Theorem 3.1 all our spanner constructions work also for graphs with non-negative edge-weights, by replacing every use of BFS with Dijkstra’s algorithm. The proofs are analogous, the running time is increased by at most a $\log n$ factor, and the stretch factor of the spanner $H$ only suffers an additive $W$ term, where $W$ is the maximum edge weight in the graph $G$.

The following lemma is a generalization of 5/3-diameter-spanner construction.

**Lemma 4.1.** Let $G$ be an $n$-vertex directed graph with diameter $D$, and $(A_1, A_2)$ and $(B_1, B_2)$ be respectively $\langle \lfloor (2/3 + \epsilon)D \rfloor, (1/3 + \epsilon)D \rangle$ and $\langle (2/3 + \epsilon)D, \lceil (2/3 + \epsilon)D \rceil \rangle$ dominating-set-pairs of size bounds $\langle \alpha \log n, n/\alpha \rangle$ and $\langle n/\alpha, \alpha \log n \rangle$, where $\alpha = (nD)^{1/3}$. Also let $H$ be a subgraph of $G$ consisting of
- IN-BFS$(s)$ and OUT-BFS$(s)$, for $s \in A_1 \cup B_2$,
- the shortest paths $\pi_G(u, v)$, for each $(u, v) \in A_2 \times B_1$,
- IN-BFS$(A_2)$ and OUT-BFS$(B_1)$.

Then $\text{diam}(H) \leq \lceil (5/3 + \epsilon)D \rceil$ and $H$ has at most $O(n^{4/3} D^{1/3} \log n)$ edges.

The next theorem presents dynamic algorithm for maintaining a 5/3-diameter-spanner (we omit the proof as it is analogous to that of Theorem 3.3).

**Theorem 4.6.** For any $\epsilon \in [0, 1/2]$, there exists an algorithm for incrementally/decrementally maintaining a $(5/3 + \epsilon)$-diameter-spanner of an $n$-vertex directed graph with at most $O(n^{4/3} D_{\text{max}}^{1/3} \log n)$ edges, where, $D_{\text{max}}$ denotes the maximum diameter of the graph throughout the run of the algorithm. The expected amortized update time of the algorithm is $O(\epsilon^{-2} n^{1/3} D_{\text{max}} (D_{\text{max}} + n/D_{\text{max}})^{1/3} \log^5 n)$ for the incremental setting and $O(\epsilon^{-1} n^{1/3} D_{\text{max}} (D_{\text{max}} + n/D_{\text{max}})^{1/3} \log^2 n)$ for the decremental setting.

## 5 Eccentricity-spanners and radius-spanners

### 5.1 Almost linear size 2-eccentricity-spanner computation in $\tilde{O}(m)$ time

Observe that a graph $H$ which is union of IN-BFS$(c)$ and OUT-BFS$(c)$, where $c$ is the centre\(^4\) of the graph $G$, is a 2-eccentricity spanner of $G$ containing just $2m$ edges. However, computation time of $H$ is large since the best known algorithm for computing a centre for directed weighted graphs takes $O(mn)$ time.

\(^4\)Centre of a graph is a vertex of minimum out-eccentricity.
In order to obtain a 2-eccentricity-spanner in near optimal time, we first show that for any $n$-vertex graph $G$, we can compute in $O(m)$ time a set $S$ containing $O(\log^2 n)$ vertices such that $\text{DEPTH}(\text{OUT-BFS}(S))$ is at most $\text{rad}(G)$. (Our results hold for the general setting of directed weighted graphs).

Algorithm 2: 2-Eccentricity-Spanner Construction

1. $B_k \leftarrow V$;
2. for $i = k - 1$ to 1 do
3.     $A_i \leftarrow$ uniformly random subset of $B_{i+1}$ of size $\min\{8n^{1/k} \log n, |B_{i+1}|\}$;
4.     $a_i \leftarrow$ vertex of maximum depth in $\text{OUT-BFS}(A_i)$;
5.     $B_i \leftarrow$ a subset of $B_{i+1}$ containing the $n^{i/k}$-closest incoming vertices to $a_i$ that lie in $B_{i+1}$;
6.     if $A_i \cap B_i = \emptyset$ then go to step 3 to re-sample $A_i$, and next recompute $a_i$ and $B_i$;
7. $S \leftarrow B_1 \cup A_1 \cup A_2 \cup \cdots \cup A_{k-1}$;
8. $H \leftarrow$ union of $\text{IN-BFS}(s)$ and $\text{OUT-BFS}(s)$, for $s \in S$;
9. return $H$;

Let $k$ be a parameter to be chosen later on. The construction of set $S$ is very simple and presented in Algorithm 2. The expected runtime of the algorithm is $O(mk + m|S|)$. To observe this note that $|B_{i+1}| = n^{(i+1)/k}$. Now we take $A_i$ to be uniformly random subset of $B_{i+1}$ of size at most $8n^{1/k} \log n$, and $B_i$ contains those $n^{i/k}$ closest incoming vertices to $a_i$ that lie in $B_{i+1}$. Since $B_i, A_i$ are both subsets of $B_{i+1}$, the expected number of re-samplings in step 6 for each $i \in [1, k-1]$ is $O(1)$. So the total time taken by steps 2-6 is $O(mk)$ on expectation, and the time taken by steps 7-9 is $O(m|S|)$.

We next prove the correctness of the algorithm through the following lemmas.

Lemma 5.1. For any index $i \in [1, k]$, the set-pair $(A_i, B_i) \in P(B_{i+1}) \times P(B_{i+1})$ is of size bound $\langle n^{1/k} \log n, n^{i/k} \rangle$ and satisfy that for each $x \in B_{i+1}$, either $\text{DEPTH}(\text{OUT-BFS}(A_i)) \leq \text{OutEcc}(x)$, or $x \in B_i$.

Proof. Let us suppose $d$ is an integer satisfying $\text{DEPTH}(\text{OUT-BFS}(A_i)) = \text{DEPTH}(a_i, \text{OUT-BFS}(A_i)) > d$. Then $\text{IN-BFS}(a_i, d)$ must have empty-intersection with $A_i$. This is possible only when $B_i$ contains the set $B_{i+1} \cap \text{IN-BFS}(a_i, d)$, since $A_i$ intersects $B_i$, but not the set $B_{i+1} \cap \text{IN-BFS}(a, d)$. Notice that $B_{i+1} \cap \text{IN-BFS}(a_i, d)$ contains each vertex $x \in B_{i+1}$ that satisfy $\text{OutEcc}(x) \leq d$. So for any vertex $x \in B_{i+1}$, on substituting $d = \text{OutEcc}(x)$, we get that either $\text{DEPTH}(\text{OUT-BFS}(A_i)) \leq \text{OutEcc}(x)$ or $x \in B_i$. \hfill $\Box$

Lemma 5.2. The size of set $S$ is at most $O(kn^{1/k} \log n)$ and it satisfies the condition that $\text{DEPTH}(\text{OUT-BFS}(S))$ is at most $\text{OutEcc}(x)$, for each $x \in V$.

Proof. Consider any vertex $x \in V$. Let $j \in [1, k]$ be the largest index such that $x \in B_j$. If $j = 1$, then $x \in B_1 \subseteq S$, and thus $\text{DEPTH}(\text{OUT-BFS}(S)) \leq \text{DEPTH}(\text{OUT-BFS}(x)) = \text{OutEcc}(x)$. If $j > 1$, then $x \notin B_{j-1}$, and by Lemma 5.1, $\text{DEPTH}(\text{OUT-BFS}(A_{j-1})) \leq \text{OutEcc}(x)$, which shows that $\text{DEPTH}(\text{OUT-BFS}(S)) \leq \text{OutEcc}(x)$. \hfill $\Box$

Since $\text{DEPTH}(\text{OUT-BFS}(S)) \leq \text{OutEcc}(x)$, for each $x \in V$, it follows that $\text{DEPTH}(\text{OUT-BFS}(S))$ is bounded by $\text{rad}(G)$. On substituting $k = \log_2 n$, we get that $|S| = O(\log^2 n)$, and time for computing $S$ is $O(mk) = O(m \log n)$. To compute a 2-eccentricity spanner $H$, we just take union of $\text{IN-BFS}(s)$ and $\text{OUT-BFS}(s)$, for $s \in S$. For any $x, y \in V$, there will exists a vertex $s \in S$ satisfying $d_G(s, y) \leq \text{rad}(G)$, and so $d_H(x, y) \leq d_H(x, s) + d_H(s, y) \leq \text{OutEcc}_G(x) + \text{rad}(G) \leq 2\text{OutEcc}_G(x)$. Also $H$ is a 2-radius spanner because if $c \in V$ is the vertex in $G$ with minimum eccentricity, then $\text{rad}(H) \leq \text{OutEcc}_H(c) \leq 2\text{OutEcc}_G(c) = 2\text{rad}(G)$. From the above discussion, we obtain the following theorem.

Theorem 5.1. There exists an algorithm that for any directed weighted graph $G = (V, E)$ with $n$ vertices and $m$ edges, computes in $O(m \log^2 n)$ expected time a 2-eccentricity-spanner (and a 2-radius-spanner) of $G$ with at most $O(n \log^2 n)$ edges.
Estimating graph eccentricities. The set $S$ can also help us to obtain a 2-approximation of graph eccentricities. For any vertex $x \in V$, we approximate its out-eccentricity by $\text{OutEcc}'(x) = \max_{s \in S} d_G(x, s) + \text{DEPTH}(\text{OUT-BFS}(S))$. Observe $\text{OutEcc}'(x) \leq \text{OutEcc}_G(x) + \text{rad}(G) \leq 2\text{OutEcc}_G(x)$. Now $\text{OutEcc}'(x) \geq \text{OutEcc}(x)$, because for any $y \in V$, if $s_y \in S$ is the vertex satisfying $d_G(s_y, y) = \text{DEPTH}(y, \text{OUT-BFS}(S))$, then $d_G(x, y) \leq d_G(x, s_y) + d_G(s_y, y) \leq \max_{s \in S} d_G(x, s) + \text{DEPTH}(\text{OUT-BFS}(S)) = \text{OutEcc}'(x)$. Thus $\text{OutEcc}'(x)$ is a 2-approximation of out-eccentricity of $x$. Observe that given the set $S$, in total $O(m \log^2 n)$ time we can compute $\text{OutEcc}'(x)$, for $x \in V$. We thus have the following theorem.

Theorem 5.2. For any directed weighted graph $G = (V, E)$ with $n$ vertices and $m$ edges, we can compute an estimate $\text{OutEcc}'(x)$, for $x \in V$, satisfying $\text{OutEcc}_G(x) \leq \text{OutEcc}'(x) \leq 2\text{OutEcc}_G(x)$ in $O(m \log^2 n)$ expected total time.

A linear size 3-Eccentricity Spanner. Given a 2-approximation of out-eccentricities of all vertices in $G$, we can compute in $O(n)$ time a vertex $w$, satisfying $\text{OutEcc}(w) \leq 2 \text{rad}(G)$, as we can just choose $w$ to be the vertex whose estimate of out-eccentricity is minimum. Now the graph $H$ which is union of IN-BFS($w$) and OUT-BFS($w$) is a 3-eccentricity spanner since for any two vertices $x, y \in V$, $d_H(x, w) \leq \text{OutEcc}(x)$ and $d_H(w, y) \leq \text{OutEcc}_G(w) \leq 2 \text{rad}(G) \leq 2\text{OutEcc}_G(w)$. The total time for computing $H$, is equal to $O(m)$ plus the time for obtaining a 2-approximation of eccentricities which is $O(m \log^2 n)$. We thus conclude with the following theorem.

Theorem 5.3. There exists an algorithm that for any directed weighted graph $G = (V, E)$ with $n$ vertices and $m$ edges, computes in $O(m \log^2 n)$ expected time a 3-eccentricity-spanner (and a 3-radius-spanner) of $G$ with at most $2n$ edges.

5.2 Dynamic Maintenance of Eccentricity-Spanner and Radius-Spanner

We now present our results on dynamic maintenance of eccentricity-spanners.

We first consider the incremental scenario. For any vertex $w \in V$, let $q(w)$ denote the maximum integer such that IN-BFS($w$) truncated to depth $q(w)$, i.e. IN-BFS($w$, $q(w)$), contains at most $\sqrt{n \log n}$ vertices. Observe that for any $w \in V$, we can incrementally maintaining IN-BFS($w$), $N^\text{inc}(w, \sqrt{n \log n})$, and $q(w)$ in a total of $O(mn)$ time, or $O(mD_{\max})$ time when $D_{\max}$ is an upper bound on the diameter of $G$ throughout the run of algorithm. Also we can dynamically maintain a set $S_{2, \text{inc}}(w)$ whose size is at most $\sqrt{n \log n}$ and contains IN-BFS($w$, $(1 - \epsilon)q(w)$) as follows. In the beginning, say at time $t_0$, we initialize $S_{2, \text{inc}}(w) = \text{IN-BFS}_{t_0}(w, (1 - \epsilon)q_{t_0}(w))$, since $S_{2, \text{inc}}(w) \subseteq \text{IN-BFS}_{t_0}(w, q_{t_0}(w)) \subseteq N^\text{inc}_{t_0}(w, \sqrt{n \log n})$, we have $|S_{2, \text{inc}}(w)| \leq \sqrt{n \log n}$. Now we store in $\ell_t$ the value $(1 - \epsilon)q_{t_0}(w)$ and keep adding all those vertices to $S_{2, \text{inc}}(w)$ whose depth in IN-BFS($w$) reaches a value $\leq \ell_0$, as long as $|S_{2, \text{inc}}(w)| \leq \sqrt{n \log n}$. When $|S_{2, \text{inc}}(w)|$ exceeds the value $\sqrt{n \log n}$, then $q(w)$ must have fallen by a ratio of at least $(1 - \epsilon)$, and we at that time recompute $\ell_0$ and $S_{2, \text{inc}}(w)$. Observe that between re-computations of $\ell_0$, the set $S_{2, \text{inc}}(w)$ only grows with time. Now the total time for maintaining $S_{2, \text{inc}}(w)$ is $O(mD_{\max})$; and the number of times it is rebuilt from scratch is at most $O(\epsilon^{-1} \log n)$.

Our incremental algorithm maintains a pair $(S_1, a) \in P(V) \times V$ such that at any time instance $t$, \text{DEPTH}$_t(a, \text{OUT-BFS}(S_1)) \geq (1 + \epsilon)^{-1} \text{DEPTH}$_t(\text{OUT-BFS}(S_1)). Recall, we showed in construction of incremental dominating set-pair that the total time for maintaining such a pair is $O(m \epsilon^{-1} D_{\max} \log^3 n + m \epsilon^{-2} \log^4 n)$, and the number of times the pair $(S_1, a)$ changes is at most $O(\epsilon^{-2} \log^2 n)$. For a given pair $(S_1, a)$, we maintain the set $S_{2, \text{inc}}(a)$ as described above that takes $O(mD_{\max})$ time. So the total time for maintaining triplet $(S_1, a, S_{2, \text{inc}}(a))$ is $O(mD_{\max} \epsilon^{-2} \log^4 n)$, and the number of times it is recomputed from scratch is $O(\epsilon^{-3} \log^3 n)$. The incremental eccentricity spanner is just the subgraph which is union of IN-BFS($s$) and OUT-BFS($s$) for $s \in S_1 \cup S_{2, \text{inc}}(a)$. This maintenance takes $O(\epsilon^{-3} \log^3 n \cdot mD_{\max} \sqrt{n \log n})$ time. To prove the correctness consider any vertex $x \in V$. Let $d$ be OutEcc$_t(x)$, at some time $t$. If
Theorem 5.6. For any $\epsilon \in [0, 1/2]$ and any incrementally changing graph on $n$ vertices, there exists an algorithm for maintaining a $(2 + \epsilon)$-eccentricity-spanner (and a $(2 + \epsilon)$-radius-spanner) containing at most $O(n^{3/2}/\sqrt{\log n})$ edges, whose expected amortized update time is $O((1/\epsilon^3)\sqrt{nD_{max} \log^{5.5} n})$, where $D_{max}$ denotes an upper bound on the maximum diameter of the graph throughout the run of the algorithm.

Let us now focus on decremental scenario. Consider a time instance $t_0$. Let $S_1$ be a uniformly random subset of $V$ of size $\sqrt{n \log n}$ that intersects $N^{|\epsilon_n}(w, \sqrt{n \log n})$, for each $w \in V$, and each time instance $t$. Let $t_0$ be depth of $\text{OUT-BFS}(S_1)$ at time $t_0$. Let $t \geq t_0$ be another time instance such that $\text{DEPTH}_t(\text{OUT-BFS}(S_1)) \leq (1 + \epsilon)\text{DEPTH}_{t_0}(S_1)$. Also let $S_2 = N^{|\epsilon_n}(a, \sqrt{n \log n})$, where $a$ is a vertex of maximum depth in tree $\text{OUT-BFS}(S_1)$. Similar to arguments in Theorem 5.7, it can be shown that at time $t_0$, for each vertex $x \in S_1$, either $\text{DEPTH}_{t_0}(\text{OUT-BFS}(S_1)) \leq (1 + \epsilon)\text{DEPTH}_{t_0}(S_1)$, or $x \in S_2$. So at time $t$, for each vertex $x \in S_1$, either $\text{DEPTH}_t(\text{OUT-BFS}(S_1)) \leq (1 + \epsilon)\text{DEPTH}_{t_0}(\text{OUT-BFS}(S_1)) \leq (1 + \epsilon)\text{OUT-Ecc}_t(x) \leq (1 + \epsilon)\text{OUT-Ecc}_t(x)$ (here the last inequality holds since expected amortized update time can only increase with time), or $x \in S_2$. This in turn implies that, for each vertex $x \in S_1$, either $\text{DEPTH}_t(\text{OUT-BFS}(S_1)) \leq (1 + \epsilon)\text{OUT-Ecc}_t(x)$ or $x \in S_2$. Therefore, $\text{DEPTH}_t(\text{OUT-BFS}(S_1 \cup S_2)) \leq (1 + \epsilon)\text{min}_{x \in V} \text{OUT-Ecc}_t(x)$. This shows that $S_1 \cup S_2$ is some dominating set pair spanning set, because for any $x, y \in V$, there exists an $S \in S_1 \cup S_2$ such that $d_G(s, y) \leq (1 + \epsilon)\text{rad}(G)$, and so $d_H(x, y) \leq d_H(x, s) + d_H(s, y) \leq d_G(s, y) \leq (1 + \epsilon)\text{rad}(G) \leq (2 + \epsilon)\text{OUT-Ecc}_t(x)$. Thus to dynamically maintain a $(2 + \epsilon)$ eccentricity spanner, we need to recompute $a$ and $S_2$ each time the depth of $\text{OUT-BFS}(S_1)$ exceeds by a factor of $(1 + \epsilon)$. Also, if $S_1 \cap S_2$ is non-empty at any time, then we re-sample $S_1$, and compute $a$ and $S_2$ again. However, expected number of re-samplings is at most $O(1)$. As in decremental maintenance of dominating-set-pair diameter-spanner, it can be shown that the expected time to maintain graph $H$ is $O(\epsilon^{-1}\sqrt{n \log n}D_{max} \log n)$, so we conclude with following theorem.

Theorem 5.5. For any $\epsilon \in [0, 1/2]$ and any decrementally changing graph on $n$ vertices, there exists an algorithm for maintaining a $(2 + \epsilon)$-eccentricity-spanner (and a $(2 + \epsilon)$-radius-spanner) containing at most $O(n^{3/2}/\sqrt{\log n})$ edges, whose expected amortized update time is $O((1/\epsilon^3)\sqrt{nD_{max} \log^{1.5} n})$, where $D_{max}$ denotes an upper bound on the maximum diameter of the graph throughout the run of the algorithm.

Dynamically maintaining 2-approximation of graph eccentricities Observe that the above discussed dynamic algorithm for eccentricity-spanners also imply a same time bound algorithm for maintaining a 2-approximation of vertex eccentricities, because if $S$ is $(\text{rad}(G), V)$-dominating-set then for any vertex $x \in V$, the value $d_G(x, s) + \text{DEPT}(\text{OUT-BFS}(S))$ is a 2-approximation of $\text{OUT-Ecc}(x)$. Since the total time for maintaining the values $d_G(x, s) + \text{DEPT}(\text{OUT-BFS}(S))$ for any vertex $x \in V$, is $O(|m| |S|D_{max})$, we obtain the following result.

Theorem 5.6. For any $\epsilon \in [0, 1/2]$, there exists an incremental (and decremental) algorithm that maintains for an $n$-vertex directed graph $G$ a $(2 + \epsilon)$-approximation of graph eccentricities. The expected amortized update time is $O((1/\epsilon^3)\sqrt{nD_{max} \log^{5.5} n})$ for incremental setting and $O((1/\epsilon^3)\sqrt{nD_{max} \log^{1.5} n})$ for decremental setting, where, $D_{max}$ denotes an upper bound on the diameter of the graph throughout the run of the algorithm.
5.3 Subset Eccentricity-Spanner and Radius-Spanner

We begin by defining the notion of subset-dominating-sets

**Definition 5.1 (Subset-Dominating-Set).** For any directed graph $G = (V, E)$ and any set $W \subseteq V$, we say that a set $S \subseteq V$ is $(h, W)$-dominating, if for each $x \in W$, either $d_G(S, x)$ or $d_G(x, S)$ is bounded by $h$. (Observe that $S$ need not be a subset of $W$).

In the following lemma, we provide an efficient construction of $(h, W)$-dominating set.

**Lemma 5.3.** For any directed graph $G = (V, E)$ with $n$ vertices and $m$ edges, and any set $W \subseteq V$, in $O(m)$ expected time we can compute a $(\text{rad}(G|W), W)$-dominating set $S$ which is a subset of $W$ and contains at most $O(\sqrt{|W| \log n})$ vertices.

**Proof.** We take $S_1$ to be a uniformly random subset of $W$ of size $\sqrt{|W| \log n}$. Next we compute a vertex $a \in W$ of maximum depth in $\text{OUT-BFS}(S_1)$. We set $S_2$ to be the set containing the $\sqrt{|W| \log n}$-closest incoming vertices to $a$ lying in set $W$, where ties are arbitrarily broken. With high probability, $S_2 \cap S_1$ is non-empty, if not, then we re-sample $S_1$, and compute $a$ and $S_2$ again. Finally, we set $S = S_1 \cup S_2$. The expected time for computing $S$ is $O(m)$. Now for any integer $d$ satisfying $\text{DEPTH}(\text{OUT-BFS}(S_1), W) \leq d$, the tree $\text{IN-BFS}(a, d)$ must have empty-intersection with $S_1$. This is possible only when $W \cap \text{IN-BFS}(a, d) \subseteq S_2$, since $S_2$ intersects with $S_1$. Observe $W \cap \text{IN-BFS}(a, d)$ contains each vertex $x \in W$ that satisfy $\text{Out Ecc}(x, W) \leq d$. So for any vertex $x \in W$, on substituting $d = \text{Out Ecc}(x, W)$, we get that either $\text{DEPTH}(\text{OUT-BFS}(S_1), W) \leq \text{Out Ecc}(x, W)$ or $x \in S_2$, and in the later case $\text{DEPTH}(\text{OUT-BFS}(S_2), W) \leq \text{Out Ecc}(x, W)$. Therefore, for each $x \in W$, $\text{DEPTH}(\text{OUT-BFS}(S), W) \leq \text{Out Ecc}(x, W)$, and thus $\text{DEPTH}(\text{OUT-BFS}(S), W) \leq \text{rad}(G|W)$. This shows that $S$ is a $(\text{rad}(G|W), W)$-dominating set.

We now discuss the construction of subset eccentricity-spanner and radius-spanner. Let $S$ be a subset of $W$ as obtained from Lemma 5.3, which is $(\text{rad}(G|W), W)$-dominating, and let $H$ be a subgraph of $G$ which is union of $\text{IN-BFS}(s)$ and $\text{OUT-BFS}(s)$, for $s \in S$. The graph $H$ contains $O(n\sqrt{|W| \log n})$ edges, and the time required for computing it is $O(m\sqrt{|W| \log n})$ on expectation. Now consider any two vertices $x, y \in W$. Let $s \in S$ be the vertex satisfying $d_H(s, y) \leq \text{rad}(G|W)$. Then $d_H(x, y) \leq d_H(x, s) + d_H(s, y) = d_G(x, s) + d_G(s, y) \leq \text{Out Ecc}(x, W) + \text{rad}(G|W) \leq 2\text{Out Ecc}(x, W)$. Since for each $y \in W$, $d_H(x, y) \leq 2\text{Out Ecc}(x, W)$, it follows that $\text{Out Ecc}_H(x, W) \leq 2\text{Out Ecc}_G(x, W)$. If $c \in W$ is the vertex satisfying $\text{Out Ecc}_G(x, W) = \text{rad}(G|W)$, then $\text{Out Ecc}_H(x, W) \leq 2 \text{rad}(G|W)$, and therefore, $\text{rad}(H|W) \leq 2 \text{rad}(G|W)$.

**Theorem 5.7.** There exists an algorithm that for any directed graph $G = (V, E)$ with $n$ vertices and $m$ edges, and any $W \subseteq V$ in $O(m\sqrt{|W| \log n})$ expected time computes a subgraph $H = (V, E') \subseteq E$ with at most $O(n\sqrt{|W| \log n})$ edges satisfying (i) $\text{Out Ecc}_H(x, W) \leq 2 \text{Out Ecc}_G(x, W)$, for each $x \in W$, and (ii) $\text{rad}(H|W) \leq 2 \text{rad}(G|W)$.

6 Dynamic Algorithms for maintaining 1.5-approximation of Diameter

We now present our results for $(1.5 + \epsilon)$-approximate maintenance of graph diameter. Let $\mathcal{A}$ be an algorithm that uses Theorem 5.2 to dynamically maintain triplet $(S_1, S_2, a)$ such that and time instance set-pair $(S_1, S_2)$ is $[(1/2 + \epsilon) \text{In Ecc}(a)], [(1/2 + \epsilon) \text{In Ecc}(a)]$-dominating and has size bound $\sqrt{n \log n}, \sqrt{n \log n}$. Let $T(\mathcal{A})$ be the expected amortized update time of $\mathcal{A}$ for maintaining $(S_1, S_2, a)$. Also let $C(\mathcal{A})$ be the total number of times the triplet $(S_1, S_2, a)$ changes throughout the run of algorithm.

Since $(S_1, S_2)$ is $[(1/2 + \epsilon) \text{diam}(G)], [(1/2 + \epsilon) \text{diam}(G)]$-dominating, for any pair of vertices $x, y$ in $V$, we have $d_G(x, y) \leq \max_{s \in S_1 \cup S_2} (1.5 + \epsilon) \max\{\text{In Ecc}(s), \text{Out Ecc}(s)\}$, which in turn is bounded by
[(1.5 + \epsilon)\text{diam}(G)]. Thus, to dynamically maintain a 1.5-approximation of diameter it suffices to maintain \text{DEPTH}(\text{IN-BFS}(s)) and \text{DEPTH}(\text{OUT-BFS}(s)) for each \(s \in S_1 \cup S_2\). This by Theorem 2.1 takes \(O(mD_{\text{max}})\) time in total for any \(s \in S_1 \cup S_2\), where, \(D_{\text{max}}\) denotes the maximum diameter of graph throughout the sequence of edge updates. Observe that the pair \((S_1, S_2)\) also alters at most \(C(A)\) times. So the total time for maintaining a 1.5-approximation of diameter is \(O(|C(A)|mD_{\text{max}}\sqrt{n \log n} + mT(A))\). On substituting the values of \(C(A)\) and \(T(A)\) from Theorem 3.2 we get that the expected amortized update time of \(\mathcal{A}\) is \(O(\epsilon^{-1}D_{\text{max}}\sqrt{n \log^{1.5} n})\) for the decremental setting, and \(O(\epsilon^{-2}D_{\text{max}}\sqrt{n \log^{4.5} n})\) for the incremental setting.

**Theorem 6.1.** For any \(\epsilon \in [0, 1/2]\), there exists an algorithm that incrementally/decrementally maintains for an \(n\)-vertex directed graph \(G\) an approximation \(\hat{D}\) of diameter \(D\) satisfying \(D \leq \hat{D} \leq (1.5 + \epsilon)D\).

The expected amortized update time of the algorithm is \(O(\epsilon^{-2}D_{\text{max}}\sqrt{n \log^{4.5} n})\) for incremental setting and \(O(\epsilon^{-1}D_{\text{max}}\sqrt{n \log^{1.5} n})\) for decremental setting, where, \(D_{\text{max}}\) denotes the maximum diameter of the graph throughout the run of the algorithm.

We next provide an algorithm for maintaining approximate-diameter whose amortized update time is independent of \(D_{\text{max}}\). Let \(D_0\) be some threshold value for diameter. We dynamically maintain a 2-approximation of \(\text{diam}(G)\), say \(\delta\), as follows. We take an arbitrary vertex \(z\) and maintain using Theorem 2.1 the value \(\text{DEPTH}(\text{IN-BFS}(z)) + \text{DEPTH}(\text{OUT-BFS}(z))\), it is easy to verify that this indeed is a 2-approximation of the diameter. This takes \(O(mn)\) time in total. Our main algorithm for maintaining a \((1.5 + \epsilon)\) approximation is a combination of two algorithms \(\mathcal{A}\) and \(\mathcal{B}\) such that \(\mathcal{A}\) is effective only when \(\delta \geq 6D_0\) and \(\mathcal{B}\) is effective only when \(\delta \leq 6D_0\).

Let \(W\) be a uniformly random subset of \(V\) containing \((8n \log n / D_0)\) vertices, computed in the beginning of the algorithm. Algorithm \(\mathcal{A}\) maintains at each stage the value \(\omega = \max_{w \in W} \max\{\text{out Ecc}(w), \text{in Ecc}(w)\}\) using Theorem 2.1 the total time for this maintenance is \(O(mn|W|) = O(mn^2 \log n / D_0)\). In order to report a \((1.5 + \epsilon)\)-approximation of the diameter, the algorithm reports the value \(3\omega / 2\). We now prove the correctness of Algorithm \(\mathcal{A}\). With high probability, throughout the run of the algorithm, set \(W\) intersects \(\pi(x, y)\) for every \(x, y \in V\) that satisfy \(d_G(x, y) \geq D_0\). Since \(\delta \geq 6D_0\), the actual diameter of \(G\) must be at least \(3D_0\). Let \(a, b\) be any two vertices in \(G\) satisfying \(d_G(a, b) = \text{diam}(G)\), then with high probability \(W\) will contain a vertex, say \(w\), that lies in the prefix of \(\pi(a, b)\) consisting of first \(D_0\) vertices. Observe that \(\text{out Ecc}(w) \geq d_G(w, b) \geq \text{diam}(G) - D_0 \geq 2\text{diam}(G)/3\). Since \(W\) contains at least one vertex whose eccentricity lies in range \([2\text{diam}(G)/3, \text{diam}(G)]\), it follows that \(1.5\omega\) must lie in range \([\text{diam}(G), 1.5\text{diam}(G)]\). The algorithm \(\mathcal{B}\) uses a Monte-Carlo variant of the algorithm presented in Theorem 6.1 to maintain a \((1.5 + \epsilon)\) approximation of diameter, and thus takes in total \(O(\epsilon^{-2}D_0m\sqrt{n \log^{4.5} n})\) time for incremental setting, and \(O(\epsilon^{-1}D_0m\sqrt{n \log^{1.5} n})\) time for decremental setting. On optimizing over \(D_0\), we get that the amortized update time of the combined algorithm is \(O(\epsilon^{-1}n^{1.25} \log^{2.75} n)\) for incremental setting, and \(O(\epsilon^{-0.5}n^{1.25} \log^{1.25} n)\) for decremental setting.

**Theorem 6.2.** For any \(\epsilon \in [0, 1/2]\), there exists a Monte-Carlo algorithm for incrementally/decrementally maintaining for an \(n\)-vertex directed graph \(a\) \((1.5 + \epsilon)\) approximation of diameter. The algorithm outputs a correct approximation with high probability, and its amortized update time is \(O(\epsilon^{-1}n^{1.25} \log^{2.75} n)\) for incremental setting, and \(O(\epsilon^{-0.5}n^{1.25} \log^{1.25} n)\) for decremental setting.

### 7 Fault-Tolerant: Diameter, Diameter-Spanners, and Eccentricity-Spanners

In order to compute fault-tolerant data-structures, our first step is to compute a set \(S_1\) of size \(\sqrt{8n \log n}\) that has non-empty intersection with \(N^m_G(w, \sqrt{n \log n})\), for each vertex \(w \in V\), and each possible failure \(x \in V \cup E\). A trivial way to even verify whether \(S_1\) satisfies the aforesaid condition requires \(O(mn^2)\) time,
since we have \( n \) choices for vertex \( w \), \( n \) choices for failures in trees \( \text{IN-BFS}(w)/\text{OUT-BFS}(w) \), and finally computing the trees \( \text{IN-BFS}_{G/x}(w)/\text{OUT-BFS}_{G/x}(w) \) requires \( O(n) \) time.

We first show a randomized computation of \( S_1 \) that takes \( \tilde{O}(\max\{n^{2.5}, mn\}) \) time. Throughout this section let \( r \) denote the value \( \sqrt{8n \log n} \). Also let \( \mathcal{O} \) denote the distance-sensitivity-oracle for directed graphs \([23, 12]\) that given any \( u, v \in V \) and \( x \in V \cup E \) can output the last edge on \( \pi_{G/x}(u, v) \) in constant time. This data structure can be computed in \( \tilde{O}(mn) \) time and takes \( O(n^2 \log n) \) space. We initialize \( S_1 \) to be a uniformly random subset of \( V \) of size \( r \). For each \( w \in V \), we compute \( \text{IN-BFS}(w) \) and check if \( S_1 \) intersects \( N^{in}(w, r) \), if it doesn’t even for a single vertex \( w \), then we re-sample \( S_1 \). Next for each possible vertex failure \( x \in N^{in}(w, r) \) (or edge failure \( x \in \text{IN-BFS}(w) \) with both end-points in \( N^{in}(w, r) \)), we compute the tree \( \text{IN-BFS}_{G/x}(w) \). Observe that \( x \) has at most \( O(r) = O(\sqrt{n}) \) relevant choices, as for any other remaining option from \( E \cup V \), the set \( N^{in}(w, r) \) remains unaltered. Also computation of tree \( \text{IN-BFS}_{G/x}(w) \) can be performed in \( O(n) \) time using \( \mathcal{O} \). Once we have tree \( \text{IN-BFS}_{G/x}(w) \), we check again if \( S_1 \) intersects \( N^{in}_{G/x}(w, r) \), if it doesn’t then we re-sample \( S_1 \). The expected number of re-samplings to compute the desired \( S_1 \) is \( O(1) \). Thus, the total expected time to compute \( S_1 \) is \( \tilde{O}(\max\{n^{2.5}, mn\}) \).

The following theorem shows construction of diameter-spanner oracle that after any edge or vertex failure reports a 1.5-diameter spanner, containing at most \( \tilde{O}(n^{1.5}) \) edges in \( \tilde{O}(n^{1.5}) \) time.

**Theorem 7.1.** Any \( n \)-vertex directed graph \( G = (V, E) \), can be preprocessed in \( \tilde{O}(\max\{n^{2.5}, mn\}) \) expected time to obtain an \( \mathcal{O}(\max\{n^{2.5}, mn\}) \) size data structure \( \mathcal{D} \) that after any edge or vertex failure \( x \), reports a 1.5-diameter spanner of graph \( G \setminus x \) containing at most \( O(n\sqrt{n \log n}) \) edges in \( O(n\sqrt{n \log n}) \) time.

Moreover, given any edge \( e \) and any failure \( x \), the data-structure can answer the query of whether or not \( e \) lies in a 1.5-diameter-spanner of graph \( G \setminus x \) in \( O(\sqrt{n \log n}) \) time.

**Proof.** We compute the set \( S_1 \) stated in beginning of the section, tree \( \text{OUT-BFS}(S_1) \), and a vertex \( a \) having maximum depth in \( \text{OUT-BFS}(S_1) \). For each edge or vertex \( x \) lying in \( \text{OUT-BFS}(S_1) \) we compute and store (i) the vertex \( a_x \) of maximum depth in \( \text{OUT-BFS}_{G/x}(S_1) \), (ii) the set \( N^{in}_{G/x}(a_x, r) \). Also for each vertex failure \( x \in N^{in}(a, r) \) (or edge failure \( x \in \text{IN-BFS}(a) \) with both end-points in \( N^{in}(a, r) \)), we compute and store \( N^{in}_{G/x}(a, r) \). This takes \( O(nr + r^2) = O(n\sqrt{n \log n}) \) space. Next, we compute the \( O(n^2 \log n) \) spaced distance sensitivity oracle \( \mathcal{O} \) from \([23, 12]\). We assume that the edge weights in \( G \) are slightly perturbed so that all distances in \( G \) are distinct even after an edge/vertex failure. Therefore, (i) for any \( w \in V \) and \( x \in V \cup E \), in linear time oracle \( \mathcal{O} \) can output \( \text{IN-BFS}_{G/x}(w) \) and \( \text{OUT-BFS}_{G/x}(w) \); (ii) given any \( w \in V \), \( x \in V \cup E \), and \( e \in E \), in constant time \( \mathcal{O} \) can output whether or not \( e \) lies in \( \text{IN-BFS}_{G/x}(w) \) and \( \text{OUT-BFS}_{G/x}(w) \). Observe that the total pre-processing time is \( \tilde{O}(\max\{n^{2.5}, mn\}) \).

We now explain the query process. Given a failing edge/vertex \( x \), we first extract a vertex \( a_0 \) having maximum depth in \( \text{OUT-BFS}_{G/x}(S_1) \) and a set \( S_2 \) consisting of vertices \( N^{in}_{G/x}(a_0, r) \). Extracting this information from \( \mathcal{D} \) takes \( O(r) = O(\sqrt{n \log n}) \) time. To output a 1.5-diameter-spanner we just output union of \( \text{IN-BFS}(s) \) and \( \text{OUT-BFS}(s) \) for \( s \in S_1 \cup S_2 \), recall that these trees are computable from \( \mathcal{O} \) in linear time. Using the same arguments as in Theorem 3.1 it can be shown the outputted graph will be a 1.5-diameter-spanner. This takes \( O(nr) = O(n\sqrt{n \log n}) \) time. To verify whether or not a given edge \( e \) lies in the 1.5-diameter-spanner, we iterate over each \( s \in S_1 \cup S_2 \), and check whether or not \( e \) lies in \( \text{IN-BFS}(s) \cup \text{OUT-BFS}(s) \). This takes \( O(\sqrt{n \log n}) \) time, for any edge \( e \). Also observe that, using the same arguments as in Theorem 6.1 it can be shown that the value \( 1.5(\max_{s \in (S_1 \cup S_2)} \{ \text{InEcc}(s), \text{OutEcc}(s) \}) \) is a 1.5-approximation of the diameter of graph \( G \setminus x \).

We now present our diameter-sensitivity-oracle. Observe that a trivial diameter-sensitivity-oracle would be to compute a \((D/2, [D/2])\) dominating set-pair \((S_1, S_2)\) of size bound \((r, r)\), and a 1.5-diameter-spanner \( H \) which is union of \( \text{IN-BFS}(s) \) and \( \text{OUT-BFS}(s) \), for \( s \in S_1 \cup S_2 \). If a failure \( x \) is not in \( H \), then \( 1.5(\max_{s \in (S_1 \cup S_2)} \{ \text{InEcc}(s), \text{OutEcc}(s) \}) \) would still be a 1.5-diameter approximation of graph \( G \setminus x \).
If a failure $x$ lies in $H$, then it has at most $O(n\sqrt{n \log n})$ choices, and for each of possible choice we can compute and store again a 1.5-diameter-approximation in $O(n \sqrt{n \log n})$ time. Thus total time for this procedure is $O(mn^2 \log n)$. Now from Theorem 7.1 $G$ can be preprocessed in $\tilde{O}(\max\{n^{2.5}, mn\})$ expected time to compute a data-structure $D$ that given any edge or vertex failure $x \in H$, computes in $O(n \sqrt{n \log n})$ time a 1.5-diameter-spanner of $G \setminus x$. Moreover, we also showed that in the same time it can compute a 1.5-approximation of the diameter of graph $G \setminus x$. Since there are $O(n \sqrt{n \log n})$ choices for $x$, and for each such choice it takes $O(n \sqrt{n \log n})$ time to compute a 1.5-diameter-approximation, the total time of this process is $\tilde{O}(n^3)$. We thus conclude with following theorem.

**Theorem 7.2.** Any $n$-vertex directed graph $G = (V, E)$, can be preprocessed in $\tilde{O}(n^3)$ expected time to obtain an $O(n \sqrt{n \log n})$ size data-structure $D$ that after any any edge or vertex failure $x$, reports a 1.5-approximation of diameter of graph $G \setminus x$ in constant time.

The data-structure for fault-tolerant-eccentricity-spanner is exactly similar to diameter-spanner data-structure from Theorem 7.1. The proof of correctness follows from the fact that in Theorem 7.1 we essentially after any failure $x$, first compute a valid-set-pair $(S_1, S_2)$ for $G \setminus x$, and next output union of shortest-path-trees in $S_1 \cup S_2$. Using the arguments in Theorem 5.7 it can be shown that $S_1 \cup S_2$ is $\langle \rad(G \setminus x), V \rangle$-dominating, and therefore, the outputted graph is also a 2-eccentricity-spanner.

**Theorem 7.3.** Any $n$-vertex directed graph $G = (V, E)$, can be preprocessed in $\tilde{O}(\max\{n^{2.5}, mn\})$ expected time to obtain an $O(\max\{n^{2.5}, mn\})$ size data-structure $D$ that after any any edge or vertex failure $x$, reports a 2-eccentricity spanner of graph $G \setminus x$ containing at most $O(n \sqrt{n \log n})$ edges in $O(n \sqrt{n \log n})$ time.

Moreover, given any edge $e$ and any failure $x$, the data-structure can answer the query of whether or not $e$ lies in 2-eccentricity spanner of graph $G \setminus x$ in $O(\sqrt{n \log n})$ time.

## 8 Lower Bounds for Diameter Spanners

In this section we prove lower bounds for the number of edges in a diameter spanners with a specific stretch.

**Theorem 8.1.** For every $n$ and every $t$, there exists an $n$-vertex directed graph $G = (V, E)$ with diameter $2(t + 1)$ such that any subgraph $H = (V, E' \subseteq E)$ of $G$ with $\diam(H) \leq 3t + 1$ contains $\Omega(n^2/t^2)$ edges.

**Proof.** Let $N$ be such that $n = N(2t + 2)$. The construction of $G$ is as follows. The vertex set $V(G)$ comprises of four sets $A, B, C, D$ respectively of size $tN, N, N$, and $tN$. The vertices in $A$ are denoted by $a_{k,i}$ where $k \in [1, t]$ and $i \in [1, N]$. The vertices in $B$ are denoted by $b_i$ where $i \in [1, N]$. The vertices in $C$ are denoted by $c_j$ where $j \in [1, N]$. The vertices in $D$ are denoted by $d_{k,j}$ where $k \in [1, t]$ and $j \in [1, N]$. The edges in $G$ are as follows: (i) each vertex $a_{k,i} \in A$ has one out-going edge, namely $(a_{k,i}, a_{k+1,i})$ if $k < t$ and $(a_{k,i}, b_i)$ when $k = t$; (ii) between sets $B$ and $C$ there is a complete bipartite graph, that is, each $(b_i, c_j)$ is an edge; (iii) each vertex $d_{k,j} \in D$ has one incoming edge, namely $(d_{k-1,j}, d_{k,j})$ if $k > 1$ and $(c_j, d_{k,j})$ when $k = 1$; (iv) for each $x \in B \cup C \cup D$ and each $a_{1,i} \in A$, there is an edge $(x, a_{1,i})$ in $G$. See Figure 1.

We will show that the diameter of $G$ is at most $2(t + 1)$.

- In order to focus on vertex pairs in product $(B \cup C \cup D) \times V(G)$, consider any vertex $x \in B \cup C \cup D$.
  - For $a_{k,i} \in A$, $(x, a_{1,i}, a_{2,i}, \ldots , a_{k,i})$ is a path of length at most $2t + 2$.
  - For $b_i \in B$, $(x, a_{1,i}, a_{2,i}, \ldots , a_{t,i}, b_i)$ is a path of length at most $2t + 2$.
  - For $c_j \in C$, $(x, a_{1,j}, a_{2,j}, \ldots , a_{t,j}, b_j, c_j)$ is a path of length at most $2t + 2$.
  - For $d_{k,j} \in D$, $(x, a_{1,j}, a_{2,j}, \ldots , a_{t,j}, b_j, c_j, d_{1,j}, d_{2,j}, \ldots , d_{k,j})$ is a path of length at most $2t + 2$. 
Next consider any vertex $a_{k,i} \in A$.
- For $a_{k',i'} \in A$, $(a_{k,i}, a_{k+1,i}, \ldots, a_{t,i}, b_i, a_{1,i'}, a_{2,i'}, \ldots, a_{k',i'})$ is a path of length at most $2t + 2$.
- For $b_{i'} \in B$, $(a_{k,i}, a_{k+1,i}, \ldots, a_{t,i}, b_i, a_{1,i'}, a_{2,i'}, \ldots, a_{t,i'}, b_{i'})$ is a path of length at most $2t + 2$.
- For $c_j \in C$, $(a_{k,i}, a_{k+1,i}, \ldots, a_{t,i}, b_i, c_j)$ is a path of length at most $2t + 2$.
- For $d_{k,j} \in D$, $(a_{k,i}, a_{k+1,i}, \ldots, a_{t,i}, b_i, c_j, d_{1,j}, d_{2,j}, \ldots, d_{t,j})$ is a path of length at most $2t + 2$.

To verify that the diameter of $G$ is exactly $2t + 2$, observe that the distance between vertices $d_{t,1}$ and $d_{t,N}$ in $G$ is equal to $2t + 2$.

Now on removal of any edge $(b_i, c_j)$ from $G$, the distance between $a_{1,i}$ and $d_{t,j}$ becomes $3t + 2$, since any shortest path from $a_{1,i}$ to $d_{t,j}$ in $G \setminus \{b_i, c_j\}$ has form $(a_{1,i}, a_{2,i}, \ldots, a_{t,i}, b_i, a_{1,i'}, a_{2,i'}, \ldots, a_{t,i'}, b_{i'}, c_j, d_{1,j}, d_{2,j}, \ldots, d_{t,j})$, were $i' \neq i$. This shows that any subgraph $H$ of $G$ whose diameter is at most $3t + 1$ must contain all the edges in set $B \times C$, that is, it should have at least $N^2 = n^2/(2t + 2)^2 = \Omega(n^2/\ell^2)$ edges.

We present below our lower-bound construction for $5/3$-diameter spanner for undirected graphs, which can be extended to directed graphs by simply making each edge bidirectional.

**Theorem 8.2.** For every $n$, there exists an $n$-vertex graph $G = (V, E)$ with $O(n^{3/2})$ edges, for which any subgraph $H = (V, E' \subseteq E)$ of $G$ with $\text{diam}(H) \leq (5/3 - \epsilon)\text{diam}(G)$ must have $\Omega(n^{3/2})$ edges.

**Proof.** We assume $n = 2\alpha^2$, for some integer $\alpha$. Let $A = (a_{ijk}, B = (b_{ijk}), C = (c_{ijk}), D = (d_{ijk})$ with $i, j \in [1, \alpha], k \in [1, 2]$. These will form partition of vertex set of $G$. The edges in $G$ are represented by Table 2. It can be verified that $G$ contains $O(n^{3/2})$ edges.

We next show that diameter of $G$ is 3 by doing a case by case analysis.

- For pair $(a_{ijk}, a_{i0j0k}) \in A \times A$: $(a_{ijk}, b_{ijk}, a_{i0j0k}, a_{i0j0k})$ is a path of length 3.
- For pair $(a_{ijk}, b_{i0j0k}) \in A \times B$: $(a_{ijk}, a_{ijk}, b_{ijk}, b_{ijk}, b_{i0j0k})$ is a path of length 3.
Consider any six indices \( i, j, i_0, j_0 \in [1, \alpha], k, k_0 \in [1, 2] \) such that \( i \neq i_0, j \neq j_0, \) and \( k \neq k_0. \) Let \( G_0 \) be the graph obtained from \( G \) by removing the edges \((a_{ijk}, b_{iojko}), (a_{ijk}, b_{ijoik0}), (c_{ijo2k0}, d_{iojko}), (c_{ijo2k0}, d_{iojko})\) and \((c_{ijo2k0}, d_{iojko})\) from \( G \). We will argue that distance between vertices \( a_{ijk} \) and \( d_{iojko} \) in \( G_0 \) is at least 5. Let us assume on the contrary that \( P = (w_0, w_1, w_2, w_3, w_4) \) is a path in \( G_0 \) of length 4 between \( w_0 = a_{ijk} \) and \( w_4 = d_{iojko}. \) Then \( P \) can move from \( A \to B, B \to C, \) and \( C \to D \) only once during which it must have reversed all three indices (that is, from \( i \) to \( i_0, \) \( j \) to \( j_0, \) and \( k \) to \( k_0). \) Since \((a_{ijk}, b_{ioj})\) and \((a_{ijk}, b_{ijo})\) are not edges in \( G_0, \) the change of either of the first two indices cannot take place on first edge \((w_0, w_1).\) Similarly we can say that the change of either of the first two indices cannot take place on last edge \((w_3, w_4).\) Also either \((w_1, w_2)\) or \((w_2, w_3)\) must be an edge from \( B \times C, \) and while passing through it the first two indices cannot change. Hence we are left with only single edge, and two changes in indices which is not possible. So the diameter of graph \( G_0 \) must be at least 5.

Therefore any subgraph \( H \) of \( G \) that is able to approximate diameter of \( G \) within \((5/3 - \epsilon)\) stretch factor must contain for each \( i, j, i_0, j_0 \in [1, \alpha], \) \( i \neq i_0, j \neq j_0, k, k_0 \in [1, 2] \) either of the four edges - \((a_{ijk}, b_{ioj}), (a_{ijk}, b_{ijo}), (c_{ijo2k0}, d_{iojko}), (c_{ijo2k0}, d_{iojko}).\) This shows that \( H \) must contain \( \Omega(n^{3/2}) \) edges.  

\[ \text{Acknowledgements} \]

We would like to thank Liam Roditty for helpful discussions.

\[ \text{References} \]

[1] Amir Abboud and Greg Bodwin. The 4/3 additive spanner exponent is tight. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing (STOC)*, pages 351–361, 2016.

[2] D. Aingworth, C. Chekuri, P. Indyk, and R. Motwani. Fast estimation of diameter and shortest paths (without matrix multiplication). *SIAM Journal on Computing*, 28(4):1167–1181, 1999.
[3] Donald Aingworth, Chandra Chekuri, Piotr Indyk, and Rajeev Motwani. Fast estimation of diameter and shortest paths (without matrix multiplication). SIAM J. Comput., 28(4):1167–1181, 1999.

[4] Ingo Althöfer, Gautam Das, David Dobkin, Deborah Joseph, and José Soares. On sparse spanners of weighted graphs. Discrete & Computational Geometry, 9(1), 1993.

[5] K. Alzoubi, X. Y. Li, Y. Wang, P. J. Wan, and O. Frieder. Geometric spanners for wireless ad hoc networks. IEEE Transactions on Parallel and Distributed Systems, 14(4):408–421, 2003.

[6] Giorgio Ausiello, Giuseppe F. Italiano, Alberto Marchetti-Spaccamella, and Umberto Nanni. Incremental algorithms for minimal length paths. J. Algorithms, 12(4):615–638, 1991.

[7] Arturs Backurs, Liam Roditty, Gilad Segal, Virginia Vassilevska Williams, and Nicole Wein. Towards tight approximation bounds for graph diameter and eccentricities. In Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC), pages 267–280, 2018.

[8] Surender Baswana, Ramesh Hariharan, and Sandeep Sen. Improved decremental algorithms for maintaining transitive closure and all-pairs shortest paths. J. Algorithms, 62(2):74–92, 2007.

[9] Surender Baswana, Telikepalli Kavitha, Kurt Mehlhorn, and Seth Pettie. Additive spanners and ($\alpha$, $\beta$)-spanners. ACM Trans. Algorithms, 7(1):5:1–5:26, December 2010.

[10] Surender Baswana and Sandeep Sen. A simple linear time algorithm for computing a (2k - 1)-spanner of $o(n^{1+1/k})$ size in weighted graphs. In Proceedings of the 30th International Conference on Automata, Languages and Programming (ICALP), pages 384–396, 2003.

[11] Surender Baswana and Sandeep Sen. Approximate distance oracles for unweighted graphs in expected $o(n^2)$ time. ACM Trans. Algorithms, 2(4):557–577, 2006.

[12] Aaron Bernstein and David R. Karger. A nearly optimal oracle for avoiding failed vertices and edges. In Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009, pages 101–110, 2009.

[13] Greg Bodwin. Linear size distance preservers. In Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pages 600–615, 2017.

[14] Shiri Chechik. New additive spanners. In Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pages 498–512, 2013.

[15] Shiri Chechik, Daniel H. Larkin, Liam Roditty, Grant Schoenebeck, Robert Endre Tarjan, and Virginia Vassilevska Williams. Better approximation algorithms for the graph diameter. In Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014, pages 1041–1052, 2014.

[16] E. Cohen. Fast algorithms for constructing t-spanners and paths with stretch t. SIAM Journal on Computing, 28(1):210–236, 1998.

[17] Edith Cohen. Polylog-time and near-linear work approximation scheme for undirected shortest paths. J. ACM, 47(1):132–166, 2000.

[18] D. Coleman, I. A. ucan, M. Moll, K. Okada, and N. Correll. Experience-based planning with sparse roadmap spanners. In 2015 IEEE International Conference on Robotics and Automation (ICRA), pages 900–905, 2015.
[19] Lenore J Cowen. Compact routing with minimum stretch. *J. Algorithms*, 38(1):170–183, 2001.

[20] Lenore J. Cowen and Christopher G. Wagner. Compact roundtrip routing in directed networks. *Journal of Algorithms*, 50(1):79 – 95, 2004.

[21] Marek Cygan, Harold N. Gabow, and Piotr Sankowski. Algorithmic applications of baur-strassen’s theorem: Shortest cycles, diameter, and matchings. *J. ACM*, 62(4):28:1–28:30, 2015.

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

[23] Camil Demetrescu, Mikkel Thorup, Rezaul Alam Chowdhury, and Vijaya Ramachandran. Oracles for distances avoiding a failed node or link. *SIAM J. Comput.*, 37(5):1299–1318, 2008.

[24] Andrew Dobson and Kostas E. Bekris. Sparse roadmap spanners for asymptotically near-optimal motion planning. *The International Journal of Robotics Research*, 33(1):18–47, 2014.

[25] Michael Elkin. Computing almost shortest paths. In *Proceedings of the 20th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 53–62, 2001.

[26] Michael Elkin and David Peleg. Approximating k-spanner problems for k > 2. In *Proceedings of the 8th International IPCO Conference on Integer Programming and Combinatorial Optimization*, pages 90–104, 2001.

[27] Michael Elkin and David Peleg. The client-server 2-spanner problem with applications to network design. In *Proceedings of the 8th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, pages 117–132, 2001.

[28] Michael Elkin and David Peleg. (1 + $\epsilon, \beta$)-spanner constructions for general graphs. *SIAM J. Comput.*, 33(3):608–631, March 2004.

[29] P. Erds. Extremal problems in graph theory. In *In Theory Of Graphs and its Applications (Proc. Sympos. Smolenice)*, pages 29–36, 1964.

[30] S. Even and Y. Shiloach. An on-line edge deletion problem. *J. ACM*, 28(1):1–4, 1981.

[31] Jie Gao and Dengpan Zhou. The emergence of sparse spanners and greedy well-separated pair decomposition. In *Proceedings of the 12th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT)*, pages 50–61, 2010.

[32] Cyril Gavoille and Christian Sommer. Sparse spanners vs. compact routing. In *Proceedings of the 23rd Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 225–234, 2011.

[33] T.W. Haynes, S. Hedetniemi, and P. Slater. *Fundamentals of Domination in Graphs*. Chapman & Hall/CRC Pure and Applied Mathematics. Taylor & Francis, 1998.

[34] M.A. Henning and A. Yeo. *Total Domination in Graphs*. Springer Monographs in Mathematics. Springer New York, 2014.

[35] Monika Henzinger, Andrea Lincoln, Stefan Neumann, and Virginia Vassilevska Williams. Conditional hardness for sensitivity problems. In *8th Innovations in Theoretical Computer Science Conference, ITCS 2017, January 9-11, 2017, Berkeley, CA, USA*, pages 26:1–26:31, 2017.

[36] Stephan Holzer, David Peleg, Liam Roditty, and Roger Wattenhofer. Distributed 3/2-approximation of the diameter. In *Distributed Computing - 28th International Symposium, DISC 2014, Austin, TX, USA, October 12-15, 2014. Proceedings*, pages 562–564, 2014.
Proof of Lemma 3.1 Let $S_1$ be a uniformly random subset of $V$ of size $n_p$. We take $a$ to be the vertex of the maximum depth in $\text{OUT-BFS}(S_1)$. Also, $S_2$ is set to $N^{\text{in}}(a, n_q)$, which is computable in just $O(m)$ time. By Lemma 2.1, with high probability, the set $N^{\text{in}}(a, n_q)$ contains a vertex of $S_1$, if not, then we re-sample $S_1$, and compute $a$ and $S_2$ again. The number of times we do re-sampling is $O(1)$ on expectation, thus the runtime for computing $(S_1, S_2)$ is $O(m)$ on expectation. Now for any positive integer $d$, if $S_1$ is not $d$-out-dominating (that is $\text{DEPTH}(\text{OUT-BFS}(S_1)) \not\leq d$), then $\text{IN-BFS}(a, d)$ must have empty-intersection with $S_1$. This is possible only when $\text{IN-BFS}(a, d)$ is a strict subset of $S_2 = N^{\text{in}}(a, n_q)$, since the later set intersects with $S_1$. In such a case for any $v \in V$, $d_G(v, S_2) \leq d_G(v, \text{IN-BFS}(a, d)) \leq \max\{0, d_G(v, a) - d\}$. On substituting $d = \lceil p \text{ InEcc}(a) \rceil$, we have that either $S_1$ is $\lceil p \text{ InEcc}(a) \rceil$-out-dominating or $S_2$ is $\lceil q \text{ InEcc}(a) \rceil$-in-dominating.