Fault-tolerant approximate shortest-path trees

Algorithmica (2018) 80:3437–3460
https://doi.org/10.1007/s00453-017-0396-z

Fault-Tolerant Approximate Shortest-Path Trees

Davide Bilò[1] · Luciano Gualà[2] · Stefano Leucci[3] · Guido Proietti[4,5]

Received: 30 September 2016 / Accepted: 10 November 2017 / Published online: 15 November 2017

Abstract The resiliency of a network is its ability to remain *effectively* functioning also when any of its nodes or links fails. However, to reduce operational and set-up costs, a network should be small in size, and this conflicts with the requirement of being resilient. In this paper we address this trade-off for the prominent case of the *broadcasting* routing scheme, and we build efficient (i.e., sparse and fast) *fault-tolerant approximate shortest-path trees*, for both the edge and vertex *single-failure* case. In particular, for an n-vertex non-negatively weighted graph, and for any constant $\varepsilon > 0$,

A preliminary version of this paper appeared on the *Proceedings of the 22nd European Symposium on Algorithms (ESA'14)*, September 8–10, 2014, Wroclaw, Poland, Vol. 8737 of Lecture Notes in Computer Science, Springer, pp. 137–148. This work was partially supported by the Research Grant PRIN 2010 "ARS TechnoMedia", funded by the Italian Ministry of Education, University, and Research.

✉ Stefano Leucci
stefano.leucci@inf.ethz.ch

Davide Bilò
davide.bilo@uniss.it

Luciano Gualà
guala@mat.uniroma2.it

Guido Proietti
guido.proietti@univaq.it

1 Dipartimento di Scienze Umanistiche e Sociali, Università di Sassari, Sassari, Italy

2 Dipartimento di Ingegneria dell'Impresa, Università di Roma "Tor Vergata", Rome, Italy

3 Department of Computer Science, ETH Zürich, Zürich, Switzerland

4 Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica, Università degli Studi dell'Aquila, L'Aquila, Italy

5 Istituto di Analisi dei Sistemi ed Informatica, CNR, Rome, Italy

we design two structures of size $O\left(\frac{n \log n}{\varepsilon^2}\right)$ which guarantee $(1+\varepsilon)$-stretched paths from the selected source also in the presence of an edge/vertex failure. This favorably compares with the currently best known solutions, which are for the edge-failure case of size $O(n)$ and stretch factor 3, and for the vertex-failure case of size $O(n \log n)$ and stretch factor 3. Moreover, we also focus on the unweighted case, and we prove that an ordinary *spanner* can be slightly augmented in order to build efficient fault-tolerant approximate *breadth-first-search trees*.

Keywords Shortest-path trees · Fault-tolerant structures · Approximate distances

1 Introduction

Broadcasting a message from a source node to every other node of a network is one of the most basic communication primitives. Since this operation should be performed by making use of a both sparse and fast infrastructure, the natural solution is to root at the source node a *shortest-path tree* (SPT) of the underlying graph. However, the SPT, as any tree-based network topology, is highly sensitive to a link/node malfunctioning, which will unavoidably cause the disconnection of a subset of nodes from the source.

To be readily prepared to react to any possible (transient) failure in an SPT, one has then to enrich the tree by adding to it a set of edges selected from the underlying graph, in order to obtain a subgraph that approximately preserves the distance from the source vertex even when a single component (i.e., edge or vertex) fails. More formally, if s denotes a distinguished source vertex of an undirected graph $G = (V, E)$ with non-negative real weights on its edges, and H is a spanning subgraph of G, then the *stretch* of $v \in V$ in H w.r.t. to G (and s, that will be omitted) is the ratio between the distance from s to v in H and in G. Then, for $\alpha \geq 1$, we say that a spanning subgraph H of G is an *Edge-fault-tolerant α-Approximate SPT of G w.r.t. s* (in short, α-EASPT), if for each edge $e \in E$, each vertex $v \in V$ has stretch at most α in $H - e = (V(H), E(H) \setminus \{e\})$ w.r.t. $G - e$. When *vertex failures* are considered, then the EASPT is correspondingly called VASPT. Ideally we would like an E/VASPT to have both a low stretch α and a small *size*, measured as the number $|E(H)|$ of edges in H. The case in which $\alpha = 1$ corresponds to requiring all the post-failure distances in $H - e$ to match the distances in $G - e$, i.e., H must contain an SPT (rooted at s) of $G - e$ for every $e \in E$. However, in this case, it is easy to see that $\Omega(n^2)$ edges might be required, as shown in Fig. 1.[1]

The aim of this paper is to show that, as soon as we allow for approximate distances, we can obtain an almost optimal stretch-size tradeoff for E/VASPTs.

1.1 Related Work

A problem that is very closely related to the design of an E/VASPT is that of computing a *single-source distance sensitivity oracle* (SDSO). Designing an efficient SDSO means

[1] Actually, a related but significantly more involved construction was originally provided in [16], in order to show a similar lower bound for on the space of any data structure for reporting exact shortest paths from a source vertex upon any vertex/edge failure.

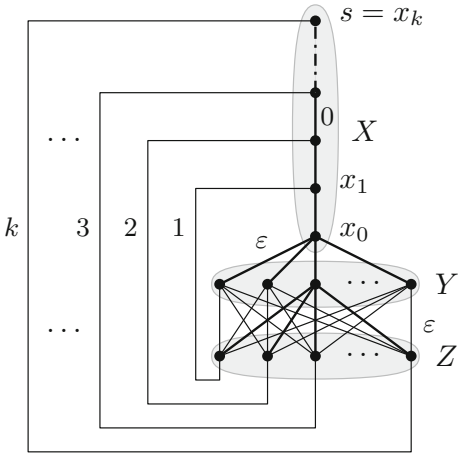

Fig. 1 A weighted graph G on n vertices and $\Theta(n^2)$ edges such that an edge-fault-tolerant SPT H of G is G itself. The vertices are partitioned into three sets $X = \{x_0, \ldots, x_k = s\}$, $Y = \{y_1, \ldots, x_k\}$, and $Z = \{z_1, \ldots, z_k\}$. Vertices in X are connected by a path whose edges have weight 0, there is a star centered in x_0 whose leaves are the vertices in Y, and the sets Y and Z induce a complete bipartite graph. All edge weights of the star and the complete bipartite graph are equal to ε with $0 < \varepsilon < \frac{1}{2}$, while, for $0 < i \leq k$, there is an edge of cost i between x_i and z_i. A SPT of G from s is shown with bold edges. If any edge $e = (x_i, x_{i+1})$ fails the new SPT of $G - e$ must include all the edges connecting z_i to the vertices in Y

to compute, with a *low* preprocessing time, a *compact* data structure which is able to *quickly* return a (possibly *approximate*) distance between a source vertex s and any other vertex of the graph, following a component failure. Notice that any E/VASPT H also implies the existence of a (trivial) SDSO having the same size, the same stretch, and a query time of $O(|E(H)| + n \log n)$: this SDSO is obtained by storing the whole graph H and by running Dijkstra's algorithm from s on the surviving graph to answer queries.

In [5] the authors compute in $O(m \log n + n^2 \log n)$ time a SDSO of size $O(n \log n)$, which reports, in constant time per query, 3-stretched distances following the failure of a single vertex. Such an oracle is also *path-reporting*, i.e., it is able to return the path associated with a distance query by paying an additional time which is proportional to the number of edges it contains. A closer inspection of this result shows that this SDSO is actually obtained through the computation of a 3-VASPT of size $O(n \log n)$. Regarding single edge failures, in [21] are (implicitly) provided (i) a path-reporting SDSO having stretch 3, size $O(n)$, and constant query time, and (ii) a corresponding 3-EABFS[2] containing $2(n-1)$ edges. Very recently, in [8], the authors show how to build a (non path-reporting) SDSO having stretch $1 + \varepsilon$, size $O(n\delta)$ and query time $O(\delta \log n)$, where $\delta = \varepsilon^{-1} \log \varepsilon^{-1}$, which can be improved to $O(1)$ for the special case $\varepsilon = 1$.

If we focus on unweighted graphs and we insist on preserving exact distances (i.e., stretch equal to 1) then, in [26], the authors provide a 1-E/VABFS of size $O(n \cdot$

[2] We use the notation E/VABFS instead of E/VASPT to stress the fact that we are dealing with unweighted graphs.

$\min\{ecc(s), \sqrt{n}\})$, where $ecc(s)$ denotes the eccentricity of the source s in G. In the same paper, the authors also exhibit a corresponding lower bound of $\Omega(n^{3/2})$ for the size of such a structure (in fact, the construction provided in Fig. 1 is obtained by elaborating such lower bound). In [5] the authors focus on the vertex-failure case and, for any $\varepsilon > 0$, they compute in $O(m\sqrt{n/\varepsilon})$ time a path-reporting SDSO of size $O(\frac{n}{\varepsilon^3} + n \log n)$, stretch $(1 + \varepsilon)$, and having constant query time. Once again, this SDSO is obtained through the construction of a $(1+\varepsilon)$-VABFS of size $O(\frac{n}{\varepsilon^3} + n \log n)$. Actually, we point out that the latter structure can be easily sparsified so as to obtain, for any $\varepsilon > 0$, a $(1 + \varepsilon)$-EABFS of size $O(\frac{n}{\varepsilon^3})$: indeed, its $O(n \log n)$-size term is associated with an auxiliary substructure that, for the case of edge failures, can be made of linear size. This result is of independent interest, since it qualifies itself as the best current solution for the EABFS problem. In [27] the authors present, among other results, a 3-EABFS having at most $4n$ edges. Interestingly, this was the first *explicit* construction for the problem, but two (better) implicit solutions were already available in the literature: the first one is the just mentioned structure which can be derived from the results presented in [5], while the second one is the 3-EASPT of size at most $2n$ (and then, *a fortiori*, a 3-EABFS of the same size) of [21] that can be easily obtained as a by-product of the results given therein (we will discuss this point in more detail later).

1.2 Our Results

Our main result is a polynomial time construction[3] of a $(1 + \varepsilon)$-E/VASPT of size $O(\frac{n \log n}{\varepsilon^2})$, for any $\varepsilon > 0$. These two structures substantially improve the stretch of the 3-EASPT of linear size implicitly given in [21], and that of the 3-VASPT of size $O(n \log n)$ given in [5], respectively, while essentially using the same number of edges (up to a logarithmic factor in the former case). To obtain our results, we perform a careful selection of edges that will be added to an initial SPT. The somewhat surprising outcome of our approach is that if we accept to have slightly stretched fault-tolerant paths, then we can drastically reduce the $\Theta(n^2)$ size of the structure that we would have to pay for having fault-tolerant *shortest* paths! Actually, the analysis of the stretch factor and of the structures' size induced by our algorithms is quite involved. Thus, for clarity of presentation, we give our result in two steps: first, we show an approach to build a $(1 + \varepsilon)$-EASPT of size $O(\frac{n \log n}{\varepsilon^2})$, then we outline how this approach can be extended to the vertex-failure case.

We also focus on the unweighted case, and we exhibit an interesting connection between a fault-tolerant approximate BFS and an (α, β)-*spanner*. An (α, β)-spanner of a graph G is a spanning subgraph H of G such that *all* the node-to-node distances in H are stretched by at most a multiplicative factor of α plus an additive term of β w.r.t. the corresponding distances in G. If such a condition holds even after an edge/vertex is deleted from both G and H, then H is an *edge/vertex-fault-tolerant* (α, β)-*spanner*. Moreover, if the guarantee on the stretch only holds for distances

[3] We do not insist on the time efficiency in building our structures, since the focus of our paper, consistently with the literature, is on the trade-off between their size and their stretch factor.

from vertices in a subset $S \subseteq V$, then the spanner is said to be *sourcewise*. We show how a (α, β)-spanner of size σ can be used to build in polynomial time a sourcewise edge-fault-tolerant (resp. vertex-fault-tolerant) (α, β)-spanner of size $O(\sigma + |S| \cdot n)$ (resp., $O(\sigma + |S| \cdot n \log n)$). This result has three main consequences. First of all notice that when $|S| = 1$, a sourcewise edge/vertex-fault-tolerant $(\alpha, 0)$-spanner is exactly an α-E/VABFS. As a consequence, for relevant values of α and β (e.g., when they are constant) the E/VABFS problem is easier than the corresponding (non fault-tolerant) spanner problem, and we regard this as an interesting hardness characterization.[4] A second consequence, is that this bridge between the two problems allows to build the sparsest $(1, \beta)$-VABFS structures known so far, by making use of the vast literature on additive $(1, \beta)$-spanners. More precisely, the $(1, 4)$-spanner of size $\widetilde{O}(n^{\frac{7}{5}})$ given in [11], and the $(1, 6)$-spanner of size $O(n^{\frac{4}{3}})$ given in [4], can be used to build corresponding VABFS structures. As a last consequence of our result, we are able to: (i) sparsify, for $|S| = \widetilde{\omega}(n^{\frac{1}{15}})$, the sourcewise edge-fault-tolerant $(1, 4)$-spanner of size $O(|S| \cdot n^{\frac{4}{3}})$ given in [27] by reducing its size to $\widetilde{O}(n^{\frac{7}{5}} + |S|n)$; and (ii) reduce the stretch of the sourcewise vertex-fault-tolerant $(1, 8)$-spanner of size $\widetilde{O}(n^{\frac{4}{3}})$ given in [23] to $(1, 6)$, for $|S| = \widetilde{O}(n^{\frac{1}{3}})$ (see Sect. 6 for the exact bounds of the obtained spanners).

1.3 Other Related Results

Additive EABFS *structures* In addition to the already cited results, in [27] the authors also consider (α, β)-EABFS, i.e., edge-fault-tolerant structures for which the length of a path is stretched by at most a factor of α plus an additive term of β. In particular, they prove that $(1, 3)$-EABFS structures admit a lower bound of $\Omega(n^{5/4})$ edges, thus showing an interesting dichotomy between multiplicative and additive stretches, i.e., the fact that additive stretches require super-linear size. Moreover, they construct a $(1, 4)$-EABFS of size $O(n^{4/3})$.

Sourcewise E/VABFS *structures* In [26], the same authors extend the already cited 1-E/VABFS of size $O(n^{\frac{3}{2}})$ to the sourcewise case, i.e., that in which the structure incorporates an edge-fault-tolerant BFS rooted at each vertex of a set $S \subseteq V$. Here, they show the existence of a solution of size $O(\sqrt{|S|} \cdot n^{3/2})$, which is tight. Moreover, they also consider the optimization problem of constructing a minimum-size sourcewise 1-E/VABFS, and they provide a corresponding tight $O(\log n)$-approximation algorithm.

Multiple edge failures Regarding multiple edge failures, Parter in [24] presented a 2-edge-fault-tolerant *exact* BFS having $O(n^{5/3})$ edges, which is tight, while in [27] it is shown the existence of a $(3(f + 1), (f + 1) \log n)$-EABFS of size $O(fn)$ for any number $f = O(1)$ of failed edges. This latter result has been improved in [9] where the authors prove the existence of a $(2|F| + 1)$-EASPT of size $O(fn)$ which tolerates the failure of any set F of edges of size at most f. This structure can be converted into a corresponding SDSO having the same size, and with query time $O(|F|^2 \log^2 n)$.

[4] For constant values of α and β, the size of an (α, β)-spanner is $\omega(n \log n)$ and hence the additive terms in the size of our E/VABFS are dominated by σ.

Moreover, if one is willing to use $O(m \log^2 n)$ space, such an oracle is also able to handle any number of edge failures (i.e., up to m). In [15], the special case of *shortest-path failures* was considered, where the set of failing edges F is supposed to form a source-leaf subpath in a given SPT of G. In particular, for the case $|F| = 2$, they give an SDSO achieving stretch 3, size $O(n \log n)$, and constant query time.

Directed graphs For single-source distances on directed graphs with integer positive edge weights bounded by M, in [20] it is shown how to build efficiently in $\widetilde{O}(Mn^{\omega})$ time, where $\omega < 2.373$ denotes the matrix multiplication exponent, a randomized edge-fault-tolerant SDSO of size $\Theta(n^2)$ returning in $O(1)$ time distances from the source which are exact w.h.p.

Fault-tolerant spanners Another setting which is very close in spirit to ours is that of *fault-tolerant spanners*. In [12], for weighted graphs and any integer $k \geq 1$, the authors present a $(2k-1, 0)$-spanner resilient to f vertex (resp., edge) failures of size $O(f^2 k^{f+1} n^{1+1/k} \log^{1-1/k} n)$ (resp., $O(f n^{1+1/k})$). This was later improved through a randomized construction in [17]. For a comparison, the sparsest known $(2k-1)$-multiplicative ordinary spanner has size $O(n^{1+1/k})$ [1], and this is believed to be asymptotically tight due to the long-standing girth conjecture of Erdős [19]. Finally, we mention that in [3] it was introduced the resembling concept of *resilient spanners*, i.e., spanners such that whenever any edge in G fails, then the relative distance increases in the spanner are very close to those in G, and it was shown how to build a resilient spanner by augmenting an ordinary spanner.

Concerning unweighted graphs, it makes instead sense to study fault-tolerant additive spanners. In particular, Braunshvig et al. [10] proposed the following general approach to build an additive spanner tolerating up to f edge failures: Let A be an f-edge-fault-tolerant $(\alpha, 0)$-spanner, and let B be an ordinary $(1, \beta)$-spanner. Then $H = A \cup B$ is an f–edge-fault-tolerant $(1, 2f(2\beta + \alpha - 1) + \beta)$-spanner. Recently, in [7] the corresponding analysis has been refined yielding a better additive bound of $2f(\beta + \alpha - 1) + \beta$, and, more in general, improved fault-tolerant additive spanners have been presented. Also very close to our present work are the (non-fault-tolerant) *sourcewise spanners* (which, again, approximately preserves all distances from a given set $S \subseteq V$ of sources). In that respect, in [14] the authors give, for any $k > 1$, a structure with additive stretch $2k$ and size $O(n^{1+1/(2k+1)}(k|S|)^{k/(2k+1)})$, which in particular for $k = \log n$ returns a structure with additive stretch $2 \log n$ and size $O(n\sqrt{|S| \log n})$. To the best of our knowledge, no results are instead known for the weighted case.

Further related works For recent achievements on all-to-all distance sensitivity oracles, we refer the reader to [5,6,13,18], while for other results on single-edge/vertex failures spanners/oracles on unweighted graphs, we finally refer the reader to [2,5,23,25].

1.4 Paper Organization

The paper is organized as follows: in Sect. 2 we introduce the notation that will be used throughout the paper; in Sect. 3 we revisit one of the swap procedures presented in [21] to formally prove that it can be used to build a simple 3-EASPT; in Sects. 4 and 5 we present our main results, namely a $(1+\varepsilon)$-EASPT and a $(1+\varepsilon)$-VASPT,

respectively; in Sect. 6 we focus on unweighted graphs, and we show the connection between an E/VABFS and an (α, β)-spanner; finally, in Sect. 7 we conclude the paper by outlining few directions for future research.

2 Preliminaries and Notation

We start by introducing our notation. For the sake of brevity, we give it for the case of edge failures, but it can be naturally extended to the node failure case.

Let $G = (V, E)$ be a non-negatively real weighted, undirected input graph. Given a source vertex $s \in V$, we denote by T a fixed SPT of G rooted at s. Let $w(e)$ or $w(u, v)$ denote the weight of an edge $e = (u, v) \in E$. For a subgraph $H = (V(H), E(H))$ of G, say $H \subseteq G$, we denote by $w(H) = \sum_{e \in E(H)} w(e)$. Moreover, given an edge $e = (u, v)$, we denote by $H - e$ or $H - (u, v)$ (resp., $H + e$ or $H + (u, v)$) the graph obtained from H by removing (resp., adding) the edge e. Similarly, for a set F of edges, $H + F$ will denote the graph obtained from H by adding the edges in F.

Given an edge $e = (u, v) \in E(T)$, we denote by $U(e)$ and $D(e)$ the partition of V induced by the two connected components of $T - e$, such that $U(e)$ contains s and u, and $D(e)$ contains v. Then, $C(e) = \{(x, y) \in E : x \in U(e), y \in D(e)\}$ will denote the *cutset* of e, i.e., the set of edges crossing the cut $(U(e), D(e))$.

We will denote by $\pi_H(x, y)$ a shortest path in H between two vertices $x, y \in V$, and by $d_H(x, y)$ its (weighted) length. Whenever the source vertex s is an endpoint of a shortest path in H, say towards another vertex u, we will simply write $\pi_H(u)$ and $d_H(u)$ instead of $\pi_H(s, u)$ and $d_H(s, u)$. Given an edge $e \in E$, we define $\pi_H^{-e}(x, y)$, $d_H^{-e}(x, y)$ and T^{-e} to be, respectively, a shortest path between x and y in $H - e$, its length, and an SPT of $G - e$ rooted at s. For the sake of simplifying the notation, shortest paths and distances in G will be denoted by omitting the subscript. When considering an edge (u, v) of T, we will assume u to be closer to s than v. Moreover, if P is a path from x to y and Q is a path from y to z, with $x, y, z \in V$, we will denote by $P \circ Q$ the path from x to z obtained by concatenating P and Q.

Throughout the rest of the paper we will assume that, when multiple shortest paths exist, ties will be broken in a consistent manner. More precisely, we assume the following:

1. T^{-e} is computed using a deterministic algorithm, i.e., multiple runs of the same algorithm to compute an SPT of $G - e$ always return the same SPT;
2. $\pi^{-e}(t)$ denotes the shortest path from s to t in $G - e$ which is also contained in T^{-e};
3. if x is a node of $\pi^{-e}(t)$, then $\pi^{-e}(x, t)$ is a subpath of $\pi^{-e}(t)$;
4. T^{-e} is computed giving higher priority to edges in T, i.e., if $\pi^{-e}(t)$ contains two distinct vertices x and y such that one of them is an ancestor of the other w.r.t. T, then the unique shortest path in T from x to y is a subpath of $\pi^{-e}(t)$.

2.1 Simplifying Assumptions on G

For the sake of avoiding technicalities, we assume that the edge weights in the input graph G are strictly positive. Indeed, our results easily extend to non-negative weights

as explained in the following, and thus all our statements are given in the more general form. Assume w.l.o.g. that the smallest non-zero weight of an edge is 1 (this can be guaranteed by multiplying all the weights by the inverse of the smallest non-zero weight), and that $\varepsilon < 2$ (as otherwise the already existing 3-E/VASPT constructions of [21] and [5] could be used instead). We define a new graph G' in which every edge of weight 0 in G has weight $\frac{\varepsilon}{2(1+\frac{\varepsilon}{2})n}$, while the other edge weights remain unaltered. Then, the edges of any $(1+\frac{\varepsilon}{2})$-E/VASPT H' of G' (w.r.t. s) induce a $(1+\varepsilon)$-E/VASPT H of G (w.r.t. s). Indeed, for every $t \in V$ and $e \in E$, we have:

$$\begin{aligned} d_H^{-e}(t) \leq d_{H'}^{-e}(t) &\leq \left(1+\frac{\varepsilon}{2}\right) d_{G'}^{-e}(t) < (1+\frac{\varepsilon}{2}) \left(d^{-e}(t) + \frac{n\varepsilon}{2(1+\frac{\varepsilon}{2})n}\right) \\ &= (1+\varepsilon)d^{-e}(t) + \left(1+\frac{\varepsilon}{2}\right)\frac{n\varepsilon}{2(1+\frac{\varepsilon}{2})n} - \frac{\varepsilon}{2}d^{-e}(t) \\ &= (1+\varepsilon)d^{-e}(t) + \frac{\varepsilon}{2}(1-d^{-e}(t)) < (1+\varepsilon)d^{-e}(t) + (1-d^{-e}(t)). \end{aligned}$$

If $d^{-e}(t) \geq 1$ then the second term is non-positive and therefore $d_H^{-e}(t) < (1+\varepsilon)d^{-e}(t)$. Otherwise, $d^{-e}(t)$ must be 0 and we have $d_H^{-e}(t) < 1$ which implies $d_H^{-e} = 0$.

We also assume, w.l.o.g., that the input graph G is 2-edge/vertex-connected, to avoid pathological failures that would disconnect the graph (this can be guaranteed by adding to G a new vertex that is connected to all the other vertices with edges of large weight).

3 A 3-EASPT Structure with at Most $2n-2$ Edges

We here provide a revisitation of one of the swap procedures presented in [21] to formally prove that it can be used to build a simple 3-EASPT with at most $2n-2$ edges, on which our construction of the $(1+\varepsilon)$-EASPT will rely. More precisely, in [21] the authors were concerned with the problem of reconnecting in a best possible way (w.r.t. a set of distance criteria) the two subtrees of an SPT undergoing an edge failure, through a careful selection of a *swap edge*, i.e., an edge with an endvertex in each of the two subtrees. In particular, they show that if we select as a swap edge for $e = (u, v)$ – with u closer to the source s than v – the edge that lies on a shortest path in $G - e$ from s to v, then the distances from the source towards all the disconnected vertices is stretched at most by a factor of 3.[5] Therefore, a 3-EASPT of size at most $2n-2$ can be obtained by simply adding to an SPT rooted at s such a swap edge for each corresponding tree edge, and interestingly this improves the 3-EASPT of size at most $4n$ provided in [27].

More formally, Algorithm 1 builds a 3-EASPT H_0 as follows: initially H_0 is an SPT of G then, for each possible failure of an edge $e = (u, v)$ in T, we augment H_0 by adding the (unique) edge (x, y) of $C(e)$ that lies on a shortest path $\pi^{-e}(v)$ from s

[5] Actually, in [21] it is not explicitly claimed the 3-stretch factor, but this is implicitly obtained by the qualitative analysis of the swap procedure therein provided.

Algorithm 1: Algorithm for building a 3-EASPT

Input : A non-negatively real weighted graph G, $s \in V$
Output: A 3-EASPT H_0 of G rooted at s

```
H_0 ← T
for e = (u, v) ∈ E(T) do
    f_e ← the single edge in C(e) ∩ π^{-e}(v)
    H_0 ← H_0 + f_e
return H_0
```

to v. Notice that this is the only edge of $\pi^{-e}(v)$ that is not already in T as both $\pi_T(x)$ and $\pi_T(v, y)$ do not contain e.

Lemma 1 *Algorithm 1 computes in polynomial time a 3-EASPT structure of size at most* $2n - 2$.

Proof The claim on the size of H_0 is a direct consequence of the fact that we add at most one swap edge for each failure, so we only need to prove that H_0 is a 3-EASPT structure. As H_0 contains all the edges of T the condition $d_{H_0}(u) = d(u)\ \forall u \in V$ is clearly true. Moreover, the above still holds whenever an edge $e \notin E(T)$ fails.

Now, let $e = (u, v) \in E(T)$ be the failed edge and let t be any vertex in V. If t belongs to $U(e)$ then H_0 contains the whole shortest path $\pi^{-e}(t) = \pi(t)$. Otherwise, H_0 contains both the path $\pi^{-e}(v)$ and the path $\pi^{-e}(v, t) = \pi(v, t)$ so we can write:

$$d^{-e}_{H_0}(t) \leq d^{-e}(v) + d(v, t) \leq d^{-e}(t) + 2d(v, t) \leq d^{-e}(t) + 2d(t) \leq 3d^{-e}(t).$$

□

4 A $(1+\varepsilon)$-EASPT Structure

First, we give a high-level description of our algorithm for computing a $(1+\varepsilon)$-EASPT (see Algorithm 2).

We build our structure H by starting from the 3-EASPT of size $O(n)$ returned by Algorithm 1. Then, our algorithm works in $n-1$ *phases*, where each phase considers the failure of an edge of T w.r.t. a fixed preorder visit of the edges, say $e_1, \ldots, e_{n-1}$. Let e_h be the edge of T of the h-th phase of the algorithm. The algorithm checks all the vertices in $D(e)$ and, whenever a vertex t *is bad for* e_h, i.e., $d^{-e_h}_H(t) > (1+\varepsilon)d^{-e_h}(t)$, it chooses a suitable value $\eta \geq 1$ and adds to H all the last η edges of $\pi^{-e_h}(t)$ that are missing in H. As we will see, the choice of η guarantees not only that the stretch of t in the augmented structure w.r.t. $G - e_h$ is within $1+\varepsilon$, but it also guarantees a substantial improvement on the stretch factors of all the nodes in $\pi^{-e_h}(t)$ in which the added edges were entering. More precisely, as a first ingredient, we will show that if the algorithm adds m_h edges to our structure during phase h, then the total decrease on the stretch factors of these nodes (once measured w.r.t. $G - e_h$) is at least $\Omega(\frac{\varepsilon m_h}{\log n})$.

However, this is not enough to get to our final result, since when passing to the next phase, i.e., when considering the removal of e_{h+1}, the current stretches now measured

w.r.t $G - e_{h+1}$ might change completely. Thus, our main difficulty will lie in putting all the phases together in the analysis. To this aim, we will make use of a second ingredient, namely a suitable non-negative monotonically decreasing *potential function* Φ, which essentially measures the overall stretches w.r.t. G, and so it is *independent* of the current failing edge. Such Φ will be such that its initial value is $O(n/\varepsilon)$, and it will be shown to decrease at each phase of a quantity $\Omega(\frac{\varepsilon\, m_h}{\log n})$. As a consequence, at the end of the last phase, we will obtain that the total number of added edges is $O(\frac{n \log n}{\varepsilon^2})$.

The main result we are going to prove in this section is then the following:

Theorem 1 *Given an n-vertex non-negatively real weighted graph $G = (V, E)$, a source vertex $s \in V$, and any $\varepsilon > 0$, the structure H returned in polynomial time by Algorithm 2 is a $(1+\varepsilon)$-EASPT of G w.r.t. s of size $O(\frac{n \log n}{\varepsilon^2})$.*

We will prove separately the bound on the stretch factor and on the size of the structure in the next two subsections (see Lemmas 3 and 10, respectively).

4.1 Stretch Factor of the Structure

To prove the correctness of our algorithm, we first show that for each bad vertex, say t, associated with a failing edge, say e, the algorithm adds a set F of η edges of $\pi^{-e}(t)$ to the current structure so as the augmented structure minus e contains a path from s to t which is $1+\varepsilon$ stretched w.r.t. $G - e$. We now describe how both η and F are chosen.

Let us denote by $\widetilde{H}$ the structure built by the algorithm just before t is considered. Let $f = (x, y)$ with $x \in U(e)$ be the unique edge (recall indeed that edge weights are positive) in $C(e) \cap E(\pi^{-e}(t))$. Consider the subpath of $\pi^{-e}(t)$ going from x to t and let $X = \{x_0 = x, x_1 = y, \ldots, x_r = t\}$ be its vertices, in order of appearance in $\pi^{-e}(x, t)$. Then, let $X' = \{x_i \in X : (x_{i-1}, x_i) \notin E(\widetilde{H})\}$. We rename the vertices of X' as $z_1, \ldots, z_k$, according to their order, and we let $z_0 = x$ (see Fig. 2). For $i = 0, \ldots k$, let $\alpha_i = \frac{d^{-e}_{\widetilde{H}}(z_i)}{d^{-e}(z_i)}$, i.e., the stretch of z_i in $\widetilde{H} - e$ w.r.t. $G - e$. Observe that $\alpha_0 = 1$.

Think of the edges in $\pi^{-e}(t)$ as being directed towards t for a moment. We now describe how the set F of edges added by the algorithm, with $|F| = \eta$, is selected. For $i = 0, \ldots, k$, consider the sequence $\gamma_i = 1 + \frac{\varepsilon}{\mathcal{H}_k}(\mathcal{H}_k - \mathcal{H}_{k-i})$, where $\mathcal{H}_n$ denotes the n-th *harmonic number*. Notice that the sequence is monotonically increasing from $\gamma_0 = 1$ to $\gamma_k = 1 + \varepsilon$. Let $0 \le j < k$ be the largest index such that $\alpha_j \le \gamma_j$ (see Fig. 3). Notice that j always exists as $\alpha_0 = \gamma_0$. Then, we add the set F of the edges entering into the last $\eta = k - j$ vertices in X' to $\widetilde{H}$.

This is enough to prove the correctness of our algorithm as the following lemma shows.

Lemma 2 *Let e be any edge of T and let t be any bad vertex for e. Let $\widetilde{H}$ be the structure built by the algorithm just before t is considered during the visit of e, and, finally, let j be the maximum index such that $\alpha_j \le \gamma_j$. Then, for every vertex t' of $\pi^{-e}(z_j, t)$, we have that $d^{-e}_{\widetilde{H}+F}(t') \le d^{-e}_{\widetilde{H}}(z_j) + d^{-e}(z_j, t') \le \alpha_j\, d^{-e}(t')$.*

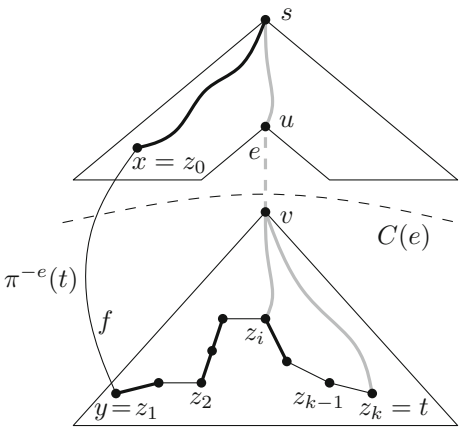

Fig. 2 Edge selection phase of Algorithm 2 when a bad vertex t for the failing edge e is considered. Bold edges belong to $\widetilde{H}$ while the black path is $\pi^{-e}(t)$. The shortest paths from s to z_i and t in G are shown in gray

Proof Observe that $\pi^{-e}(z_j, t)$ is a subpath of $\pi^{-e}(t)$. Therefore

$$d^{-e}_{\widetilde{H}+F}(t') \leq d^{-e}_{\widetilde{H}}(z_j) + d^{-e}(z_j, t') \leq \alpha_j\, d^{-e}(z_j) + d^{-e}(z_j, t')$$
$$\leq \alpha_j \big(d^{-e}(z_j) + d^{-e}(z_j, t')\big) = \alpha_j\, d^{-e}(t').$$

□

Since the structure H returned by the algorithm contains $\widetilde{H} + F$ and since $\alpha_j \leq \gamma_j \leq 1 + \varepsilon$, Lemma 2 immediately implies the following.

Lemma 3 *The structure H returned by Algorithm 2 is a $(1+\varepsilon)$-EASPT.*

4.2 Size of the Structure

Now we describe the edge selection process and we analyze the size of our final structure. To this aim, we first describe how a bad vertex is handled, then we show how this will be exploited within a single phase, and finally in the development of all the algorithm's phases.

4.2.1 Handling a Bad Vertex

Let us fix the failed edge $e = (u, v)$ and a single bad vertex t for e. We here show that the choice of F described above ensures that the overall decrease of the values α_i, once they are computed w.r.t. $\widetilde{H} + F$, will be at least $\frac{\varepsilon}{\mathcal{H}_n}\eta$. As we will see later, this fact will be instrumental to show the final bound on our structure.

Let $Z(t) = \{z_{j+1}, \ldots, z_k\}$ be the set of vertices for which an incoming edge has been added in F, that we call the *special vertices* associated with t. For every vertex $z \in Z(t)$, we define in $\widetilde{H} + F - e$ the following path $P(z) := \pi^{-e}_{\widetilde{H}}(z_j) \circ \pi^{-e}(z_j, z)$

Algorithm 2: Algorithm for building an $(1+\varepsilon)$-EASPT

Input : A non-negatively real weighted graph G, $s \in V$, $\varepsilon > 0$
Output: A $(1+\varepsilon)$-EASPT of G rooted at s

$H \leftarrow H_0$.
for $e \in E(T)$ *in preorder w.r.t.* T **do**
  for *every* $t \in D(e)$ **do**
    if $d_H^{-e}(t) > (1+\varepsilon)d^{-e}(t)$ **then** // vertex t is *bad* for edge e
      // Select a suitable set of edges $F \subseteq E(\pi^{-e}(t))$
      Let $f = (x, y)$ with $x \in U(e)$ be the unique edge in $C(e) \cap E(\pi^{-e}(t))$,
      Let $\pi^{-e}(x, t) = \langle x_0, x_1, x_2, \ldots \rangle$.
      Let $X = \{x_i : (x_{i-1}, x_i) \notin E(H)\}$.
      Let $z_0 = x$ and let $X' = \{z_1, \ldots, z_k\}$ be the vertices of X, in order of appearance in $\pi^{-e}(x, t)$.
      Let $\alpha_i = \frac{d_H^{-e}(z_i)}{d^{-e}(z_i)}$ for $i = 0, \ldots, k$.
      $j \leftarrow \max\{j \mid \alpha_j \leq 1 + \frac{\varepsilon}{\mathcal{H}_k}(\mathcal{H}_k - \mathcal{H}_{k-i})\}$
      // F contains the edges of $\pi^{-e}(x, t)$ entering the last $k - j = \eta$ vertices in X'
      $F \leftarrow \{(x_{i-1}, x_i) \mid x_i \in \{z_{j+1}, \ldots, z_k\}\}$
      // Add the edges in F to the current subgraph
      $H \leftarrow H + F$
return H

(notice indeed that by construction $\pi^{-e}(z_j, z)$ is entirely contained in $\widetilde{H} + F - e$). Moreover, we define $\alpha'_i = \frac{w(P(z_i))}{d^{-e}(z_i)}$, and note that α'_i is an upper bound to the stretch of z_i in $\widetilde{H} + F - e$ w.r.t. $G - e$. The following holds:

Lemma 4 *For every* $i = j+1, \ldots, k$, $\alpha'_i \leq \alpha_j < \alpha_i$.

Proof Indeed, by definition of j, we have $\alpha_j \leq \gamma_j < \gamma_i < \alpha_i$. Moreover, since z_i is a vertex of $\pi^{-e}(z_j, t)$, from Lemma 2 we have that

$$\alpha'_i = \frac{w(P(z_i))}{d^{-e}(z_i)} = \frac{d_{\widetilde{H}}^{-e}(z_j) + d^{-e}(z_j, z_i)}{d^{-e}(z_i)} \leq \alpha_j. \qquad \square$$

Next lemma provides a lower-bound on the overall decrease on the stretch factors of nodes in $Z(t)$, once $\widetilde{H}$ is enriched with new added edges in F (see also Fig. 3):

Lemma 5 $\sum_{z \in Z(t)} \left(\frac{d_{\widetilde{H}}^{-e}(z)}{d^{-e}(z)} - \frac{w(P(z))}{d^{-e}(z)} \right) \geq \frac{\varepsilon}{\mathcal{H}_n}\eta$.

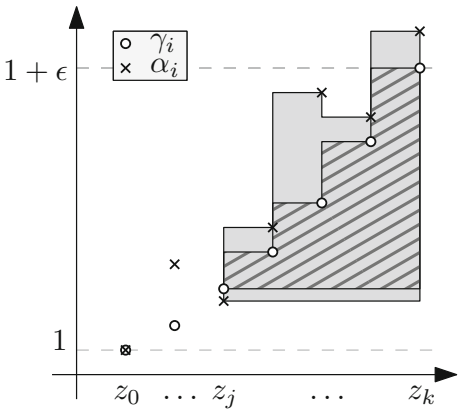

Fig. 3 Representation of the sequences α_i and γ_i. The gray area is a lower bound to the overall decrease of the α'_i values w.r.t. α_i with $i > j$. This area is, in turn, lower bounded by the area of the striped region which is $\frac{\varepsilon}{\mathcal{H}_k}(k-j)$

Proof Indeed (notice that the first inequality follows from Lemma 4):

$$\sum_{z\in Z(t)}\left(\frac{d^{-e}_{\widetilde{H}}(z)}{d^{-e}(z)}-\frac{w(P(z))}{d^{-e}(z)}\right)=\sum_{i=j+1}^{k}(\alpha_i-\alpha'_i)\geq\sum_{i=j+1}^{k}(\alpha_i-\alpha_j)\geq\sum_{i=j+1}^{k}(\gamma_i-\gamma_j)$$

$$=\frac{\varepsilon}{\mathcal{H}_k}\sum_{i=j+1}^{k}\left(\mathcal{H}_{k-j}-\mathcal{H}_{k-i}\right)=\frac{\varepsilon}{\mathcal{H}_k}\sum_{i=j+1}^{k}\left(\sum_{\ell=1}^{k-j}\frac{1}{\ell}-\sum_{\ell=1}^{k-i}\frac{1}{\ell}\right)$$

$$=\frac{\varepsilon}{\mathcal{H}_k}\sum_{i=j+1}^{k}\left(\sum_{\ell=k-i+1}^{k-j}\frac{1}{\ell}\right)=\frac{\varepsilon}{\mathcal{H}_k}\sum_{\ell=1}^{k-j}\frac{|\{i\mid k-\ell+1\leq i\leq k\}|}{\ell}$$

$$=\frac{\varepsilon}{\mathcal{H}_k}\sum_{\ell=1}^{k-j}1=\frac{\varepsilon}{\mathcal{H}_k}(k-j)=\frac{\varepsilon}{\mathcal{H}_n}\eta.$$

□

4.2.2 Analysis of a Single Phase

In each phase, the above selection procedure is repeated by the algorithm for every bad vertex that should arise. To exploit how our careful selection of additional edges for a single bad vertex will impact on the final size of our structure, we now provide a lower-bound on the overall decrease on the stretch factors of all the special nodes of a phase.

Let us then focus on the h-th phase of the algorithm, when edge e_h is removed. We call B_h the set of all the bad vertices considered in this phase, and moreover, let $Z_h = \cup_{t\in B_h} Z(t)$ (notice that $B_h \subseteq Z_h$).

It is worth noting that for every $t, t' \in B_h$, with $t \neq t'$, we have $Z(t) \cap Z(t') = \emptyset$, i.e., sets $Z(\cdot)$ are pairwise disjoint. Indeed, once $z \in Z(t)$, we add the edge of $\pi^{-e_h}(t)$ entering z, say (z', z), and so when any further bad vertex t' is taken into consideration during phase h, and it happens that the corresponding path $\pi^{-e_h}(x, t')$ passes through

z, then it will clearly uses edge (z', z), and so z cannot be special also for t'. Hence, we let $P_h(z)$ be the unique path $P(z)$ which is built during phase h. Finally, recall that H_0 is the initial 3-EASPT structure, and let H'_h, $\widetilde{H}(t)$, and H_h be the structures currently built by the algorithm at the beginning of phase h, just before the bad vertex $t \in B_h$ is processed, and at the end of phase h, respectively. The following corollary follows from Lemma 4.

Corollary 1 *For every $z \in Z_h$, $w(P_h(z)) \leq d^{-e_h}_{H'_h}(z)$.*

Proof Let t be a bad vertex during phase h, and let $\widetilde{H}$ be the structure right before t is visited within phase h. Let $\alpha = \frac{d^{-e_h}_{\widetilde{H}}(z)}{d^{-e_h}(z)}$ and $\alpha' = \frac{w(P(z))}{d^{-e_h}(z)}$. Then, we have that $w(P_h(z)) = w(P(z)) = \alpha' d^{-e_h}(z) < \alpha d^{-e_h}(z) = d^{-e_h}_{\widetilde{H}}(z) \leq d^{-e_h}_{H'_h}(z)$. □

Let m_h be the number of new edges added during the phase h. Next lemma, which shows that the total decrease on the stretch factors of the nodes Z_h is at least $\Omega(\frac{\varepsilon m_h}{\log n})$, is the first ingredient for bounding the size of H.

Lemma 6 $$\sum_{z \in Z_h} \left(\frac{d^{-e_h}_{H'_h}(z)}{d^{-e_h}(z)} - \frac{w(P_h(z))}{d^{-e_h}(z)} \right) \geq m_h \frac{\varepsilon}{\mathcal{H}_n}.$$

Proof For a bad vertex $t \in B_h$, let η_t be the number of edges selected by the algorithm when t is considered (i.e., $|F|$ according to the notation we used when we focused on a single bad vertex). In the following chain of (in)equalities we use, in this order, the fact that (i) sets $Z(t)$ are pairwise disjoint, (ii) every $\widetilde{H}(\cdot)$ is a supergraph of H'_h, and (iii) Lemma 5.

$$\sum_{z \in Z_h} \left(\frac{d^{-e_h}_{H'_h}(z)}{d^{-e_h}(z)} - \frac{w(P_h(z))}{d^{-e_h}(z)} \right) = \sum_{t \in B_h} \sum_{z \in Z(t)} \left(\frac{d^{-e_h}_{H'_h}(z)}{d^{-e_h}(z)} - \frac{w(P_h(z))}{d^{-e_h}(z)} \right)$$
$$\geq \sum_{t \in B_h} \sum_{z \in Z(t)} \left(\frac{d^{-e_h}_{\widetilde{H}(t)}(z)}{d^{-e_h}(z)} - \frac{w(P_h(z))}{d^{-e_h}(z)} \right) \geq \sum_{t \in B_h} \eta_t \frac{\varepsilon}{\mathcal{H}_n} \geq m_h \frac{\varepsilon}{\mathcal{H}_n}.$$

□

4.2.3 Putting all the Phases Together

In order to conclude our size analysis, we now consider the whole sequence of phases executed by our algorithm. Let us define recursively a function $\phi_h(z)$, for every $h = 0, \ldots, n-1$ and every $z \in V$: first we set $\phi_0(z) = \frac{6}{\varepsilon} d(z)$, and then we set

$$\phi_h(z) = \begin{cases} w(P_h(z)) & \text{if } z \in Z_h; \\ \phi_{h-1}(z) & \text{if } z \notin Z_h. \end{cases}$$

Now, let $\mathcal{S}_h = \bigcup_{i=1}^{h} Z_i$ (notice that $\mathcal{S}_0 = \emptyset$), namely the set of vertices that has been special in one or more phases up to the end of phase h. In the following, we first show that if $z \in \mathcal{S}_h$, then $\phi_{h-1}(z)$ is an upper bound to $d_{H'_h}^{-e_h}(z)$ (recall that H'_h is the structure at the beginning of phase h). This will be the key property needed to prove our bound on the size of H.

In order to show the first mentioned property of $\phi_h(z)$, we separately consider the cases $z \in Z_h \backslash \mathcal{S}_{h-1}$ and $z \in \mathcal{S}_{h-1}$ in the following two lemmas:

Lemma 7 *For every $z \in Z_h \backslash \mathcal{S}_{h-1}$ we have $\phi_{h-1}(z) \geq d_{H'_h}^{-e_h}(z)$.*

Proof Since $z \in Z_h$ we know that an incoming edge to z has been selected when the algorithm was considering some bad vertex t for the edge $e_h = (u, v)$. Recalling that $H_0 - e_h \subseteq H'_h - e_h$ contains by construction $\pi^{-e_h}(v)$ and $\pi(v, t)$, we have:

$$\begin{aligned} d^{-e_h}(v) + d(v,t) &\geq d_{H'_h}^{-e_h}(t) > (1+\varepsilon) d^{-e_h}(t) = (1+\varepsilon)(d^{-e_h}(z) + d(z,t)) \\ &\geq (1+\varepsilon)(d^{-e_h}(z) + |d(z) - d(t)|). \end{aligned} \tag{1}$$

Moreover, we also have:

$$d^{-e_h}(v) + d(v,t) \leq d^{-e_h}(z) + d(z,v) + d(v,t) \leq d^{-e_h}(z) + d(z) + d(t)$$

The above inequalities together imply:

$$d^{-e_h}(z) < \frac{d(z) + d(t) - (1+\varepsilon)\,|d(z) - d(t)|}{\varepsilon}.$$

If $d(z) \geq d(t)$, the above formula becomes:

$$d^{-e_h}(z) < \frac{d(z) + d(t) - (1+\varepsilon)(d(z) - d(t))}{\varepsilon} = \frac{2d(t) + \varepsilon(d(t) - d(z))}{\varepsilon} \leq \frac{2d(z)}{\varepsilon}.$$

Otherwise, $d(z) < d(t)$ and we have:

$$d^{-e_h}(z) < \frac{d(z) + d(t) - (1+\varepsilon)(d(t) - d(z))}{\varepsilon} = \frac{2d(z) + \varepsilon(d(z) - d(t))}{\varepsilon} \leq \frac{2d(z)}{\varepsilon}.$$

By hypothesis $z \notin \mathcal{S}_{h-1}$ (i.e., z does not belong to any set $Z_{h'}$ with $h' < h$), therefore $\phi_{h-1}(z) = \phi_0(z) = \frac{6}{\varepsilon} d(z)$. Since H'_h is a supergraph of H_0, which is a 3-EASPT, we immediately have:

$$d_{H'_h}^{-e_h}(z) \leq d_{H_0}^{-e_h}(z) \leq 3 d^{-e_h}(z) < \frac{6}{\varepsilon} d(z) = \phi_0(z) = \phi_{h-1}(z). \tag{2}$$

□

We now consider the remaining case:

Lemma 8 *For $z \in \mathcal{S}_{h-1}$, $\phi_{h-1}(z) \geq d_{H'_h}^{-e_h}(z)$.*

Proof For the sake of consistency with our notation, we prove the following equivalent claim: for any node $z \in Z_h$, the weight of the path $P_h(z)$ built by the algorithm when $e_h = (u, v)$ fails is an upper bound to $d_{H'_p}^{-e_p}(z)$, for any $p > h$. In particular, we argue that $P_h(z)$ is vertex disjoint (and then edge disjoint) from $\pi(v, z)$ (except for z). As a consequence, when e_p fails, either (i) e_p is not in $P_h(z)$, and so $P_h(z)$ is still all in $H'_p - e_p$, and then clearly $d_{H'_p}^{-e_p}(z) \leq w(P_h(z))$, or (ii) e_p is not in $\pi(v, z)$, and since e_p cannot belong to $\pi(s, v)$ (as failing edges are considered in preorder), we have $d_{H'_p}^{-e_p}(z) = d(z) \leq w(P_h(z))$.

Let $\widetilde{H}$ be the structure constructed by the algorithm just before $P_h(z)$ is built, t be the corresponding bad vertex, and z_j be the vertex chosen as described in Sect. 4.2.1. Recall that $z = z_i$ for some $i > j$, and that $P_h(z) = \pi_{\widetilde{H}}^{-e_h}(z_j) \circ \pi(z_j, z)$. Suppose by contradiction that $P_h(z)$ and $\pi(v, z)$ intersect at some vertex $q \neq z$. Clearly $q \in D(e_h)$, and we have two cases:

1. If $q \in V(\pi(z_j, z))$ we have that the subpath π' of $\pi(v, z)$ going from q to z is a shortest path in T that only traverses vertices in $D(e_h)$ and hence, by our tie-breaking rule, the subpath of $\pi^{-e_h}(t)$ going from q to z coincides with π'. It follows that all the vertices in $V(\pi(q, z))\backslash\{q\}$ cannot belong to Z_h. This implies $q = z$, a contradiction.
2. Otherwise $q \in V(\pi_{\widetilde{H}}^{-e_h}(z_j))$. In this case, $d_{\widetilde{H}}^{-e_h}(z_j) = d_{\widetilde{H}}^{-e_h}(q) + d_{\widetilde{H}}^{-e_h}(q, z_j)$. Moreover, z_j precedes z in $\pi^{-e_h}(t)$ and hence $d^{-e_h}(z) = d^{-e_h}(z_j) + d^{-e_h}(z_j, z)$. Therefore (notice that the first inequality follows from the triangle inequality and from the fact that $q \in \pi(v, z)$, and so $\pi(q, z)$ is in $T - e_h$ and then in $\widetilde{H} - e_h$):

$$\alpha_i = \frac{d_{\widetilde{H}}^{-e_h}(z)}{d^{-e_h}(z)} \leq \frac{d_{\widetilde{H}}^{-e_h}(q) + d(q, z)}{d^{-e_h}(z)} \leq \frac{d_{\widetilde{H}}^{-e_h}(q) + d^{-e_h}(q, z_j) + d^{-e_h}(z_j, z)}{d^{-e_h}(z)} =$$
$$\frac{d_{\widetilde{H}}^{-e_h}(z_j) + d^{-e_h}(z_j, z)}{d^{-e_h}(z_j) + d^{-e_h}(z_j, z)} \leq \max\left\{\frac{d_{\widetilde{H}}^{-e_h}(z_j)}{d^{-e_h}(z_j)}, \frac{d^{-e_h}(z_j, z)}{d^{-e_h}(z_j, z)}\right\} = \max\{\alpha_j, 1\} = \alpha_j$$

where we used the inequality $\frac{a+b}{c+d} \leq \max\left\{\frac{a}{c}, \frac{b}{d}\right\}$ $\forall a, b, c, d > 0$, and the fact that $\alpha_j \geq 1$. This is impossible as it contradicts Lemma 4. □

To summarize, by combining Lemmas 7 and 8 together, we immediately have:

Corollary 2 *If $z \in \mathcal{S}_h$, then $\phi_{h-1}(z) \geq d_{H'_h}^{-e_h}(z)$.*

We are now ready to prove the bound on the size of our structure. To this aim, as the second ingredient for bounding the size of H, we define a non-increasing global *potential function* $\Phi(h)$, $h = 0, \ldots, n-1$:

$$\Phi(h) = \sum_{z \in V\backslash\{s\}} \frac{\phi_h(z)}{d(z)}.$$

In other words, $\Phi(h)$ maps a phase h to an upper bound of the overall stretch of $H_h - e_h$ w.r.t. $G - e_h$: indeed, $\phi_h(z) \geq d_{H_h}^{-e_h}(z)$ and $d^{-e_h}(z) \leq d(z)$. Observe that, by definition of $\phi_0(z)$, we have $\Phi(0) = \sum_{z \in V \setminus \{s\}} \frac{\phi_0(z)}{d(z)} \leq \frac{6}{\varepsilon} n$.

Next lemma shows that the decrease of Φ after each phase, say h, of the algorithm is at least equal to the lower bound that we gave in Lemma 6 on the overall decrease on the stretch factors (as measured in $H_h - h$ w.r.t. $G - e_h$) of all the special nodes of that phase. Since this bound depends on the number of edges added in that phase, we will then be able to put into relationship the final size of our structure with the global decreasing of Φ.

Lemma 9 $\Phi(h-1) - \Phi(h) \geq m_h \frac{\varepsilon}{\mathcal{H}_n}$.

Proof Using the definitions we have:

$$\begin{aligned}
\Phi(h-1) - \Phi(h) &= \sum_{z \in V \setminus \{s\}} \frac{\phi_{h-1}(z)}{d(z)} - \sum_{z \in V \setminus \{s\}} \frac{\phi_h(z)}{d(z)} \\
&= \sum_{z \in Z_h} \frac{\phi_{h-1}(z) - \phi_h(z)}{d(z)} + \sum_{z \in V \setminus (Z_h \cup \{s\})} \frac{\phi_{h-1}(z) - \phi_h(z)}{d(z)} \\
&= \sum_{z \in Z_h} \frac{\phi_{h-1}(z) - \phi_h(z)}{d(z)} \geq \sum_{z \in Z_h} \left(\frac{d_{H'_h}^{-e_h}(z) - w(P_h(z))}{d(z)} \right)
\end{aligned}$$

where the latter equality follows from the fact that $\phi_h(z) = \phi_{h-1}(z)$ whenever $z \notin Z_h \cup \{s\}$, while last inequality follows from Corollary 2 and by definition of $\phi_h(z)$. Since Corollary 1 implies that every term at the numerator is non-negative, using Lemma 6 we obtain

$$\sum_{z \in Z_h} \left(\frac{d_{H'_h}^{-e_h}(z) - w(P_h(z))}{d(z)} \right) \geq \sum_{z \in Z_h} \left(\frac{d_{H'_h}^{-e_h}(z)}{d^{-e}(z)} - \frac{w(P_h(z))}{d^{-e}(z)} \right) \geq \frac{\varepsilon}{\mathcal{H}_n} m_h.$$

□

We are finally able to prove the following:

Lemma 10 *The size of the structure H returned by Algorithm 2 is $O\left(\frac{n \log n}{\varepsilon^2}\right)$.*

Proof Since H_0 contains $O(n)$ edges, we only focus on bounding the number $\mu = \sum_{h=1}^{n-1} m_h$ of edges in $E(H) \setminus E(H_0)$. Recall that $\Phi(0) \leq \frac{6}{\varepsilon} n$. Moreover, as every $\phi_h(z)$ is non-negative, $\Phi(n-1) \geq 0$ holds. Using these inequalities together with Lemma 9, we can write:

$$\frac{6}{\varepsilon} n \geq \Phi(0) - \Phi(n-1) = \sum_{h=1}^{n-1} (\Phi(h-1) - \Phi(h)) \geq \frac{\varepsilon}{\mathcal{H}_n} \sum_{h=1}^{n-1} m_h = \frac{\varepsilon}{\mathcal{H}_n} \mu$$

which can be solved for μ to get $\mu = O(\frac{n \log n}{\varepsilon^2})$. □

Algorithm 3: Algorithm for building a 3-VASPT.

Input : A non-negatively real weighted graph G, $s \in V$, $\varepsilon > 0$
Output: A 3-VASPT H_0 of G rooted at s

$H_0 \leftarrow T$
Compute a decomposition of T into a set $\mathcal{Q}$ of ancestor-leaf vertex-disjoint paths $Q_1, Q_2, \ldots$ as described in [5].
for $Q \in \mathcal{Q}$ **do**
  for $u \in V(Q)$ *such that* u *is not a leaf in* T **do**
    Let v be the vertex following u in Q.
    $U \leftarrow$ vertices of the unique connected component of $T - u$ containing s.
    $D \leftarrow$ vertices of the unique connected component of $T - u$ containing v.
    $O \leftarrow$ vertices of the connected components of $T - u$ not containing s, v.
    $T' \leftarrow$ SPT of $G - u$ with edges directed away from s.
    $E(H_0) \leftarrow E(H_0) \cup \{(x, y) \in E(T') \ : \ y \notin D\}$
    $E(H_0) \leftarrow E(H_0) \cup \{(x, y) \in E(\pi^{-u}(v)) : \ (x \in U \cup O) \wedge y \in D\}$
return H_0

5 A $(1+\varepsilon)$-VASPT Structure

In this section we extend our previous $(1+\varepsilon)$-EASPT structure to deal with vertex failures. In this case, when a vertex u is removed, the tree $T - u$ breaks into several subtrees (not just in 2, as for the edge failure case), and so a different approach is needed. Informally, by following the ideas developed in [5], we decompose T into several subpaths, and we let fail all the vertices along each subpath. Then, whenever each such vertex fails, say u, we handle the bad vertices that will arise only in the tree of the forest $T - u$ that contains the subsequent vertices of the subpath. As we will see, by using the path-decomposition of T developed in [5], we will be able to provide the sought bound on the size of our structure.

Let us then see how our method works. First of all, we build a different initial subgraph H_0 (see Algorithm 3), which is a 3-VASPT. The construction of H_0 is similar to that given by Baswana and Khanna [5] for the related problem of computing a vertex-fault-tolerant SDSO which reports (post-failure) 3-approximate distances from s. In particular, the key difference between their construction and ours is pointed out within the proof of the forthcoming Lemma 12, and such a difference is instrumental to guarantee the correctness of our approach. In the following, we first describe the construction of our structure H_0, and then we argue on how the analysis for the edge-failure case can be adjusted to show the same bound on the size of H for the vertex failure case as well.

Initially, H_0 is equal to T. Then, proceeding as proposed in [5], T is decomposed into ancestor-leaf vertex-disjoint paths in the following recursive way: select a path Q from the root of T to a leaf such that the removal of Q splits the tree into a forest where the size of each subtree is at most half the size of the original tree, and then proceed recursively on each subtree. After this preliminary path-decomposition step of T, for each generated path an approximate structure is built. This structure will provide approximate distances towards the vertices $V \backslash \{u\}$ whenever any vertex u along the path fails. The union of T with all these structures will form H_0.

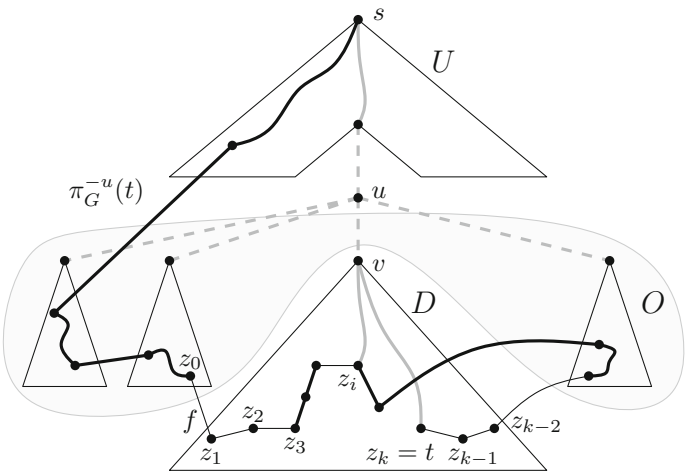

Fig. 4 Edge selection phase of the vertex-version of Algorithm 2 when a bad vertex t for the failing vertex u is considered. Bold edges belong to $\widetilde{H}$ while the black path is $\pi^{-u}(t)$. The shortest paths from v to z_i and t in G are shown in gray. Notice that all the special vertices z_i's belong to the down set D

Let us then describe how to build the initial structure for a fixed path Q of the previous decomposition. Let q be the starting (i.e., closest to s) vertex of Q, and let T_q be the subtree of T rooted at q. Moreover, let $u \in V(Q)$ be a failing vertex, and let v be the next vertex in Q.[6] Similarly to what is done in [5,22], we partition the vertices of the forest $T - u$ into three sets: (i) the *up set* U, containing all the vertices of the tree of $T - u$ rooted at s, (ii) the *down set* D, containing all the vertices of the tree of $T - u$ rooted at v, and (iii) the *others set* O, containing all the remaining vertices (see Fig. 4).

In order to select the set of additional edges associated with Q, we construct an SPT T' of $G - u$ rooted at s, and we imagine that its edges are directed towards the leaves. We select all the edges of $E(T')\backslash E(T)$ that do not lead to a vertex in D, plus the unique edge of $\pi^{-u}(v)$ that crosses the cut C induced by the sets $U \cup O$ and D.[7] Notice that $T - u$ contains all the paths in T' towards the vertices in U, and that each vertex has at most one incoming edge in T'. This implies that the number of selected edges is at most $|O| + 1$.

The above procedure is repeated for all the failing vertices of Q, in order. As the sets O associated with the different vertices are disjoint we have that, while processing Q, at most $|V(T_q)| + |Q| = O(|V(T_q)|)$ edges are selected. Finally, the procedure is repeated for all the paths of the decomposition, and since such a decomposition is

[6] W.l.o.g. we are assuming that the failing vertex u is not a leaf, as otherwise $T - u$ is already an SPT of $G - u$.

[7] The uniqueness of such an edge follows from the suboptimality property of shortest paths, and from the fact that if (x, y) is the first edge in the considered cut C that is encountered along the path $\pi^{-u}(v)$, then $\pi^{-u}(v) = \pi^{-u}(x) \circ (x, y) \circ \pi^{-u}(y, v)$, where $\pi^{-u}(y, v) = \pi_T^{-u}(v, y)$ from our shortest-path tie-breaking rule.

done as suggested in [5], it immediately follows that the size of the entire structure H_0 is $O(n \log n)$.

We now prove some useful properties of the structure H_0. First of all, observe that, by construction and similarly to the edge-failure case, we immediately have:

Lemma 11 *Let u be a failed vertex. Then, we have that:* (i) *for the root v of the unique tree in $T - u$ that contains the vertices in D, $d^{-u}_{H_0}(v) = d^{-u}(v)$, and* (ii) *for any $z \in D$, $d^{-u}_{H_0}(z) \leq 3d^{-u}(z)$.*

Moreover, we also have the following:

Lemma 12 *Let H be a subgraph of G containing H_0 and let u be a failed vertex. If, for every $t \in D$, $d^{-u}_H(t) \leq (1+\varepsilon)d_{-u}(t)$, then $d^{-u}_H(t) \leq (1+\varepsilon)d_{-u}(t)$, for every $t \in V$.*

Proof Let $t \in V$. Clearly, if $t \in U \cup D$, then the claim trivially follows. Therefore, we assume that $t \in O$. Let t' be the last vertex of $\pi^{-u}(t)$ which is contained in D during a traversal of $\pi^{-u}(t)$ from s to t. Since H contains H_0 and, furthermore, since H_0 contains the shortest path from t' to t in $G - u$, we have that

$$d^{-u}_H(t) \leq d^{-u}_H(t') + d^{-u}(t', t) \leq (1+\varepsilon)d^{-u}(t') + d^{-u}(t', t) \leq (1+\varepsilon)d^{-u}(t).$$

□

Lemma 12 suggests the following simple modification of Algorithm 2. More precisely, when u is failing, not all the vertices which are disconnected from s in $T - u$ must be visited, but it is sufficient to visit all the vertices in D.

At this point, the same analysis given for the case of edge failures can be retraced for vertex failures as well. Indeed, all the special vertices for a bad vertex t are, by construction, in D as well (see Fig. 4), since in H_0 we were adding all the edges of a SPT of $G - u$ rooted at s not leading to a vertex in D, and so in particular we were adding all the edges of $\pi^{-u}(t)$ not leading to a vertex in D. Finally, we use Lemma 11 in the proof of Lemma 7, and more precisely statement (i) in the inequality of (1) and statement (ii) in the second inequality of Eq. (2). From this, it will be again possible to show through the use of the potential function Φ that the number of edges added to H_0 is $O(\frac{n \log n}{\varepsilon^2})$. Hence we have:

Theorem 2 *Given an n-vertex non-negatively real weighted graph $G = (V, E)$, a source vertex $s \in V$, and any $\varepsilon > 0$, the vertex-version of Algorithm 2 computes in polynomial time a $(1+\varepsilon)$-VASPT of G w.r.t. s of size $O(\frac{n \log n}{\varepsilon^2})$.*

6 Relation with (α, β)-Spanners in Unweighted Graphs

In this section we turn our attention to the unweighted case, and we provide two polynomial-time algorithms that augment an (α, β)-spanner of G so as to obtain an (α, β)-EABFS/VABFS. We first present the algorithm for the vertex-failure case, and then we show how it can be adapted to the edge-failure case.

The algorithm for computing an (α, β)-VABFS of G w.r.t. s, say H, works as follows: it first computes the structure H_0 as explained in Sect. 5, then augments it with a set of edges, and finally further augments it with the edges of the given (α, β)-spanner of G. The structure H_0 is initially augmented as follows. The vertices of the BFS of G rooted at s are visited one at a time. Let u be a vertex visited by the algorithm and let D be the set of vertices of the tree defined so as explained in Sect. 5 w.r.t the path decomposition computed for H_0. For every $t \in D$, the algorithm checks whether $\pi^{-u}(t)$ contains no vertex in $D\backslash\{t\}$ and $d^{-u}(t) < d^{-u}_{H_0}(t)$. If this is the case, then the algorithm augments H_0 with the edge of $\pi^{-u}(t)$ incident to t.

The following observation is crucial to prove the algorithm correctness.

Fact 1 *For every vertex u and every vertex $t \in V\backslash\{u\}$ such that $\pi^{-u}(t)$ contains a vertex in D, let x and y be the first and last vertex of $\pi^{-u}(t)$ that belong to D, respectively. We have $d^{-u}_{H_0}(x) = d^{-u}(x)$ and $d^{-u}_{H_0}(y, t) = d^{-u}(y, t)$.*

We can now give the following:

Theorem 3 *Given an n-vertex unweighted graph $G = (V, E)$, a source vertex $s \in V$, and an (α, β)-spanner of G of size σ, it is possible to compute in polynomial time an (α, β)-VABFS of G w.r.t. s of size $O(\sigma + n \log n)$.*

Proof Let H be the subgraph of G computed as described above. We first prove that H is an (α, β)-VABFS of G w.r.t. s by showing that $d^{-u}_H(t) \leq \alpha \cdot d^{-u}(t) + \beta$, for two distinct vertices $u, t \in V$. W.l.o.g., we can assume that $\pi^{-u}(t)$ contains some vertices in $D\backslash\{t\}$ because, if our assumption was not true, then, by Fact 1, $d^{-u}_H(t) = d^{-u}(t) \leq \alpha \cdot d^{-u}(t) + \beta$.

Let x and y be the first and last vertex of $\pi^{-u}(t)$ contained in $D\backslash\{t\}$ in a path traversal from s to t, respectively. We have that $\pi^{-u}(t) = \pi^{-u}(x) \circ \pi^{-u}(x, y) \circ \pi^{-u}(y, t)$, i.e.,

$$d^{-u}(t) = d^{-u}(x) + d^{-u}(x, y) + d^{-u}(y, t). \tag{3}$$

By Fact 1, H contains $\pi^{-u}(x)$ as well as $\pi^{-u}(y, t)$. Therefore,

$$d^{-u}_H(x) = d^{-u}(x) \quad \text{and} \quad d^{-u}_H(y, t) = d^{-u}(y, t). \tag{4}$$

We now prove that $d^{-u}_H(x, y) \leq \alpha \cdot d^{-u}(x, y) + \beta$. Since H contains an (α, β)-spanner of G, H contains a path P from x to y such that $w(P) \leq \alpha \cdot d(x, y) + \beta$. Clearly, if $u \notin V(P)$, then $H - u$ contains P and therefore $d^{-u}_H(x, y) \leq w(P) \leq \alpha \cdot d(x, y) + \beta$. Otherwise, if $u \in V(P)$, then let v be the least common ancestor of x and y in the BFS of G rooted at s. Since $v \in D$, it follows that

$$d^{-u}_H(x, y) \leq d_H(x, v) + d_H(v, y) < d_H(x, u) + d_H(u, y) \leq w(P) \leq \alpha \cdot d(x, y) + \beta.$$

Using the last inequality together with Eqs. (3) and (4), we have that

$$\begin{aligned} d^{-u}_H(t) &\leq d^{-u}_H(x) + d^{-u}_H(x, y) + d^{-u}_H(y, t) \\ &\leq d^{-u}(x) + \alpha \cdot d(x, y) + \beta + d^{-u}(y, t) \leq \alpha \cdot d^{-u}(t) + \beta. \end{aligned}$$

We now prove that the size of H is $O\big(\sigma + n \log n\big)$ by showing that the size of H_0 is $O(n \log n)$. We have already shown in the previous section that the number of edges of H_0 before the algorithm augments it is $O(n \log n)$. Therefore, it remains to bound the number of edges initially added to H_0. Let F be the set of such edges. Our argument is similar to the one used in [27] to bound the size of a 3-EABFS. Namely, we prove that $|F| \leq 3n$ by showing that each vertex t caused the addition of at most 3 edges to F. Let t be a fixed vertex. Let $u_0, \ldots, u_\ell$ be the vertices of the path $\pi(t)$, in a traversal of the path from s to t whose failures caused the insertion of the edge (v_i, t) of $\pi^{-u_i}(t)$ incident to t in F. Since G is unweighted, $d(t) = d(v_i) + j$, where $j \in \{-1, 0, 1\}$. Furthermore, for every vertex $u' \neq t$ which is a proper descendant of u_0 in the BFS tree of G rooted at s, $H - u'$ contains the path $\pi(v_0) \circ \pi(v_0, t)$ of length at most $d(t) + 1 + 1 = d(t) + 2$. Finally, observe that for every $1 \leq i \leq \ell$ and for every vertex $u' \neq t$ which is a descendant of u_i in the BFS tree of G rooted at s, $H - u'$ contains the path $\pi^{-u_i}(t)$. Therefore, for every $2 \leq i \leq \ell$, we have that

$$d(t) \leq d^{-u_i}(t) \leq d(t) + 2 - i.$$

The above inequality implies that $\ell \leq 2$. Hence each vertex t caused the addition of at most $\ell + 1 \leq 3$ edges to F. □

Now, we adapt the algorithm to prove a similar result for the (α, β)-EABFS. The algorithm first augments a BFS tree T of G rooted at s and then adds its edges to the (α, β)-spanner of G. The tree T is augmented by visiting its edges. Let e be the edge visited by the algorithm. For every $t \in D(e)$, the algorithm checks whether $\pi^{-e}(t)$ contains no vertex in $D(e)\backslash\{t\}$ and $d^{-e}(t) < d_T^{-e}(t)$. If this is the case, then the algorithm augments T with the edge of $\pi^{-e}(t)$ incident to t. It is easy to see that the proof of Theorem 3 can be adapted to prove the following:

Theorem 4 *Given an n-vertex unweighted graph $G = (V, E)$, a source vertex $s \in V$, and an (α, β)-spanner of G of size σ, it is possible to compute in polynomial time an (α, β)-EABFS of G w.r.t. s of size at most $\sigma + 3n$.*

Notice that the obtained (α, β)-E/VABFS structures can be easily adapted to the multisource case, by simply rooting at each given source vertex $s \in S$ an augmented BFS. This will immediately provide corresponding (α, β)-stretched sourcewise edge/vertex-fault-tolerant spanners (SES/SVS) of size $O\big(\sigma + |S| \cdot n\big)$ and $O\big(\sigma + |S| \cdot n \log n\big)$, respectively.

Interestingly, this immediately allows to improve some existing contructions. Indeed, by using the $(1, 4)$-spanner of size $\widetilde{O}(n^{\frac{7}{5}})$ given in [11] we obtain the following result:

Corollary 3 *Given an unweighted graph $G = (V, E)$ with n vertices, and a set of source vertices $S \subseteq V$, it is possible to compute an $(1, 4)$-SES of G w.r.t. S having size $\widetilde{O}(n^{\frac{7}{5}} + |S| \cdot n)$ in polynomial time.*

This sparsifies the $(1, 4)$-SES of size $O(|S| \cdot n^{\frac{4}{3}})$ given in [27] as soon as $|S| = \widetilde{\omega}(n^{\frac{1}{15}})$.

Moreover, by using the $(1, 6)$-spanner of size $O(n^{4/3})$ provided in [4], we also have:

Corollary 4 *Given an unweighted graph $G = (V, E)$ with n vertices, and a set of source vertices $S \subseteq V$, it is possible to compute an $(1, 6)$-SVS of G w.r.t. S having size $O(n^{\frac{4}{3}} + |S| \cdot n \log n)$ in polynomial time.*

This improves the additive stretch of the $(1, 8)$-SVS of size $\widetilde{O}(n^{\frac{4}{3}})$ given in [23], which holds for $|S| = \widetilde{O}(n^{\frac{1}{3}})$.

7 Conclusions

In this paper, we have studied the problem of designing single-edge/vertex-fault-tolerant structures rooted at a source vertex, aiming at finding a compact set of edges of the input (either weighted or unweighted) graph that will provide approximate shortest paths from the source following the failure of an edge/vertex in the graph. The main contribution of our research is that we can get almost shortest paths with almost linear size, in sharp contrast with a corresponding true-shortest paths structure which may require a quadratic size. Another interesting contribution we provided is the bridging between (α, β)-spanners and (α, β)-E/VABFS.

The problem of designing good fault-tolerant approximate-shortest-path structures deserves further investigation. For the single-source case, we mention three intriguing problems: (1) designing a SDSO with stretch arbitrary close to 1, almost linear size and constant query time for both the single-edge and the single-vertex failure scenario. The closest result is the SDSO given in [8] that has a logarithmic query time (w.r.t. the number of vertices of the graph) and only works for single edge failures; (2) removing the log-factor from the size of our structure, either improving its analysis or by further sparsifying it; (3) studying the multiple vertex-failure case. To the best of our knowledge there are no non-trivial VASPTs or SDSOs for this case. Other future directions involve the study of the multisource case (i.e., a sourcewise fault-tolerant spanner), with the goal of designing a structure which only adds a sublinear (in the number of sources) term to the size of our single-source structure. Moreover, we also plan to investigate the existence of efficient fault-tolerant structures for other notable network topologies, like the minimum spanning tree, the tree spanner, or the minimum-routing cost spanning tree.

Acknowledgements We wish to thank the anonymous reviewers for their insightful comments and for their helpful suggestions for improving the paper.

References

1. Althöfer, I., Das, G., Dobkin, D.P., Joseph, D., Soares, J.: On sparse spanners of weighted graphs. Discrete Comput. Geom. **9**, 81–100 (1993)
2. Ausiello, G., Ribichini, A., Franciosa, P.G., Italiano, G.F.: Computing graph spanners in small memory: fault-tolerance and streaming. Discrete Math. Algorithms Appl. **2**(4), 591–606 (2010)
3. Ausiello, G., Franciosa, P.G., Italiano, G.F., Ribichini, A.: On Resilient Graph Spanners. In: Proceedings of the 21st European Symposium on Algorithms (ESA'13). Lecture Notes in Computer Science, vol. 8125, pp. 85–96. Springer (2013)
4. Baswana, S., Kavitha, T., Mehlhorn, K., Pettie, S.: Additive spanners and (α, β)-spanners. ACM Trans. Algorithms **7**, A.5 (2010)

5. Baswana, S., Khanna, N.: Approximate shortest paths avoiding a failed vertex: near optimal data structures for undirected unweighted graphs. Algorithmica **66**(1), 18–50 (2013)
6. Bernstein, A., Karger, D.R.: A nearly optimal oracle for avoiding failed vertices and edges. In: Proceedings of the 41st Symposium on the Theory of Computing (STOC'09), pp. 101–110. ACM Press (2009)
7. Bilò, D., Grandoni, F., Gualà, L., Leucci, S., Proietti, G.: Improved purely additive fault-tolerant spanners. In: Proceedings of the 23rd European Symposium on Algorithms (ESA'15). Lecture Notes in Computer Science, vol. 9294, pp. 167–178. Springer (2015)
8. Bilò, D., Gualà, L., Leucci, S., Proietti, G.: Compact and fast sensitivity oracles for single-source distances. In: Proceedings of the 24th Annual European Symposium on Algorithms (ESA'16). Leibniz International Proceedings in Informatics (LIPIcs), vol. 57, pp. 13:1–13:14 (2016)
9. Bilò, D., Gualà, L., Leucci, S., Proietti, G.: Multiple-edge-fault-tolerant approximate shortest-path trees. In: Proceedings of the 33rd Symposium on Theoretical Aspects of Computer Science (STACS'16). Leibniz International Proceedings in Informatics (LIPIcs), vol. 47, pp. 18:1–18:14 (2016)
10. Braunschvig, G., Chechik, S., Peleg, D.: Fault tolerant additive spanners. In: Proceedings of the 38th Workshop on Graph-Theoretic Concepts in Computer Science (WG'12). Lecture Notes in Computer Science, vol. 7551, pp. 206–214. Springer (2012)
11. Chechik, S..: New additive spanners. In: Proceedings of the 24th Symposium on Discrete Algorithms (SODA'13), pp. 498–512. ACM Press (2013)
12. Chechik, S., Langberg, M., Peleg, D., Roditty, L.: Fault-tolerant spanners for general graphs. In: Proceedings of the 41st Symposium on the Theory of Computing (STOC'09), pp. 435–444. ACM press (2009)
13. Chechik, S., Langberg, M., Peleg, D., Roditty, L.: f-Sensitivity distance oracles and routing schemes. In: Proceedings of the 18th European Symposium on Algorithms (ESA'10). Lecture Notes in Computer Science, vol. 6942, pp. 84–96. Springer (2010)
14. Cygan, M., Grandoni, F., Kavitha, T.: On pairwise spanners. In: Proceedings of the Symposium on Theoretical Aspects of Computer Science (STACS'13). Leibniz International Proceedings in Informatics (LIPIcs), vol. 20, pp. 209–220 (2013)
15. D'Andrea, A., D'Emidio, M., Frigioni, D., Leucci, S., Proietti, G.: Path-fault-tolerant approximate shortest-path trees. In: Proceedings of the 22nd International Colloquium on Structural Information and Communication Complexity (SIROCCO'15). Lecture Notes in Computer Science, vol. 9439, pp. 224–238. Springer (2015)
16. Demetrescu, C., Thorup, M., Chowdhury, R.A., Ramachandran, V.: Oracles for distances avoiding a failed node or link. SIAM J. Comput. **37**(5), 1299–1318 (2008)
17. Dinitz, M., Krauthgamer, R.: Fault-tolerant spanners: better and simpler. In: Proceedings of the 30th Symposium on Principles of Distributed Computing (PODC'11), pp. 169–178. ACM Press (2011)
18. Duan, R., Pettie, S.: Dual-failure distance and connectivity oracles. In: Proceedings of the 20th Symposium on Discrete Algorithms (SODA'09), pp. 506–515. ACM Press (2009)
19. Erdős, P.: Extremal problems in graph theory. In: Proceedings of the Symposium on Theory of Graphs and its Applications, pp. 29–36 (1964)
20. Grandoni, F., Williams, V.Vassilevska: Improved distance sensitivity oracles via fast single-source replacement paths. In: Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS'12), pp. 748–757 (2012)
21. Nardelli, E., Proietti, G., Widmayer, P.: Swapping a failing edge of a single source shortest paths tree is good and fast. Algorithmica **36**(4), 361–374 (2003)
22. Nardelli, E., Proietti, G., Widmayer, P.: Finding the most vital node of a shortest path. Theor. Comput. Sci. **296**(1), 167–177 (2003)
23. Parter, M.: Vertex fault tolerant additive spanners. In: Proceedings of the 28th International Symposium on Distributed Computing (DISC'14). Lecture Notes in Computer Science, vol. 8784, pp. 167–181. Springer (2014)
24. Parter, M.: Dual failure resilient BFS structure. In: Proceedings of the 34th Symposium on Principles of Distributed Computing (PODC'15), pp. 481–490. ACM Press (2015)
25. Parter, M.: Fault-tolerant logical network structures. Bull. EATCS. **118**, 75–100 (2016)
26. Parter, M., Peleg, D.: Sparse fault-tolerant BFS trees. In: Proceedings of the 21st European Symposium on Algorithms (ESA'13). Lecture Notes in Computer Science, vol. 8125 , pp. 779–790. Springer (2013)
27. Parter, M., Peleg, D.: Fault tolerant approximate BFS structures. In: Proceedings of the 25th Symposium on Discrete Algorithms (SODA'14), pp. 1073–1092. ACM Press (2014)