Algorithmica (2016) 74:1363–1385
DOI 10.1007/s00453-015-0006-x

On Resilient Graph Spanners

Giorgio Ausiello[1] · Paolo G. Franciosa[2] · Giuseppe F. Italiano[3] · Andrea Ribichini[1]

Received: 30 May 2014 / Accepted: 23 April 2015 / Published online: 2 May 2015

Abstract We introduce and investigate a new notion of resilience in graph spanners. Let S be a spanner of a weighted graph G. Roughly speaking, we say that S is resilient if all its point-to-point distances are resilient to edge failures. Namely, whenever any edge in G fails, then as a consequence of this failure all distances do not degrade in S substantially more than in G (i.e., the relative distance increases in S are very close to those in the underlying graph G). In this paper we show that sparse resilient spanners exist, and that they can be computed efficiently.

Work partially supported by the Italian Ministry of Education, University, and Research (MIUR) under PRIN 2012C4E3KT national research project "AMANDA—Algorithmics for MAssive and Networked DAta". A preliminary version of this paper was presented at the 21st Annual European Symposium on Algorithms, LNCS 8125, pp. 85–96.

✉ Paolo G. Franciosa
paolo.franciosa@uniroma1.it

Giorgio Ausiello
ausiello@dis.uniroma1.it

Giuseppe F. Italiano
giuseppe.italiano@uniroma2.it

Andrea Ribichini
ribichini@dis.uniroma1.it

1 Dipartimento di Ingegneria Informatica, Automatica e Gestionale, Università di Roma "La Sapienza", via Ariosto 25, 00185 Rome, Italy

2 Dipartimento di Scienze Statistiche, Università di Roma "La Sapienza", piazzale Aldo Moro 5, 00185 Rome, Italy

3 Dipartimento di Ingegneria Civile e Ingegneria Informatica, Università di Roma "Tor Vergata", via del Politecnico 1, 00133 Rome, Italy

Keywords Graph algorithms · Graph spanners · Fault tolerance · Algorithm complexity

1 Introduction

Spanners are fundamental graph structures that have been extensively studied in the last decades since they were introduced in [24]. Given a graph G, a *spanner* is a (sparse) subgraph of G that preserves the approximate distance between each pair of vertices. More precisely, a t-spanner of a graph $G = (V, E)$ is a subgraph $S = (V, E_S)$, $E_S \subseteq E$, that distorts distances in G up to a multiplicative factor t: i.e., for all vertices x, y, $d_S(x, y) \leq t \cdot d_G(x, y)$, where d_G denotes the distance in graph G. We refer to t as the *stretch factor* (or *distortion*) of the spanner S. It is known how to compute in $O(m+n)$ time a $(2k-1)$-spanner, with $O(n^{1+\frac{1}{k}})$ edges [2,18] (which is conjectured to be optimal for any k), where m and n are respectively the number of edges and vertices in the original graph G. We note that t-spanners are only considered for $t \geq 3$, as 2-spanners can have as many as $\Theta(n^2)$ edges.

Several other spanners have been considered in the literature. For $\alpha \geq 1$ and $\beta \geq 0$, an (α, β)-spanner of an unweighted graph $G = (V, E)$ is a subgraph $S = (V, E_S)$, $E_S \subseteq E$, that distorts distances in G up to a multiplicative factor α and an additive term β: i.e., for all vertices x, y, $d_S(x, y) \leq \alpha \cdot d_G(x, y) + \beta$. In [7], it is shown how to compute a $(k, k-1)$-spanner containing $O(k \cdot n^{1+1/k})$ edges, for any integer $k \geq 2$. Note that t-spanners can be referenced to as $(t, 0)$-spanners, while $(1, \beta)$-spanners are also known as *purely additive spanners* ($d_S(x, y) \leq d_G(x, y) + \beta$). Algorithms for computing (1,2)-spanners with $O(n^{3/2})$ edges are given in [1,16,25], for (1,4)-spanners with $\tilde{O}(n^{7/5})$ edges in [11], and for (1,6)-spanners with $O(n^{4/3})$ edges in [7].

Spanners have been investigated also in the fully dynamic setting, where edges may be added to or deleted from the original graph. In [4], efficient dynamic deterministic algorithms are first presented for low-stretch spanners. A faster randomized dynamic algorithm for spanners has been later proposed by Baswana [6]: given an unweighted graph, a $(2k-1)$-spanner of expected size $O(k \cdot n^{1+1/k})$ can be maintained in $O\left(\frac{m}{n^{1+1/k}} \cdot \text{polylog } n\right)$ amortized expected time for each edge insertion/deletion, where m is the current number of edges in the graph. For $k = 2, 3$ (i.e., 3- and 5-spanners, respectively), the amortized expected time of the randomized algorithm becomes constant. The algorithm by Elkin [17] maintains a $(2k-1)$-spanner with expected $O(kn^{1+1/k})$ edges in expected constant time per edge insertion and expected $O\left(\frac{m}{n^{1/k}}\right)$ time per edge deletion. More recently, Baswana et al. [8] proposed two faster fully dynamic randomized algorithms for maintaining $(2k-1)$-spanners of unweighted graphs: the expected amortized time per insertion/deletion is $O(7^{k/2})$ for the first algorithm and $O(k^2 \log^2 n)$ for the second algorithm, and in both cases the spanner expected size is optimal up to a polylogaritmic factor.

As observed in [12], this traditional fully dynamic model may be too pessimistic in several application scenarios, where the possible changes to the underlying graph are rather limited. Indeed, there are cases where there can be only temporary network

failures: namely, graph edges may occasionally fail, but only for a short period of time, and it is possible to recover quickly from such failures. In those scenarios, rather than maintaining a fully dynamic spanner, which has to be updated after each change, one may be more interested in working with a static spanner capable of retaining many of its properties during edge deletions, i.e., capable of being resilient to transient failures.

Being inherently sparse, a spanner is not necessarily resilient to edge deletions and it may indeed lose some of its important properties during a transient failure. Indeed, let S be a t-spanner of G: if an edge e fails in G, then the distortion of the spanner may substantially degrade, i.e., $S\backslash e$ may no longer be a t-spanner or even a valid spanner of $G\backslash e$, where $G\backslash e$ denotes the graph obtained after removing edge e from G. In their pioneering work, Chechik et al. [12] addressed this problem by introducing the notion of *fault-tolerant spanners*, which are defined as follows. Given an integer $f \geq 1$, a spanner is said to be f-edge (resp., vertex) fault-tolerant if it preserves its original distortion under the failure of any set of at most f edges (resp., vertices). More formally, an *f-edge (resp. vertex) fault-tolerant t-spanner* of $G = (V, E)$ is a subgraph $S = (V, E_S)$, $E_S \subseteq E$, such that for any subset $F \subseteq E$ (resp. $F \subseteq V$), with $|F| \leq f$, and for any pair of vertices $x, y \in V$ (resp. $x, y \in V\backslash F$) we have $d_{S\backslash F}(x, y) \leq t \cdot d_{G\backslash F}(x, y)$, where $G\backslash F$ denotes the subgraph of G obtained after deleting the edges (resp. vertices) in F. Algorithms for computing efficiently fault-tolerant spanners can be found in [5,10,12,15].

The distortion is not the only property of a spanner that may degrade because of edge failures. Indeed, even when the removal of an edge cannot change the overall distortion of a spanner (such as in the case of a fault-tolerant spanner), it may still cause a sharp increase in some of its distances. Note that while the distortion is a *global* property, distance increases are *local* properties, as they are defined for pairs of vertices. To address this problem, one would like to work with spanners that are not only *globally resilient* (such as fault-tolerant spanners) but also *locally resilient*. In other terms, we would like to make the distances between any pair of vertices in a spanner resilient to edge failures, i.e., whenever an edge fails, then the increases in distances in the spanner must be very close to the increases in distances in the underlying graph. More formally, given a graph G and an edge e in G, we define the *fragility of edge e* as the maximum relative increase in distance between any two vertices when e is removed from G:

$$\mathrm{frag}_G(e) = \max_{x,y \in V} \left\{ \frac{d_{G\backslash e}(x, y)}{d_G(x, y)} \right\}$$

The fragility of edge e is a measure of how much e is crucial for the distances in G, as it provides an upper bound to the increase in distance in G between any pair of vertices when edge e fails: the higher the fragility of e, the higher is the relative increase in some distance when e is deleted. Note that the fragility of an edge in a subgraph cannot be smaller than its fragility in the original graph. Thus, the best we can expect for a spanner is that it preserves the same edge fragilities as in G. Unfortunately, as we will show in the sequel, fragility of edges is not always preserved in edge fault-tolerant spanners.

Our definition of fragility is somewhat reminiscent of the notions of *shortcut value* [21] and of *Vickrey pricing* [23] of an edge, where the distance increase is alternatively

measured by the difference, instead of the ratio, between distances in $G\backslash e$ and in G. Note that for unweighted graphs $\text{frag}_G(e) \geq 2$ for any edge e. Although, in our definition, the fragility of an edge measures the maximum global distance increase under the deletion of that edge, we show that it is equivalent to a local property, i.e., one only needs to guarantee a bounded distance increase for the endpoints of that edge.

1.1 Our contribution

To obtain spanners whose distances are resilient to transient edge failures, the fragility of each edge in the spanner must be as close as possible to its fragility in the original graph. In this perspective, given a positive σ, we say that a spanner S of G is σ-*resilient* if $\text{frag}_S(e) \leq \max\{\sigma, \text{frag}_G(e)\}$ for each edge $e \in S$, where $\sigma \geq 1$. Note that in case of unweighted graphs, for $\sigma = 2$ this is equivalent to $\text{frag}_S(e) = \text{frag}_G(e)$ for any $e \in S$. We show that finding sparse 2-resilient spanners may be an overly ambitious goal, as we prove that there exists a family of dense graphs for which the only 2-resilient spanner coincides with the graph itself. It can be easily seen that, in general, spanners are not necessarily σ-resilient. Furthermore, it can be shown that even edge fault-tolerant multiplicative t-spanners are not σ-resilient, since they can only guarantee that the fragility of a spanner edge is at most t times its fragility in the graph. In fact, we exhibit 1-edge fault tolerant t-spanners, for any $t \geq 3$, with edges whose fragility in the spanner is at least $t^2/2$.

It seems quite natural to ask whether sparse σ-resilient spanners exist, and how efficiently they can be computed. We show that it is possible to compute σ-resilient 3-spanners containing $O(W \cdot n^{3/2})$ edges for graphs with positive edge weights in $[w_{\min}, w_{\max}]$, where $W = \frac{w_{\max}}{w_{\min}}$. The size is optimal for small edge weights. The total time required to compute our spanners is $O(mn + n^2 \log n)$ in the worst case. To compute our σ-resilient spanners, we start from a non-resilient spanner, and then add to it $O(W \cdot n^{3/2})$ edges from a carefully chosen set of short cycles in the original graph. The algorithm is simple and thus amenable to practical implementations, while the upper bound on the number of added edges is derived from sophisticated combinatorial arguments.

The same approach can be used for turning any given t-spanner into a σ-resilient t-spanner, for $\sigma \geq t > 3$. Once again, the total number of edges added to the initial spanners is $O(W \cdot n^{3/2})$. Our results for $\sigma = t = 3$ and for small edge weights seem to be the most significant ones, both from the theoretical and from the practical point of view. From a theoretical perspective, our σ-resilient 3-spanners have the same asymptotic size as their non-resilient counterparts. From a practical perspective, there is empirical evidence [3] that small stretch spanners provide the best performance in terms of stretch/size trade-offs, and that spanners of larger stretch are not likely to be of practical value. Table 1 puts our results in perspective with the fragility and the size of previously known spanners. Note that, for $\sigma = 2k-1$, our resilient spanner ensures $\text{frag}_S(e) \leq \max\{2k-1, \text{frag}_G(e)\}$, while for 1-edge fault-tolerant $(2k-1)$-spanner it holds $\text{frag}_S(e) \leq (2k-1) \cdot \text{frag}_G(e)$. This is achieved at the price of higher space requirements, for $k > 2$.

Table 1 Fragility and size of spanners

Spanner S	$\text{frag}_S(e)$	Size	Refs.
$(2k-1)$-spanner, $k \geq 2$	Unbounded	$O\left(n^{1+\frac{1}{k}}\right)$	[2]
1-edge fault-tolerant $(2k-1)$-spanner, $k \geq 2$	$\leq (2k-1) \cdot \text{frag}_G(e)$	$O\left(n^{1+\frac{1}{k}}\right)$	[12]
σ-resilient $(2k-1)$-spanner, $\sigma \geq 2k-1, k \geq 2$	$\leq \max\{\sigma, \text{frag}_G(e)\}$	$O\left(W \cdot n^{\frac{3}{2}}\right)$	This paper

Factor W in the last line is the ratio between maximum and minimum edge weight

Also (α, β)-spanners of unweighted graphs can be turned into σ-resilient (α, β)-spanners, for any $\sigma \geq \alpha + \beta$, using the same technique, adding $O(n^{3/2})$ edges in the worst case.

The remainder of this paper is organized as follows. We start with few preliminary definitions and basic observations in Sect. 2. In Sect. 3 we show some negative results on 2-resilient spanners and 1-edge fault-tolerant spanners. In Sect. 4 we describe our algorithm for computing σ-resilient spanners. In particular, we first describe in Sect. 4.1 a trivial approach. Next, in Sect. 4.2, we show how to bound the size of σ-resilient spanners. Finally, in Sect. 4.3, we show how to compute efficiently σ-resilient spanners. Section 5 lists some concluding remarks.

2 Preliminaries

We assume that the reader is familiar with standard graph terminology. In our paper, we deal with weighted undirected graphs, i.e., undirected graphs having weights associated to their edges. The length of a path is the sum of the weights of its edges. In unweighted graphs the length of a path is given by the number of its edges. Note that unweighted graphs can be seen as special cases of weighted graphs, where all the weights are 1. A shortest path is a path of minimum length between two vertices, and the distance between two vertices is given by the length of a shortest path between the two vertices. Let $G = (V, E)$ be an undirected graph. Throughout this paper, we use the notation $d_G(u, v)$ to denote the distance between vertices u and v in G. Let $F \subseteq E$ be any subset of edges in G: we denote by $G \setminus F$ the graph obtained after deleting edges in F from G. Note that, as a special case $G \setminus e$ denotes the graph obtained after deleting edge e from G. Similarly, we let $G \cup H$ denote the graph obtained after adding edges in H to G, where H and G have the same set of vertices.

Let $G = (V, E)$ be an undirected (weighted or unweighted) graph, with m edges and n vertices. A *bridge* is an edge $e \in E$ whose deletion increases the number of connected components of G. Note that an edge is a bridge if and only if it is not contained in any cycle of G. Graph G is *2-edge-connected* if it does not have any bridges. The *2-edge-connected components* of G are its maximal 2-edge-connected subgraphs. Let e be an edge in G, and denote by $\mathcal{C}_e$ the set of all the cycles containing e: if G is 2-edge-connected, then $\mathcal{C}_e$ is non-empty for each $e \in E$. We refer to a shortest

cycle among all cycles in $\mathcal{C}_e$ as a *short cycle for* edge e. Note that if G is 2-edge-connected, then at least one short cycle always exists for any edge. Short cycles are not necessarily unique: for each $e \in E$, we denote by $\Gamma_e(G)$ the set of short cycles for e in graph G. Let G be an undirected unweighted graph: the *girth* of G, denoted by $girth(G)$, is the length of a shortest cycle in G.

A t-spanner of a graph $G = (V, E)$ is a subgraph $S = (V, E_S)$, $E_S \subseteq E$ such that, for all vertices x, y, $d_S(x, y) \leq t \cdot d_G(x, y)$. For $\alpha \geq 1$ and $\beta \geq 0$, an (α, β)-spanner of an unweighted graph $G = (V, E)$ is a subgraph $S = (V, E_S)$, $E_S \subseteq E$ such that, for all vertices x, y, $d_S(x, y) \leq \alpha \cdot d_G(x, y) + \beta$.

The *fragility* of an edge $e = (u, v)$ in graph G is defined as $\text{frag}_G(e) = \max_{x,y \in V} \left\{ \frac{d_{G\setminus e}(x,y)}{d_G(x,y)} \right\}$. Given a graph G and a t-spanner S of G, and given $\sigma \geq 1$, we say that edge e is *σ-fragile in S* if $\text{frag}_S(e) > \max\left\{\sigma, \text{frag}_G(e)\right\}$. A t-spanner R is *σ-resilient* if $\text{frag}_R(e) \leq \max\{\sigma, \text{frag}_G(e)\}$ for each edge $e \in R$, i.e., if R does not contain σ-fragile edges.

The following lemma shows that in the definition of fragility of an edge $e = (u, v)$, the maximum is obtained for $\{x, y\} = \{u, v\}$, i.e., exactly at its two endpoints.

Lemma 1 *Let $G = (V, E)$ be a connected graph with positive edge weights, and let $e = (u, v)$ be any edge in G. Then* $\text{frag}_G(e) = \frac{d_{G\setminus e}(u,v)}{d_G(u,v)}$.

Proof Let x and y be any two vertices in G. To prove the lemma it suffices to show that $\frac{d_{G\setminus e}(x,y)}{d_G(x,y)} \leq \frac{d_{G\setminus e}(u,v)}{d_G(u,v)}$. We distinguish two cases, depending on whether there is a shortest path in G between x and y that avoids edge e or not. If there is such a shortest path, then $d_{G\setminus e}(x, y) = d_G(x, y)$. Since $d_{G\setminus e}(u, v) \geq d_G(u, v)$, the lemma trivially holds.

Assume now that all shortest paths between x and y in G go through edge $e = (u, v)$. In this case, $d_G(x, y) \geq d_G(u, v)$. If edge e is a bridge, then $\text{frag}_G(e) = \frac{d_{G\setminus e}(u,v)}{d_G(u,v)} = +\infty$, and again the lemma trivially holds. If e is not a bridge, then the graph $G\setminus e$ is connected. Since there is at least a (not necessarily shortest) path in $G\setminus e$ between x and y containing the shortest path in $G\setminus e$ from u to v, we have that $d_{G\setminus e}(x, y) \leq d_G(x, y) - d_G(u, v) + d_{G\setminus e}(u, v)$, or equivalently

$$\frac{d_{G\setminus e}(x, y)}{d_G(x, y)} \leq \frac{d_G(x, y) - d_G(u, v) + d_{G\setminus e}(u, v)}{d_G(x, y)} \tag{1}$$

Since $d_G(x, y) - d_G(u, v) + d_{G\setminus e}(u, v) \geq d_G(x, y)$, we can upper bound the right-hand side of (1) by subtracting $d_G(x, y) - d_G(u, v) \geq 0$ from both its numerator and denominator, thus yielding the lemma. □

Note that for unweighted graphs $d_G(u, v) = 1$, and thus Lemma 1 can be simply stated as $\text{frag}_G(e) = d_{G\setminus e}(u, v)$. The following simple lemma shows that, when inserting new edges into a graph G, the fragility of the old edges cannot increase.

Lemma 2 *Let G and H be any pair of weighted graphs on the same set of vertices, and let $G \cup H$ be the graph obtained after adding edges in H to G. Then,* $\text{frag}_{G\cup H}(e) \leq \text{frag}_G(e)$ *for each edge e in G.*

Proof Consider an edge $e = (u, v)$ in G. Since $G \subseteq G \cup H$, $d_{G\cup H}(u, v) \leq d_G(u, v)$. If $d_{G\cup H}(u, v) < d_G(u, v)$, a shortest path from u to v in $G \cup H$ avoids edge e. This is equivalent to saying that $d_{(G\cup H)\setminus e}(u, v) = d_{G\cup H}(u, v)$, and hence $\text{frag}_{G\cup H}(e) = 1 \leq \text{frag}_G(e)$. Otherwise, $d_{G\cup H}(u, v) = d_G(u, v)$ and $d_{(G\cup H)\setminus e}(u, v) \leq d_{G\setminus e}(u, v)$: again, $\text{frag}_{G\cup H}(e) \leq \text{frag}_G(e)$. □

Since a spanner S is a subgraph of the original graph G, an immediate consequence of Lemma 2 is the following:

Corollary 3 *If S is a spanner of G, then for each edge $e \in S$ we have* $\text{frag}_S(e) \geq \text{frag}_G(e)$.

The fragility of all edges in a graph $G = (V, E)$ with positive edge weights can be trivially computed in a total of $O(m^2n + mn^2 \log n)$ worst-case time by simply computing, for each edge $e \in E$, all-pairs shortest paths in graph $G\setminus e$. A faster bound of $O(mn + n^2 \log n)$ can be achieved by using either a careful modification of algorithm `fast-exclude` in [14] or by applying n times a modified version of Dijkstra's algorithm, as described in [19]. For unweighted graphs, the above bound reduces to $O(mn)$.

3 Some Negative Results

In this section we show some negative results on 2-resilient spanners and edge fault-tolerant spanners. We first establish that finding sparse 2-resilient spanners may be an overly ambitious goal, as there are dense graphs for which the only 2-resilient spanner is the graph itself.

Theorem 4 *There is an infinite family $\mathcal{F}$ of graphs such that for each graph $G \in \mathcal{F}$ the following properties hold:*

(1) *G has $\Theta(n^\delta)$ edges, with $\delta > 1.72598$, where n is the number of vertices of G.*
(2) *No proper subgraph of G is a 2-resilient spanner of G.*
(3) *There exists a 2-spanner S of G such that $\Theta(n^\delta)$ edges of $G\setminus S$, with $\delta > 1.72598$, need to be added back to S in order to make it 2-resilient.*

Proof The family $\mathcal{F}$ is defined as the set of graphs $\{I_3, I_6, I_9, \ldots, I_{3k}, \ldots\}$, with each I_{3k} being the complement of the intersection graph of all the k-sets contained in a $3k$-set, $k \geq 1$. Given a set U, with $|U| = 3k$, graph I_{3k} contains a vertex v_A for each subset $A \subset U$ with $|A| = k$, and vertex v_A is adjacent to vertex v_B if and only if $A \cap B = \emptyset$.

Graph I_{3k} has $n = \binom{3k}{k}$ vertices. Each vertex has degree $\binom{2k}{k}$, since this is the number of k-sets that can be chosen from the remaining $2k$ elements. Thus, I_{3k} has $m = \frac{n}{2}\binom{2k}{k}$ edges. The density δ of graph I_{3k}, where $\delta = \frac{\log_2 m}{\log_2 n}$, can be derived by Stirling approximation as follows:

$$n = \binom{3k}{k} \approx \frac{3^{3k}}{2^{2k}} \cdot \sqrt{\frac{3}{4\pi k}}$$

and

$$m = \frac{n}{2} \cdot \binom{2k}{k} \approx 3^{3k} \cdot \frac{\sqrt{3}}{4\pi k}$$

for sufficiently large k we can state that

$$\delta = \frac{\log_2 m}{\log_2 n} \approx \frac{3k \cdot \log_2 3}{3k \cdot \log_2 3 - 2k \cdot \log_2 2} = 1 + \frac{2}{3\log_2 3 - 2}$$

giving

$$m = \Theta\left(n^{1+\frac{2}{3\log_2 3-2}}\right)$$

where $\frac{2}{3\log_2 3-2} > 0.72598$. This proves Property (1).

We now turn to Property (2). We first claim that there is only one path of length 2 between any pair of adjacent vertices in I_{3k}. Indeed, for any two adjacent vertices v_A and v_B, there is exactly one vertex, namely $v_{U\setminus(A\cup B)}$, which is adjacent to both v_A and v_B. Thus each edge belongs to exactly one triangle, which implies that the fragility of any edge in I_{3k} is 2, and that there is only one path of length 2 between any pair of adjacent vertices.

Let $S \subset I_{3k}$ be a 2-spanner of I_{3k}. We show that S is not 2-resilient. Let v_A and v_B be two adjacent vertices in I_{3k} such that $(v_A, v_B) \notin S$, and let $C = U\setminus(A \cup B)$. We know that v_A, v_C, v_B is the only path of length 2 from v_A to v_B in I_{3k}. Since S is a 2-spanner of I_{3k}, both (v_A, v_C) and (v_C, v_B) must be in S. For the same reason above, the only path of length 2 in I_{3k} from v_A to v_C is v_A, v_B, v_C, so $d_{S\setminus\{(v_A,v_C)\}}(v_A, v_C) > 2$, because $(v_A, v_B) \notin S$. Thus $\text{frag}_S((v_A, v_C)) > 2$, while $\text{frag}_{I_{3k}}((v_A, v_C)) = 2$, which implies that S is not 2-resilient.

To prove Property (3), let S be a subgraph of I_{3k} obtained by deleting exactly one edge from each triangle in I_{3k}. Since each edge in I_{3k} is contained in exactly one triangle, there is always such an S, and it is a 2-spanner of I_{3k}. Furthermore, by Property (1), $\frac{m}{3} = \Theta(n^\delta)$ edges, with $\delta > 1.72598$, have to be deleted from I_{3k} in order to produce S. By Property (2), $\Theta(n^\delta)$ edges of $I_{3k}\setminus S$, with $\delta > 1.72598$, need to be added back to S in order to make it a 2-resilient 2-spanner of I_{3k}. □

Edge fault-tolerant spanners [12] provide a simple way to bound distance increases under edge faults. However, they are not σ-resilient, as the next theorem shows.

Theorem 5 *Let $G = (V, E)$ be a graph.*

(a) *Let S_f be any 1-edge fault tolerant t-spanner of G. Then* $\text{frag}_{S_f}(e) \le t \cdot \text{frag}_G(e)$ *for each $e \in S_f$.*
(b) *There exist 1-edge fault-tolerant t-spanners that are not σ-resilient, for any $\sigma < t^2/2$.*

Proof We first prove (a). By definition of 1-edge fault tolerant t-spanner, we have $d_{S_f \setminus e}(u, v) \leq t \cdot d_{G \setminus e}(u, v)$ for each edge $e = (u, v)$, and since $e \in S_f$ we have $d_{S_f}(u, v) \geq d_G(u, v)$.

The fragility of e in S_f is then:

$$\text{frag}_{S_f}(e) = \frac{d_{S_f \setminus e}(u, v)}{d_{S_f}(u, v)} \leq \frac{t \cdot d_{G \setminus e}(u, v)}{d_{S_f}(u, v)} \leq \frac{t \cdot d_{G \setminus e}(u, v)}{d_G(u, v)} = t \cdot \text{frag}_G(e).$$

To prove (b), consider the graph illustrated in Fig. 1. The subgraph defined by bold edges (i.e., the whole graph except edges e_i, with $1 \leq i \leq t$) is a 1-edge fault tolerant t-spanner. The fragility of edge e in the original graph is t, while its fragility in the spanner is $t^2/2$, i.e., it is greater than the fragility in the original graph by a factor of $t/2$. □

4 On σ-Resilient Spanners

In this section we present our algorithm for computing σ-resilient spanners, We start by describing a trivial approach in Sect. 4.1. Then, in Sect. 4.2, we introduce the notion of parsimonious sequence of cycles, which allows us to bound the size of a σ-resilient spanner. Finally, in Sect. 4.3, we show how to compute efficiently a parsimonious sequence of cycles.

4.1 A Simple-Minded Algorithm for Computing σ-Resilient Spanners

Given a graph G, a σ-resilient spanner R of G can be computed with the following simple approach:

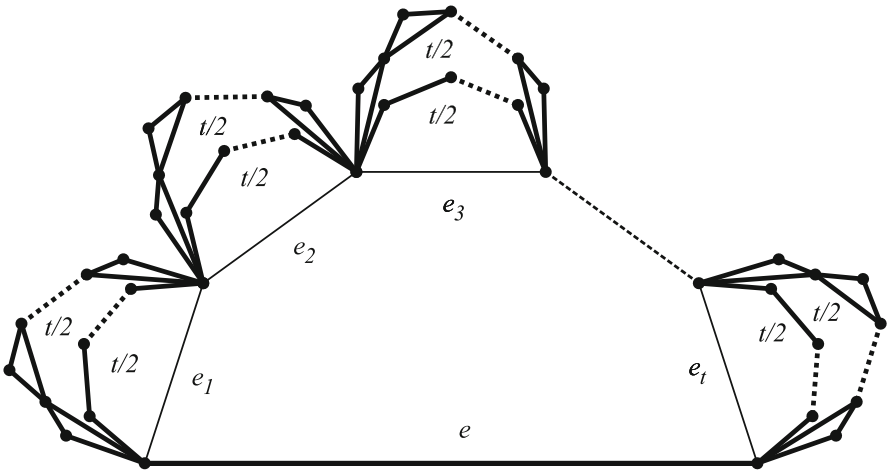

Fig. 1 A 1-edge fault tolerant t-spanner that is not σ-resilient for any $\sigma < t^2/2$. Edges e_i, $1 \leq i \leq t$, are not included in the t-spanner

1. Initialize R to S, where S is any t-spanner of G, with $t \leq \sigma$.
2. For each edge $e = (u, v)$ that is σ-fragile in S, select a shortest path between u and v in $G \backslash e$ and add it to R.

We refer to a shortest path between u and v in $G \backslash (u, v)$ as a *backup path* for edge (u, v). The correctness of our approach hinges on the following theorem.

Theorem 6 *Let S be a t-spanner of a weighted graph G, and let R be computed by adding to S a backup path for each σ-fragile edge e in S, with $\sigma \geq t$. Then R is a σ-resilient t-spanner of G.*

Proof R is trivially a t-spanner of G, since it contains a t-spanner S. It remains to show that R is σ-resilient, i.e., $\text{frag}_R(e) \leq \max\{\sigma, \text{frag}_G(e)\}$ for each edge $e \in R$. We first claim that this holds for each edge $e \in R \backslash S$. Indeed, in this case we have that $d_{R \backslash e}(u, v) \leq d_S(u, v) \leq t \cdot d_G(u, v)$, where the latter inequality follows immediately from the fact that S is a t-spanner of G, hence also $R \backslash e$ is a t-spanner of G. This implies that

$$\text{frag}_R(e) = \frac{d_{R \backslash e}(u, v)}{d_R(u, v)} \leq t \cdot \frac{d_G(u, v)}{d_R(u, v)} \leq t \leq \sigma \leq \max\{\sigma, \text{frag}_G(e)\}.$$

To complete the proof, it suffices to show that $\text{frag}_R(e) \leq \max\{\sigma, \text{frag}_G(e)\}$ for each edge $e \in S$. Let $e = (u, v)$ be any edge in S. If $\text{frag}_S(e) \leq \sigma$, the fact that $S \subseteq R$ implies by Lemma 2 that $\text{frag}_R(e) \leq \sigma \leq \max\{\sigma, \text{frag}_G(e)\}$. If $\text{frag}_S(e) > \sigma$, then a shortest path between u and v in $G \backslash e$ is added as a backup path for e, at which point the fragility of edge e in the resulting graph will be equal to $\text{frag}_G(e)$. Once again, Lemma 2 guarantees that the fragility of e will never decrease after adding other backup paths for different edges. At the end, when all the backup paths have been added, we will have $\text{frag}_R(e) \leq \text{frag}_G(e) \leq \max\{\sigma, \text{frag}_G(e)\}$. □

In the special case of unweighted graphs, Theorem 6 can be extended to (α, β)-spanners:

Corollary 7 *Let S be an (α, β)-spanner of an unweighted graph G, and let R be computed by adding to S a backup path for each σ-fragile edge $e \in S$, with $\sigma \geq \alpha + \beta$. Then R is a σ-resilient (α, β)-spanner of G.*

Proof We proceed as in the proof of Theorem 6, except in showing that $\text{frag}_R(e) \leq \max\{\sigma, \text{frag}_G(e)\}$ for each edge $e \in R \backslash S$. Since $R \backslash e$ is an (α, β)-spanner of G and $d_R(u, v) \geq 1$, we have

$$\begin{aligned}\text{frag}_R(e) = \frac{d_{R \backslash e}(u, v)}{d_R(u, v)} &\leq \frac{\alpha \cdot d_G(u, v) + \beta}{d_R(u, v)} \leq \alpha + \frac{\beta}{d_R(u, v)} \leq \alpha + \beta \leq \sigma \\ &\leq \max\{\sigma, \text{frag}_G(e)\}.\end{aligned}$$

□

Note that this approach has the additional benefit of producing a σ-resilient spanner R which inherits all monotone increasing properties of the underlying spanner S, i.e., all properties that are preserved under edge additions: for example, if S is fault-tolerant then R is fault-tolerant too. On the other hand, there can be several choices of backup paths for an edge with high fragility: if no particular care is taken in selecting suitable backup paths, we may end up with a resilient spanner of large size. In more detail, let $S(n)$ and $T(m, n)$ be respectively the number of edges of the initial spanner S and the time required for its computation, where m and n are respectively the number of edges and vertices in the original graph G. A trivial implementation of the above algorithm computes a σ-resilient spanner R with $O(n \cdot S(n))$ edges in a total of $O(T(m, n) + (m + n \log n) \cdot S(n))$ time.

In the next sections we will show how to refine our algorithm in order to improve these bounds, by reducing both the total number of edges added to the initial spanner S and the total time required to compute a σ-resilient spanner R from S.

4.2 Improving the Size of σ-Resilient Spanners

In this section we show how to refine our algorithm in order to build a σ-resilient 3-spanner for a graph with positive edge weights, containing $O(W \cdot n^{\frac{3}{2}})$ edges in the worst case, where W is the ratio between maximum and minimum edge weight. Our improvement is based on the following two high-level ideas:

(1) Bound the number of edges with high fragility (Theorem 8).
(2) Select carefully the shortest paths to be added as backup paths so that the total number of additional edges required is small (Theorem 9).

We start by bounding the number of high fragility edges.

Theorem 8 *Let $G = (V, E)$ be a graph with positive edge weights, an let $\sigma \geq 1$. Then, the number of edges of G having fragility greater than σ is $O(n^{1+1/\lfloor(\sigma+1)/2\rfloor})$.*

Proof Let L be the subgraph of G containing only the edges with fragility greater than σ in G. If L contains no cycle, then L has at most $(n-1)$ edges and the theorem trivially holds.

Otherwise, let C be a cycle in L, and let ℓ be the number of edges in C. Let $e = (u, v)$ be a maximum weight edge in C, and let $e_1, e_2, \ldots, e_{\ell-1}$ be the remaining edges in C. Since L contains e and it is a subgraph of G, we have by Lemma 2:

$$\sigma < \mathrm{frag}_G(e) \leq \mathrm{frag}_L(e) \,. \tag{2}$$

We claim that it must be $d_L(u, v) = w(e)$. Indeed, if $d_L(u, v) < w(e)$, we would have $d_{L\setminus e}(u, v) = d_L(u, v)$, and thus by Lemma 1

$$\mathrm{frag}_L(e) = \frac{d_{L\setminus e}(u, v)}{d_L(u, v)} = 1$$

which contradicts (2). Since $d_L(u, v) = w(e)$ and $w(e_i) \leq w(e)$, for $1 \leq i < \ell - 1$, we have that

$$\sigma < \text{frag}_G(e) \leq \text{frag}_L(e) = \frac{d_{L\setminus e}(u, v)}{d_L(u, v)} \leq \frac{\sum_{i=1}^{\ell-1} w(e_i)}{w(e)} \leq \ell - 1.$$

This implies that any cycle in L must have more than $(\sigma + 1)$ edges. Let L' be the unweighted version of L, i.e., L' has the same vertices and edges as L, but its edges are unweighted. Clearly,

$$\text{girth}(L') > \sigma + 1.$$

The theorem now follows directly from a result by Bondy and Simonovits [9], which states that a graph with girth greater than $\sigma + 1$ contains at most $O(n^{1+1/\lfloor(\sigma+1)/2\rfloor})$ edges. □

We now show that the shortest paths to be added as backup paths can be suitably chosen, so that the total number of additional edges is small. In the following, we assume that our input graph G is 2-edge-connected. This is without loss of generality: if G is not 2-edge-connected, then our algorithm can be applied separately to every 2-edge-connected component of G. Let $e = (u, v)$ be an edge of high fragility in the initial t-spanner. Note that, in order to identify a backup path for edge e, we can refer either to a shortest path between u and v in $G\setminus e$ or, equivalently, to a short cycle for e in G (i.e., the short cycle defined by one of the shortest paths in $G\setminus e$ and the edge e itself). In the following, we will identify backup paths by short cycles in G rather than by shortest paths in $G\setminus e$.

An ordered sequence of cycles $C_1, C_2, \ldots, C_q$ is said to be *parsimonious* if the following property holds: for any pair of cycles C_i and C_j, with $1 \leq i < j \leq q$, if C_i and C_j have two common vertices x and y, where x and y split C_j into paths P' and P'', then either $P' \subseteq \bigcup_{k=1}^{j-1} C_k$ or $P'' \subseteq \bigcup_{k=1}^{j-1} C_k$. Intuitively speaking, in a parsimonious sequence of cycles, each new cycle C_j reuses as much as possible portions of paths from the union of previous cycles $C_1, \ldots, C_{j-1}$. Figure 2 illustrates the notion of parsimonious sequence of cycles. The sequence of cycles C_1, C_2, C_3, C_4 shown in the left side of Fig. 2 is not parsimonious, since each path joining a and b along C_4 is not contained into $C_1 \cup C_2 \cup C_3$. A parsimonious sequence C_1, C_2, C_3, C'_4 is shown in the right side of Fig. 2, where cycle C'_4 is shown in bold. Cycles C'_4 and C_3 intersect in two points, namely b and d, and the path in C'_4 joining b to d through c is contained in $C_1 \cup C_2 \cup C_3$. The same holds for the path joining a and c, the two common vertices of C_1 and C'_4, through b, and for the path joining c and e, the two common vertices of C_2 and C'_4, through d.

The following theorem bounds the total number of different edges in a parsimonious sequence of cycles.

Theorem 9 *Given a graph G, any parsimonious sequence $C_1, C_2, \ldots, C_q$ of cycles in G contains $O(\min\{q\sqrt{n} + n, n\sqrt{q} + q\})$ edges.*

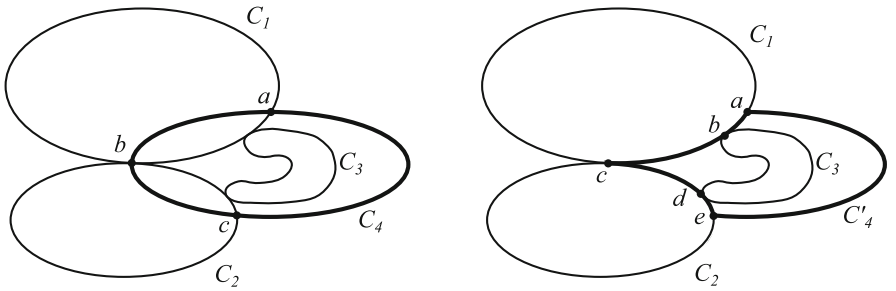

Fig. 2 The sequence of four cycles C_1, C_2, C_3, C_4 on the left is not parsimonious, while the sequence C_1, C_2, C_3, C'_4 on the right (where C'_4 is represented by the *bold line*) is parsimonious

Proof Let V_j and E_j be respectively the vertex set and the edge set of cycle C_j, for $1 \leq j \leq q$. We partition each edge set E_j into the following three disjoint sets:

- E_j^{old} (*old edges*): edges already in some E_i, $i < j$, i.e., edges in $E_j \cap \left(\bigcup_{i=1}^{j-1} E_i\right)$.
- E_j^{new} (*new edges*): edges with at least one endpoint not contained in $\bigcup_{i=1}^{j-1} V_i$.
- E_j^{cross} (*cross edges*): edges not contained in E_j^{old} but with both endpoints in $\bigcup_{i=1}^{j-1} V_i$.

To prove the theorem, we have to bound the total number of edges in $\bigcup_{j=1}^{q} E_j$. Note that we only need to bound the total number of new and cross edges (i.e., $|\bigcup_{j=1}^{q} E_j^{\text{new}}|$ and $|\bigcup_{j=1}^{q} E_j^{\text{cross}}|$), since each old edge in E_j^{old}, for any $1 \leq j \leq q$, is already accounted for either as a new edge or as a cross edge in some cycle C_i, $i < j$. Furthermore, each new edge in E_j^{new} can be amortized against a newly discovered vertex (i.e., a vertex $v \notin \bigcup_{i=1}^{j-1} V_i$), and in cycle C_j there can be at most two such edges which are incident to the same newly discovered vertex v: this implies that $|\bigcup_{j=1}^{q} E_j^{\text{new}}| \leq 2 \cdot n$.

To complete the proof, it remains to bound the total number of cross edges in the sequence. We do this as follows. For each cycle C_j we choose arbitrarily an orientation $\overrightarrow{C_j}$, in one of the two possible directions, and direct its edges accordingly. Given a directed edge $e = (x, y)$, we denote its endpoint x as tail(e) and its endpoint y as head(e). We build a bipartite graph $\mathcal{B}$ in which one vertex class represents the n vertices $v_1, v_2, \ldots, v_n$ in G, and the other vertex class represents the q directed cycles $\overrightarrow{C_1}, \overrightarrow{C_2}, ..., \overrightarrow{C_q}$. There is an edge in $\mathcal{B}$ joining vertex v and cycle C_j if and only if v is the tail of an edge in E_j^{cross} (hence, the degree of C_j in $\mathcal{B}$ is equal to the size of E_j^{cross}). Note that there is a one-to-one correspondence between edges in $\mathcal{B}$ and $\bigcup_{j=1}^{q} E_j^{\text{cross}}$. Thus, to prove the theorem it suffices to show that the number of edges in $\mathcal{B}$ is $O(\min\{q\sqrt{n} + n, n\sqrt{q} + q\})$.

We claim that there cannot exist two vertices x and y that are tails of two pairs of directed edges in E_i^{cross} and E_j^{cross} (see Fig. 3). Indeed, assuming $i < j$, the fact that the sequence of cycles is parsimonious implies that one of the two portions of C_j defined by x and y must contain only edges in E_j^{old}. The previous claim implies that

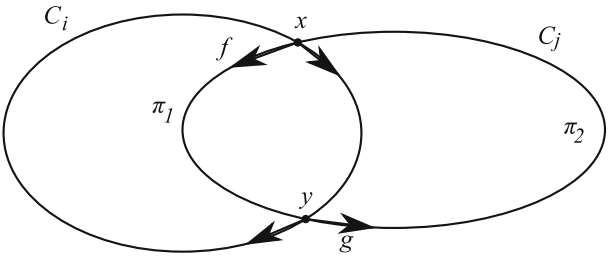

Fig. 3 On the proof of Theorem 9. If $i < j$, then either edge f or g in C_j is not in E_j^{cross}. In fact, either π_1 or π_2 should be included in $\bigcup_{k=1}^{j-1} C_k$

the bipartite graph $\mathcal{B}$ does not contain $K_{2,2}$ as a subgraph. Determining the maximum number of edges in $\mathcal{B}$ is a special case of Zarankiewicz's problem [26]. This problem has been solved by Kővári, Sós, Turán [20] (see also [22], p. 65), who proved that any bipartite graph G with vertex classes of size m and n containing no subgraph $K_{r,s}$, with the r vertices in the class of size m and the s vertices in the class of size n, has $O\left(\min\left\{mn^{1-1/r}+n, m^{1-1/s}n+m\right\}\right)$ edges, where the constant of proportionality depends on r and s. Since in our case the bipartite graph $\mathcal{B}$ has vertex classes of size n and q, and $r = s = 2$, it follows that $\mathcal{B}$ contains $O(\min\{q\sqrt{n}+n, n\sqrt{q}+q\})$ edges, which yields the theorem. □

We observe that Theorem 9 is related to a result of Coppersmith and Elkin [13] on distance preservers. Given a graph G and p pairs of vertices $\{(\} v_1, w_1), (v_2, w_2), \ldots, (v_p, w_p)$, a pairwise distance preserver is a subgraph S of G such that $d_S(v_i, w_i) = d_G(v_i, w_i)$, for $1 \leq i \leq p$. In particular, Coppersmith and Elkin [13] showed that it is always possible to compute a pairwise distance preserver containing $O(\min\{p\sqrt{n}+n, n\sqrt{p}+p\})$ edges. Their approach relies on the uniqueness of shortest paths between any pair of vertices, a property that does not hold in general in the case of parsimonious sequence of cycles, while our approach relies on weaker assumptions.

4.3 Efficiently Computing a Parsimonious Sequence of Short Cycles

To compute a σ-resilient t-spanner of graph G we start from a t-spanner S of G and add to S a parsimonious sequence of short cycles, in order to apply Theorem 9. Let $E_S(\sigma)$ be the set of σ-fragile edges in S (i.e., edges e with $\text{frag}_S(e) > \max\left\{\sigma, \text{frag}_G(e)\right\}$). For each edge $e \in E_S(\sigma)$, we find a short cycle for edge e in graph G and add that cycle to S. To guarantee the parsimonious property, we choose in a greedy fashion short cycles that reuse paths contained in the union of previously added cycles. We first describe how to find σ-fragile edges and next show how to compute a short cycle for each such edge.

The computation of σ-fragile edges can be accomplished in $O(mn + n^2 \log n)$ worst-case time by using an algorithm by Brandes for computing shortcut values (described in [19, Section 4.2.2]). We recall here that, given an edge e in graph G, the *shortcut value* of e, denoted by $\text{shval}_G(e)$, is defined as the maximum distance increase between any two vertices after the deletion of e:

$$\text{shval}_G(e) = \max_{x,y \in V} \left\{ d_{G \setminus e}(x, y) - d_G(x, y) \right\}.$$

Brandes' algorithm performs n Dijkstra-like visits, each time considering a different vertex as root; when starting a new visit from a root r, the algorithm computes the shortcut value of each edge incident to r. It can be seen ([21, Section 3.6.3]) that the distance increase after the deletion of edge $e = (u, v)$ is maximum for the endpoints of e, i.e.,

$$\text{shval}_G(e) = d_{G \setminus e}(u, v) - d_G(u, v).$$

As a consequence of Lemma 1, the fragility of an edge $e = (u, v)$ can be expressed as

$$\text{frag}_G(e) = \frac{\text{shval}_G(e)}{d_G(u, v)} + 1,$$

and thus it can be easily determined once the shortcut value of the same edge e is known.

A trivial algorithm for computing a parsimonious sequence of short cycles can be obtained as follows. For each edge $(u, v) \in E_S(\sigma)$, we compute a shortest path from u to v in $G \setminus (u, v)$: when comparing paths with the same weight, we select the path containing the smallest number of new edges (ties can be broken arbitrarily). This can be done by means of a slight modification of Dijkstra's algorithm and it requires a total of $O(|E_S(\sigma)| \cdot (m + n \log n))$ worst-case time, which is $O\left(n^{1+1/\lfloor(\sigma+1)/2\rfloor} \cdot (m + n \log n)\right)$ by Theorem 8.

We next show how to improve this time to $O\left(mn + n^2 \log n\right)$ in the worst case. Consider Algorithm `ResilientSpanner` illustrated in Fig. 4. For each vertex r in G, we first compute all the σ-fragile edges incident to r. Next, we augment the current spanner with one short cycle for each σ-fragile edge. Throughout, we will guarantee the important property that the total sequence of short cycles added during all the iterations of Algorithm `ResilientSpanner` is parsimonious. This will be accomplished by Algorithm `ParsimoniousCycles`, invoked in Line 4, which is the crux of the method.

Before describing the Algorithm `ParsimoniousCycles` in detail, we need few preliminary definitions. Let S' be the spanner at a generic iteration of the loop of Algorithm `ResilientSpanner`. Given a vertex r, we define a shortest path tree T_r of G rooted at r to be *parsimonious* if the following condition holds:

> for each vertex $x \neq r$, the path from r to x in T_r is a shortest path between r and x in G with the smallest number of edges in $G \setminus S'$.

It can be seen that, if T_r is parsimonious, then for each pair of vertices x', x'' such that x' is an ancestor of x'' in T_r, also the path from x' to x'' in T_r is a shortest path in G with the smallest number of edges in $G \setminus S'$. Note that a parsimonious shortest path tree T_r can be computed using a slight modification of the shortest path algorithm by Dijkstra: whenever two or more alternative paths with the same weight reach a vertex,

Algorithm ResilientSpanner(G, S, σ)
input:
graph G
a t-spanner S of G
a fragility threshold σ, with $\sigma \geq t$
output:
a σ-resilient t-spanner R of G, with $R \supseteq S$

1. **let** $S' = S$
2. **for each** vertex r in G
3. **let** E_r be the set of σ-fragile edges in S' incident to r
4. $S' = S' \cup \text{ParsimoniousCycles}(G, S', r, E_r)$
5. **return** $R = S'$

Fig. 4 Algorithm `ResilientSpanner`

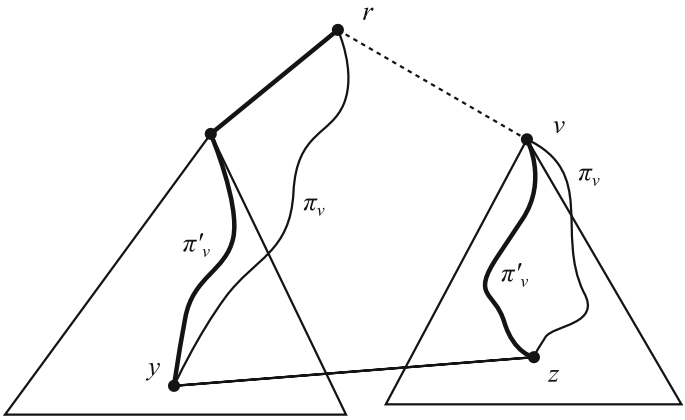

Fig. 5 On the proof of Lemma 10. *Solid paths* are contained in T_r

we select the path with the smaller number of edges in $G \setminus S'$ (ties can be broken arbitrarily). Moreover, let v be a vertex adjacent to r in T_r and let $\Pi_r(v)$ be the set of all shortest paths from r to v in $G \setminus (r, v)$. We denote a path $\pi_v \in \Pi_r(v)$ having the smallest number of edges from $G \setminus S'$ as a *best backup path* for edge (r, v). By definition, a best backup path for (r, v) does not contain edge (r, v) and thus it must contain at least one edge in $G \setminus T_r$. The following lemma shows that there must exist a best backup path for (r, v) containing exactly one edge of $G \setminus T_r$.

Lemma 10 *Let v be a vertex adjacent to r in T_r. There is at least one best backup path for (r, v) that contains exactly one edge of $G \setminus T_r$.*

Proof Let π_v be a best backup path for (r, v), and let $T_r(v)$ be the subtree of T_r rooted at v.

Walk along the path π_v starting from the root r, and let z be the first vertex encountered in the subtree $T_r(v)$ (see Fig. 5). Let y be the vertex immediately before z in π_v; clearly y is not in $T_r(v)$ and (y, z) is an edge of $G \setminus T_r$.

Let $\pi(r, y)$ be the path between r and y in T_r. Since T_r is a parsimonious shortest path tree, $\pi(r, y)$ is a shortest path between r and y in G with the smallest number

of edges in $G \setminus S'$. The same argument holds for the path $\pi(z, v)$ between z and v in T_r. Hence, the path $\pi'_v = \pi(r, y) \cdot (y, z) \cdot \pi(z, v)$ obtained by concatenating the path between r and y in T_r, edge (y, z), and the path between z and v in T_r must be a best backup path for (r, v), and it contains only one edge, i.e., (y, z), from $G \setminus T_r$. □

We denote by *single-cross backup paths* the best backup paths that contain exactly one edge from $G \setminus T_r$ (as in Lemma 10). Single-cross backup paths will be a crucial ingredient for finding efficiently a parsimonious sequence of short cycles. Note that, given a root r and a vertex v adjacent to r in T_r, combining the edge (r, v) with a single-cross backup path for (r, v) yields a short cycle for edge (r, v).

While computing a parsimonious shortest path tree T_r in $O(m + n \log n)$ time, in the same bound we can compute and store in each vertex x the following information:

- $\delta(x)$: the distance from r to x;
- $k(x)$: the number of edges from $G \setminus S'$ in the (shortest) path from r to x in T_r;
- $\text{apex}(x)$: the vertex such that $(r, \text{apex}(x))$ is the first edge in the (shortest) path from r to x in T_r;
- $p(x)$: the vertex immediately before x in the (shortest) path from r to x in T_r.

We are now ready to complete the low-level details of Algorithm `ResilientSpanner` of Fig. 4 by showing how to implement Algorithm `ParsimoniousCycles`, whose pseudo-code is illustrated in Fig. 6. We first sketch the main ideas behind the algorithm. We are given the current spanner S', a vertex r, and the set E_r of σ-fragile edges incident to r. In the following, we denote by π_v a single-cross backup path for (r, v). The objective of Algorithm `ParsimoniousCycles` is to compute π_v for each vertex v such that $(r, v) \in E_r$. By Lemma 10, π_v must be of the form $\pi_v = \pi(r, y) \cdot (y, z) \cdot \pi(z, v)$, where $\pi(u, w)$ denotes the unique path between vertices u and w in T_r, and (y, z) is an edge in $G \setminus T_r$ with $\text{apex}(y) \neq v$ and $\text{apex}(z) = v$. To compute π_v, it thus suffices to identify such an edge (y, z) in $G \setminus T_r$, so that the following two properties hold:

(1) The path $\pi_v = \pi(r, y) \cdot (y, z) \cdot \pi(z, v)$ is a shortest path in $G \setminus (r, v)$. Note that the weight of the path $\pi(r, y) \cdot (y, z) \cdot \pi(z, v)$ can be computed in constant time as $(\delta(y) + w(y, z) + (\delta(z) - w(r, v)))$.
(2) Among all shortest paths between r and v in $G \setminus (r, v)$, π_v has the smallest number of edges from $G \setminus S'$. Note that the number of edges from $G \setminus S'$ in $\pi(r, y) \cdot (y, z) \cdot \pi(z, v)$ can be computed in constant time as $(k(y) + k(z))$ if $(y, z) \in S'$, and as $(k(y) + k(z) + 1)$ otherwise.

Having this in mind, Algorithm `ParsimoniousCycles` works as follows. It stores in $\text{best}(v)$ the currently best single-cross backup path computed for each edge $(r, v) \in E_r$, where $\text{best}(v)$ is initialized in Lines 2–3 of Fig. 6. Next, the algorithm scans all edges (y, z) in $G \setminus T_r$ such that $(r, \text{apex}(z)) \in E_r$, as illustrated in Lines 4–5. Note that $\text{apex}(y) \neq \text{apex}(z)$ is further checked on Line 5, since by Lemma 10 when $\text{apex}(y) = \text{apex}(z)$ then the edge (y, z) cannot be in a single-cross backup path. Otherwise, the edge (y, z) can potentially induce a single-cross backup path $\gamma = \pi(r, y) \cdot (y, z) \cdot \pi(z, \text{apex}(z))$ for edge $(r, \text{apex}(z))$: if γ improves the previously known value for $\text{best}(\text{apex}(z))$, then $\text{best}(\text{apex}(z))$ gets updated in Line 8. At the end

Algorithm ParsimoniousCycles(G, S', r, E_r)
input:
graph G,
t-spanner S' of G,
the root vertex r,
the set E_r of σ-fragile edges in S incident to r
output:
a set of short cycles, one for each edge in E_r

1. compute a parsimonious shortest path tree T_r
2. **for each** v such that $(r, v) \in E_r$
3. **set** the current backup path best$(v) = \emptyset$
4. **for each** $y \in V \setminus r$
5. **for each** edge $(y, z) \in G \setminus T_r$ with apex$(y) \neq$ apex(z) and such that $(r, \text{apex}(z)) \in E_r$
6. **let** $\gamma = \pi(r, y) \cdot (y, z) \cdot \pi(z, \text{apex}(z))$ /* *where* $\cdot$ *denotes path concatenation* */
7. **if** path γ improves over best(apex(z))
8. best(apex(z)) $= \gamma$
9. **let** $C_r = \emptyset$
10. **for each** v such that $(r, v) \in E_r$
11. $C_r = C_r \cup \{\text{best}(v) \cup (r, v)\}$
12. **return** C_r

Fig. 6 Algorithm `ParsimoniousCycles`

of the loop in Lines 4–8, the algorithm has computed $\pi_v = \text{best}(v)$ for all edges $(r, v) \in E_r$. On Lines 9–11 it returns the corresponding short cycles.

The next theorem shows that the set of short cycles computed in Algorithm `ResilientSpanner` by the n calls to `ParsimoniousCycles` yields a parsimonious sequence (of short cycles).

Theorem 11 *There exists an ordering of the short cycles computed by Algorithm* `ResilientSpanner` *so that the resulting sequence is parsimonious.*

Proof Let $r_1, r_2, \ldots, r_n$ be the order in which the vertices in G are considered as roots by Algorithm `ResilientSpanner` (on Line 2 in Fig. 4). For each root r_i, with $1 \leq i \leq n$, let E_{r_i} be the set of σ-fragile edges incident to r_i in the original spanner S. Note that Algorithm `ResilientSpanner` computes a short cycle $C_{i,j}$ for each edge $(r_i, v_{i,j}) \in E_{r_i}$, with $1 \leq j \leq |E_{r_i}|$. Moreover, the short cycle $C_{i,j}$ consists of a single-cross backup path for edge $(r, v_{i,j})$ and the edge $(r_i, v_{i,j})$ itself. To prove the theorem, we show that the sequence of short cycles, $C_{i,j}$, for $1 \leq i \leq n$ and $1 \leq j \leq |E_{r_i}|$, sorted lexicographically by increasing (i, j), is parsimonious.

Let $C_{i,j}$ and $C_{i',j'}$ be any two short cycles in this sequence that share two vertices, with the pair (i', j') preceding pair (i, j) lexicographically. We now distinguish two cases, depending on whether $(i' < i)$ or $(i' = i$ and $j' < j)$.

Case $i' < i$: when cycle $C_{i,j}$ is computed by Algorithm `ParsimoniousCycles`, all the edges in $C_{i',j'}$ are already in the current spanner S'. Recall that $C_{i,j}$ is a short cycle for edge $(r_i, v_{i,j})$, and let (y, z) be the unique edge in $C_{i,j}$ that does not belong to T_{r_i}. By Lemma 10, the short cycle $C_{i,j}$ consists of the edge (y, z) plus two subpaths in T_{r_i}: a path π_1 from r_i to y and a path π_2 consisting of edge $(r_i, v_{i,j})$ followed by a path from $v_{i,j}$ to z (see Fig. 7). If $C_{i,j}$ and $C_{i',j'}$ share two vertices a and b, two cases may occur: either a and b are in the same subpath or

they are in two different subpaths of $C_{i,j}$. Furthermore, since both $C_{i,j}$ and $C_{i',j'}$ are short cycles, the portion of $C_{i,j}$ from a to b must have the same weight as the portion of $C_{i',j'}$ from a to b. We now consider the two cases separately.

In the first case, assume without loss of generality that both a and b are in π_1 (see Fig. 7a). Note that at this point the portion of the cycle $C_{i',j'}$ between a and b is already contained in S'. Since T_{r_i} is a parsimonious shortest path tree, the subpath π_1 in T_{r_i} must contain the minimum number of edges in $G \setminus S'$, which implies that also the portion of the subpath π_1 between a and b must be contained in S' (i.e., cycles $C_{i',j'}$ and $C_{i,j}$ satisfy the condition for being part of a parsimonious sequence).

In the second case, assume without loss of generality that a is in π_1 and b is in π_2 (see Fig. 7b). Once again, at this point the portion of the cycle $C_{i',j'}$ between a and b is already contained in S'. When $C_{i,j}$ is computed by Algorithm `ParsimoniousCycles`, the portion of $C_{i,j}$ between a and b passing through edge (y, z) must be contained in S' (otherwise the portion of $C_{i',j'}$ between a to b would have produced a cycle with the same weight and fewer edges of $G \setminus S'$). Once again, cycles $C_{i',j'}$ and $C_{i,j}$ satisfy the condition for being part of a parsimonious sequence.

Case $i' = i$ and $j' < j$: cycles $C_{i,j}$ and $C_{i,j'}$ are computed during the same call of Algorithm `ParsimoniousCycles` from a root r_i. Since Algorithm `ParsimoniousCycles` computes single-cross backup paths, each short cycle produced passes through the root r_i and traverses exactly two subtrees of r_i. We observe that if $C_{i,j}$ and $C_{i,j'}$ do not share any subtree, then they can intersect only at the root r_i. Hence, only the following two cases are possible for short cycles $C_{i,j}$ and $C_{i',j'}$ intersecting at two vertices:

- $C_{i,j}$ and $C_{i,j'}$ are contained in the same two subtrees: this case is shown in Fig. 8a. The intersection among cycles $C_{i,j}$ and $C_{i,j'}$ is the path in T_{r_i} joining the lowest common ancestor a of y and z' and the lowest common ancestor b of y' and z, where (y, z) (resp., (y', z')) is the cross edge for $C_{i,j}$ (resp., $C_{i,j'}$). In this case, any relative ordering among $C_{i,j}$ and $C_{i,j'}$ produces a parsimonious sequence.
- $C_{i,j}$ and $C_{i,j'}$ share one subtree: this case is shown in Fig. 8b. Also in this case, cycles $C_{i,j}$ and $C_{i,j'}$ share exactly a path, namely the path between the root r_i and the lowest common ancestor of z and y'. The same argument as in the previous case applies. □

The following theorems bound the running time of Algorithm `ResilientSpanner` and the number of edges in the computed σ-resilient t-spanner.

Theorem 12 *Algorithm* `ResilientSpanner` *runs in* $O(mn + n^2 \log n)$ *time in the worst case.*

Proof We first bound the time required by Algorithm `ParsimoniousCycles` in Fig. 4. The parsimonious shortest path tree T_r in Line 1 can be computed by a slight modification of Dijkstra's shortest path algorithm in $O(m + n \log n)$ worst case time, together with the auxiliary information about $\delta(x)$, $k(x)$, apex(x) and $p(x)$ for each

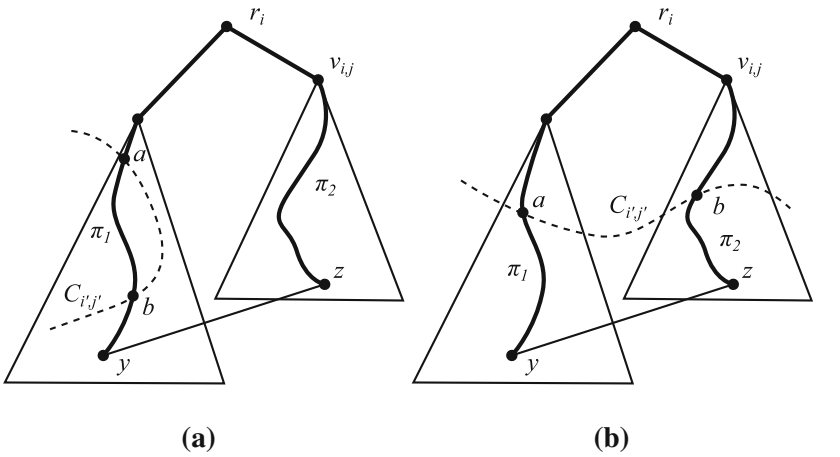

Fig. 7 Proof of Theorem 11. Case $i' < i$

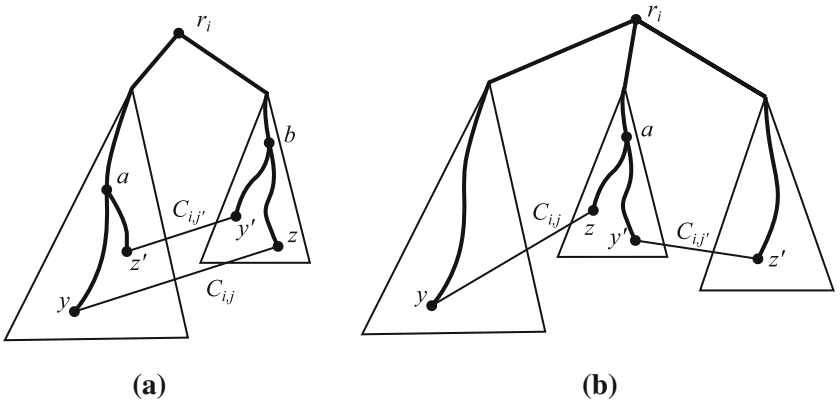

Fig. 8 Proof of Theorem 11. Case $i' = i$

vertex $x \in T_r$. Using this auxiliary information, each edge (y, z) can be processed in constant time in Lines 5–8. This implies that the total time spent through the loop in Lines 4–8 is bounded by $O(m)$. Also the time spent in the initialization (Lines 2–3) and for returning all short cycles (Lines 9–12) is $O(m)$. As a result, each call to Algorithm `ParsimoniousCycles` can be implemented in time $O(m + n \log n)$ in the worst case.

We now turn to Algorithm `ResilientSpanner` of Fig. 6. As it was previously mentioned, all the σ-fragile edges (Lines 2–3) can be computed in $O(mn + n^2 \log n)$ worst-case time by using Brandes' algorithm for computing shortcut values [19]. Since each call to Algorithm `ParsimoniousCycles` requires $O(m+n \log n)$ worst-case time, the overall running time of the algorithm is $O(mn + n^2 \log n)$ in the worst case. □

Theorem 13 *Let G be a graph with n vertices, with positive edge weights in $[w_{\min}, w_{\max}]$, and let $W = \frac{w_{\max}}{w_{\min}}$. Algorithm* `ResilientSpanner` *computes a σ-resilient t-spanner R of G, $\sigma \geq t$, with $R \supseteq S$, containing $O\left(W \cdot n^{3/2}\right)$ edges.*

Proof By Theorem 6, the subgraph R computed by Algorithm `ResilientSpanner` is a σ-resilient t-spanner of G, since it is obtained by adding a parsimonious sequence $\mathcal{C}$ of short cycles to a t-spanner S, one for each σ-fragile edge e in S.

Let C_e be the cycle in $\Gamma_e(G)$ added to the spanner, and let $S(n)$ be the number of edges in S. We partition σ-fragile edges $e \in S$ into three subsets, E_ℓ, E_m and E_h, according to their fragility in G. For each subset we separately bound the number of edges in the union of cycles in $\mathcal{C}$.

Low fragility edges: $E_\ell = \left\{e \in S \mid \sigma < \text{frag}_G(e) \leq 5\right\}$. By Theorem 8, we have

$$|E_\ell| = O\left(\min\left\{S(n), n^{1+\frac{1}{\left\lfloor\frac{\sigma+1}{2}\right\rfloor}}\right\}\right) = O\left(n^{1+\frac{1}{\left\lfloor\frac{\sigma+1}{2}\right\rfloor}}\right),$$

Thus, if $3 \leq \sigma < 5$ we have $|E_\ell| = O\left(\min\left\{S(n), n^{3/2}\right\}\right)$, while $E_\ell = \emptyset$ for $\sigma > 5$. Let e be any edge in E_ℓ. Since $\text{frag}_G(e) \leq 5$, cycle C_e contains at most $5W + 1$ edges (C_e contains exactly $\text{frag}_G(e) \cdot W + 1$ edges when $w(e) = w_{\max}$ and all other edges in C_e have weight $w_{\min}$). So, we have

$$\left|\bigcup_{e \in E_\ell} C_e\right| = O\left(W \cdot n^{3/2}\right) \text{ for } 3 \leq \sigma < 5,$$

while $\left|\bigcup_{e \in E_\ell} C_e\right| = 0$ for $\sigma \geq 5$;

Medium fragility edges: $E_m = \left\{e \in S \mid 5 < \text{frag}_G(e) < \log n\right\}$. By Theorem 8, since the fragility of each edge in E_m is greater than $\max\{\sigma, 5\}$, then

$$|E_m| = O\left(\min\left\{S(n), n^{1+\frac{1}{\lfloor(\sigma+1)/2\rfloor}}, n^{\frac{4}{3}}\right\}\right) = O\left(\min\left\{n^{1+\frac{1}{\lfloor(\sigma+1)/2\rfloor}}, n^{\frac{4}{3}}\right\}\right).$$

Each cycle C_e, with $e \in E_m$, contains at most $\log n \cdot W + 1$ edges, so we have

$$\left|\bigcup_{e \in E_m} C_e\right| = O\left(W \cdot \log n \cdot \min\left\{n^{1+\frac{1}{\lfloor(\sigma+1)/2\rfloor}}, n^{\frac{4}{3}}\right\}\right)$$

High fragility edges: $E_h = \left\{e \in S \mid \text{frag}_G(e) \geq \log n\right\}$. By Theorem 8, $|E_h| = O\left(n^{1+\frac{2}{\log n}}\right) = O(n)$, and by Theorem 9 we have

$$\left|\bigcup_{e \in E_h} C_e\right| = O\left(n \cdot \sqrt{|E_h|}\right) = O\left(n^{\frac{3}{2}}\right)$$

The total number of edges in R depends on values of σ and W:

- for $3 \leq \sigma < 5$
  the number of edges is $O\left(W \cdot n^{\frac{3}{2}}\right)$;
- for $5 \leq \sigma < \log n$ and $W = \Omega\left(\left(n^{\frac{1}{2} - \frac{1}{\lfloor(\sigma+1)/2\rfloor}}\right) / \log n\right)$
  the number of edges is $O\left(W \cdot \log n \cdot n^{1+\frac{1}{\lfloor(\sigma+1)/2\rfloor}}\right)$;
- for $\sigma \geq \log n$, or $\sigma \geq 5$ and $W = O\left(\left(n^{\frac{1}{2} - \frac{1}{\lfloor(\sigma+1)/2\rfloor}}\right) / \log n\right)$
  the number of edges is $O\left(n^{\frac{3}{2}}\right)$.

□

Note that, in the case of unweighted graphs, the number of edges in R is always $O\left(n^{3/2}\right)$. Thanks to Corollary 7, Algorithm `ResilientSpanner` can also be used to compute a σ-resilient (α, β)-spanner R of an unweighted graph, for any $\sigma \geq \alpha + \beta$, containing $O\left(n^{3/2}\right)$ edges in the worst case.

5 Conclusions and Open Problems

In this paper, we have investigated a new notion of resilience in graph spanners by introducing the concept of σ-resilient spanners. In particular, we have shown that it is possible to compute small stretch σ-resilient spanners of optimal size for graphs with small positive edge weights.

The techniques introduced for small stretch σ-resilient t-spanners can be used to turn any generic spanner (e.g., a fault-tolerant spanner, or, in the unweighted case, an (α, β)-spanner, for $\sigma \geq \alpha + \beta > 3$) into a σ-resilient spanner, by adding a suitably chosen set of at most $O(W \cdot n^{3/2})$ edges (that is, $O(n^{3/2})$ in the unweighted case).

Note that our upper bound on the size of σ-resilient spanners does not depend on σ, while the construction shows that the size decreases as σ increases. This leaves open the question of whether an improved analysis may yield better bounds for $\sigma > 3$. We expect that in practice our σ-resilient t-spanners, for $\sigma \geq t > 3$, will be substantially sparser than what it is implied by Theorem 13, and thus of higher value in applicative scenarios. Towards this aim, we plan to perform a thorough experimental study. It also remains an open question whether σ-resilient spanners of weighted graphs exist whose size does not depend on edge weights.

The concept of fragility can be naturally extended to vertices. In this framework, the study of vertex resilient spanners deserves further investigation.

Acknowledgments We thank the anonymous referees for their thoughtful reading of the paper and the valuable comments.

References

1. Aingworth, D., Chekuri, C., Indyk, P., Motwani, R.: Fast estimation of diameter and shortest paths (without matrix multiplication). SIAM J. Comput. **28**(4), 1167–1181 (1999)

2. Althofer, I., Das, G., Dobkin, D.P., Joseph, D., Soares, J.: On sparse spanners of weighted graphs. Discrete Comput. Geom. **9**, 81–100 (1993)
3. Ausiello, G., Demetrescu, C., Franciosa, P.G., Italiano, G.F., Ribichini, A.: Graph spanners in the streaming model: an experimental study. Algorithmica **55**(2), 346–374 (2009)
4. Ausiello, G., Franciosa, P.G., Italiano, G.F.: Small stretch spanners on dynamic graphs. J. Graph Algorithms Appl. **10**(2), 365–385 (2006)
5. Ausiello, G., Franciosa, P.G., Italiano, G.F., Ribichini, A.: Computing graph spanner in small memory: fault-tolerance and streaming. Discrete Math. Algorithms Appl. **2**(4), 591–605 (2010)
6. Baswana, S.: Dynamic algorithms for graph spanners. In: Proceedings of 13th Annual European Symposium on Algorithms (ESA'06), pp. 76–87 (2006)
7. Baswana, S., Kavitha, T., Mehlhorn, K., Pettie, S.: Additive spanners and (alpha, beta)-spanners. ACM Trans. Algorithms **7**(1), 5:1–5:26 (2010)
8. Baswana, S., Khurana, S., Sarkar, S.: Fully dynamic randomized algorithms for graph spanners. ACM Trans. Algorithms **8**(4), 35:1–35:51 (2012)
9. Bondy, J.A., Simonovits, M.: Cycles of even length in graphs. J. Comb. Theory Ser. B **16**(2), 97–105 (1974)
10. Braunschvig, G., Chechik, S., Peleg, D.: Fault tolerant additive spanners. In: Graph-Theoretic Concepts in Computer Science—38th International Workshop, (WG'12), volume 7551 of LNCS, pp 206–214. Springer, Berlin (2012)
11. Chechik, S.: New additive spanners. In: Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'13), pp. 498–512 (2013)
12. Chechik, S., Langberg, M., Peleg, D., Roditty, L.: Fault tolerant spanners for general graphs. SIAM J. Comput. **39**(7), 3403–3423 (2010)
13. Coppersmith, D., Elkin, M.: Sparse sourcewise and pairwise distance preservers. SIAM J. Discrete Math. **20**(2), 463–501 (2006)
14. 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)
15. Dinitz, M., Krauthgamer, R.: Fault-tolerant spanners: better and simpler. In: Proceedings of the 30th Annual ACM Symposium on Principles of Distributed Computing (PODC'11), pp. 169–178 (2011)
16. Dor, D., Halperin, S., Zwick, U.: All-pairs almost shortest paths. SIAM J. Comput. **29**(5), 1740–1759 (2000)
17. Elkin, M.: Streaming and fully dynamic centralized algorithms for constructing and maintaining sparse spanners. ACM Trans. Algorithms **7**(2), 1–17 (2011)
18. Halperin, S., Zwick, U.: Linear time deterministic algorithm for computing spanners for unweighted graphs (1996) (unpublished manuscript)
19. Jacob, R., Koschützki, D., Lehmann, K.A., Peeters, L., Tenfelde-Podehl, D.: Algorithms for centrality indices. In: Brandes, U., Erlebach, T. (eds.) Network Analysis, volume 3418 of LNCS, pp. 62–82. Springer, Berlin (2005)
20. Kővári, T., Sós, V.T., Turán, P.: On a problem of K. Zarankiewicz. Colloq. Math. **3**(1), 50–57 (1954)
21. Koschützki, D., Lehmann, K.A., Peeters, L., Richter, S., Tenfelde-Podehl, D., Zlotowski, O.: Centrality indices. In: Brandes, U., Erlebach, T. (eds.) Network Analysis, volume 3418 of LNCS, pp. 16–61. Springer, Berlin (2005)
22. Matoušek, J.: Lectures on Discrete Geometry. Springer, Berlin (2002)
23. Nisan, N., Ronen, A.: Algorithmic mechanism design. Games Econ. Behav. **35**(1–2), 166–196 (2001)
24. Peleg, D., Shäffer, A.: Graph spanners. J. Graph Theory **13**, 99–116 (1989)
25. Roditty, L., Thorup, M., Zwick, U.: Deterministic constructions of approximate distance oracles and spanners. In: Proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP'05), volume 3580 of LNCS, pp. 261–272. Springer, Berlin (2005)
26. Zarankiewicz, K.: Problem p 101. Colloq. Math. **2**, 301 (1951)