Small stretch spanners on dynamic graphs*

Giorgio Ausiello† Paolo G. Franciosa‡ Giuseppe F. Italiano§

Abstract

We present fully dynamic algorithms for maintaining 3- and 5-spanners of undirected graphs under a sequence of update operations. For unweighted graphs we maintain a 3- or 5-spanner under insertions and deletions of edges; each operation is performed in $O(n)$ amortized time over a sequence of $\Omega(n)$ updates. The maintained 3-spanner (resp., 5-spanner) has $O(n^{3/2})$ edges (resp., $O(n^{4/3})$ edges), which is known to be optimal. On weighted graphs with d different edge cost values, we maintain a 3- or 5-spanner in $O(n)$ amortized time over a sequence of $\Omega(d \cdot n)$ updates. The maintained 3-spanner (resp., 5-spanner) has $O(d \cdot n^{3/2})$ edges (resp., $O(d \cdot n^{4/3})$ edges). The same approach can be extended to graphs with real-valued edge costs in the range $[1, C]$.

All our algorithms are deterministic and are substantially faster than recomputing a spanner from scratch after each update.

1 Introduction

Graph spanners arise in many applications, including communication networks, computational biology, computational geometry, distributed computing, and robotics ([1, 2, 3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15]). Intuitively, a spanner of a graph is a subgraph that preserves approximate distances between all pairs of vertices. More formally, given $t \geq 1$, a t-spanner of a graph G is a subgraph S of G such that for each pair of vertices the distance in S is at most t times the distance in G; t is referred to as the *stretch factor* of the spanner. The best time bound for computing a t-spanner of a weighted graph with n vertices and m edges is $O(m+n)$, and is given by Baswana and Sen [4]. Their algorithm is randomized and computes spanners of size $O\left(t \cdot n^{1+\frac{2}{t+1}}\right)$. In the case of unweighted graphs, it is possible to compute a t-spanner in $O(m+n)$ time with a deterministic algorithm [18]. For weighted graphs, a deterministic algorithm is given in [1]; the best known implementation of this algorithm has running time $O(m \cdot n^{1+2/(t+1)})$.

Small stretch spanners offer a good compromise between sparsity and distance stretch: maintaining a t-spanner may be practical in the case of very large graphs, whose edges must be stored in external memory, while spanner edges could fit into main memory. A graph with million vertices could need TeraBytes to store its edges, while the edges in its 3-spanner or 5-spanner only need order of GigaBytes, at the cost of a limited distance stretch.

While there has been a lot of progress in the area of dynamic graph problems, to the best of our knowledge no fully dynamic algorithm for maintaining a t-spanner of a general weighted graph

*Partially supported by the Italian MIUR Project ALGO-NEXT "Algorithms for the Next Generation Internet and Web: Methodologies, Design and Experiments".

†Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", via Salaria 113, I-00198 Roma, Italy. ausiello@dis.uniroma1.it

‡Dipartimento di Statistica, Probabilità e Statistiche Applicate, Università di Roma "La Sapienza", piazzale Aldo Moro 5, I-00185 Roma, Italy. paolo.franciosa@uniroma1.it

§Dipartimento di Informatica, Sistemi e Produzione, Università di Roma "Tor Vergata", via del Politecnico 1, 00133 Roma, Italy. italiano@disp.uniroma2.it

under edge insertions and/or deletions is known, and for this problem only partially dynamic solutions were announced in [4]. A related direction of research is concerned with the maintenance of approximate distances, i.e., a query on the distance between two vertices is answered with a guaranteed approximation factor (see [16] for recent results and references). These results are usually obtained using $\Omega(n^2)$ space, while in the case of t-spanners we are interested in representing a sparse structure that maintains distances in the original graph.

In this paper, we contribute a first step towards the maintenance of dynamic graph spanners by presenting a fully dynamic deterministic algorithm for maintaining 3- and 5-spanners of unweighted graphs. Our algorithm supports an intermixed sequence of $\Omega(n)$ edge insertions and deletions in $O(n)$ amortized time per operation. The maintained 3-spanner has $O(n^{3/2})$ edges, while the 5-spanner has $O(n^{4/3})$ edges. Wenger [17] shows how to build graphs with $\Theta(n^{3/2})$ edges having no cycles of length less than 5, or with $\Theta(n^{4/3})$ edges having no cycles of length less than 7. For such graphs, no proper subgraphs preserve distances within stretch factor 3 (resp., 5). This implies that the size of our spanners is asymptotically optimal.

The same approach can be extended to weighted graphs with d different edge cost values. On a sequence of $\Omega(d \cdot n)$ intermixed edge insertions and deletions the amortized time per operation is still $O(n)$. The maintained 3-spanner has $O(d \cdot n^{3/2})$ edges, while the 5-spanner has $O(d \cdot n^{4/3})$ edges. This is optimal for constant d.

On graphs with real-valued edge costs in $[1, C]$, for $t > 3$ we can maintain a t-spanner with $O(n^{3/2} \cdot \log_{t/3} C)$ edges in $O(n)$ amortized time per operation over a sequence of $\Omega(n \cdot \log_{t/3} C)$ edge insertions and edge deletions. For $t > 5$, a t-spanner with $O(n^{4/3} \cdot \log_{t/5} C)$ edges can be maintained in $O(n)$ amortized time per operation over a sequence of $\Omega(n \cdot \log_{t/5} C)$ edge insertions and edge deletions.

Our algorithms are deterministic and are substantially faster than recomputing a spanner from scratch. To achieve our results, we dynamize the static randomized technique of Baswana and Sen [4], and make the resulting algorithm deterministic rather than randomized. Our algorithms use simple data structures, and thus seem amenable to practical implementations.

The remainder of the paper is organized as follows. We present a clustering scheme in Section 2. In Section 3 we show a tight relationship between this clustering and 3- and 5-spanners. Next, our dynamic algorithm for unweighted graphs is presented in Section 4, where we show how to build a clustering and the associated 3- or 5-spanner, and how a clustering and the associated 3- or 5-spanner can be updated under edge deletions; the amortized complexity on a sequence of edge deletions is discussed in Subsection 4.3. This decremental algorithm is made fully dynamic in Section 5. In Section 6 we show how the same approach can be extended to graphs with d different edge costs and to graphs with positive real edge costs. Section 7 lists some concluding remarks.

2 Definitions

2.1 Basic definitions

We assume that the reader is familiar with the standard graph terminology, as contained for instance in [8]. Let $G = (V, E)$ be an undirected graph, with V being the set of vertices and E the set of edges. Throughout the paper, we denote by n the number of vertices and by m the number of edges in a graph. If the graph is weighted, there is a real-valued cost $c(e) \geq 0$ associated with each edge $e \in E$.

Given a vertex x, its *neighborhood* is the set $N(x) = \{x\} \cup \{y \mid (x, y) \in E\}$ (note that $x \in N(x)$ by definition). Given two vertices $u, v \in V$, a *path* π *in* $G = (V, E)$ *connecting vertex* u *to vertex* v is a sequence of vertices $u = v_0, v_1, \ldots, v_\ell = v$ such that $(v_{i-1}, v_i) \in E$, for $0 < i \leq \ell$. We say

that each edge (v_{i-1}, v_i) is in path π, for $0 < i \leq \ell$. The *length of a path* π is given by the number of edges in π. If the graph is weighted, the *cost of a path* π is the sum of the costs of edges in π:

$$c(\pi) = \sum_{i=1}^{\ell} c(v_{i-1}, v_i) \ .$$

In the case of unweighted graphs the cost of a path is simply its length.

The *distance* $dist_G(u, v)$ *from* u *to* v *in* G is given by the minimum cost of a path in G from u to v (or $+\infty$ if there is no such path). A *shortest path* from u to v is then defined as any path π from u to v with $c(\pi) = \text{dist}_G(u, v)$. A graph $G' = (V', E')$ is a *subgraph* of graph $G = (V, E)$ if $V' \subseteq V$ and $E' \subseteq E$.

Given a graph G, a t-spanner S is a subgraph of G which preserves distances up to a factor t (the *stretch factor*). More formally,

Definition 1 *Let $G = (V, E)$ be a weighted graph, and let t be a real value, with $t \geq 1$. A* t-spanner *of G is a graph $S = (V, E')$ with $E' \subseteq E$ such that the following holds:*

$$\forall\ u, v \in V \qquad dist_S(u, v) \leq t \cdot dist_G(u, v). \tag{1}$$

The following lemma is by Peleg and Shäffer [13]:

Lemma 1 [13] *A subgraph $S = (V, E')$ of $G = (V, E)$, is a t-spanner of G if and only if the following holds:*

$$\forall\ (x, y) \in E \qquad dist_S(x, y) \leq t \cdot c(x, y). \tag{2}$$

2.2 Clustering

Definition 2 *Let $x_1, x_2, \ldots, x_k$, with $k \geq 1$, be distinct vertices in V, and let $\Gamma = \{Cl(x_1), Cl(x_2), \ldots, Cl(x_k)\}$ be a family of subsets of V. Given a real number $\ell \geq 1$, Γ is an* ℓ-clustering *of $G = (V, E)$ if the following properties hold:*

1. *$Cl(x_i) \subseteq N(x_i)$ for $1 \leq i \leq k$;*
2. *$Cl(x_i) \cap Cl(x_j) = \emptyset$, for each $i \neq j$;*
3. *$\bigcup_{i=1}^{k} Cl(x_i) = \bigcup_{i=1}^{k} N(x_i)$;*
4. *$|Cl(x_i)| \geq \ell$ for $1 \leq i \leq k$.*

Set $Cl(x_i)$ is called cluster*, and x_i is denoted as its* center*.*

Note that, according to the previous definition, the center x_i of cluster $Cl(x_i)$ may belong to another cluster $Cl(x_j)$, with $i \neq j$. As a special case, we define an *empty clustering* to be a clustering with no clusters. We assume that the empty clustering is an ℓ-clustering, for any ℓ. An example of 5-clustering is shown in the left part of Figure 1

Given an ℓ-clustering, a vertex is called *clustered* if it belongs to a cluster, and *free* otherwise; if vertex y is clustered, $center(y)$ denotes the center of the cluster containing y. For each $v \in V$, we define its *free neighborhood* $FN(v)$ as $FN(v) = N(v) \setminus \left(\bigcup_{i=1}^{k} Cl(x_i)\right)$.

Note that an ℓ-clustering contains at most n/ℓ clusters, each of size at least ℓ. We say that an ℓ-clustering is *maximal* if $|FN(v)| < 2\ell$, for each $v \in V$.

We remark that our definition of clustering is more strict that the one given in [4]. In the case of unweighted graphs, the construction of spanners starting from our definition of clustering is simpler and deterministic.

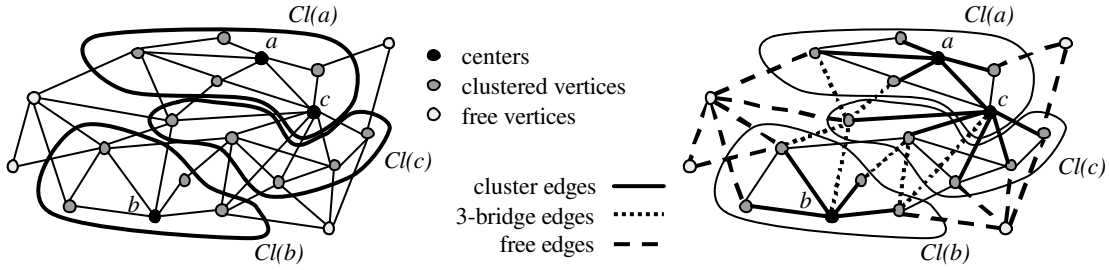

Figure 1: a 5-clustering of a graph and an associated 3-spanner

---

3 Clusterings and spanners

In this section we show how small stretch spanners of unweighted graphs can be produced by proper ℓ-clusterings. In particular, we describe how to produce a 3-spanner from a $n^{1/2}$-clustering, and a 5-spanner from a $n^{1/3}$-clustering.

3.1 Clusterings for 3-spanners

Definition 3 *Given a $n^{1/2}$-clustering Γ of $G = (V, E)$, we say that a subgraph $G' = (V, E')$ of G is* 3-compatible *with Γ if E' is the union of the following sets of edges:*

cluster edges: *all edges (x, y) such that y is clustered and $x = center(y)$;*

free edges: *all edges $(x, y) \in E$ such that either x or y is a free vertex;*

3-bridge edges: *for each cluster $Cl(x_i)$ and each vertex $y \in (Cl(x_j) \setminus \{x_i\})$, with $x_j \neq x_i$, one arbitrary edge $(x, y) \in E$ such that $x \in Cl(x_i)$. We say that edge (x, y)* connects *vertex y to cluster $Cl(x_i)$.*

Theorem 1 *Given a graph $G = (V, E)$ and a $n^{1/2}$-clustering Γ, if $G' = (V, E')$ is a subgraph of G 3-compatible with Γ then G' is a 3-spanner of G.*

Proof. We show that for any edge $(a, b) \in E$ there is a path of length at most 3 in G'. There can be only 3 cases, depending on a and b.

- Both a and b belong to the same cluster: in this case one of the following holds:
  - one among a and b is the center of the cluster, thus (a, b) is a cluster edge in G';
  - a third vertex x is the center of the cluster, thus (a, x) and (x, b) are cluster edges in G'.
- Vertices a and b belong to different clusters. Let $a \in Cl(x_i)$ and $b \in Cl(x_j)$: in such a case there must be a 3-bridge edge (b, y), where $y \in Cl(x_i)$. If $y \neq a$, then the cluster edges provide a path of length at most 2 from y to a, thus giving a path of length at most 3 from a to b in G'.
- At least one among a and b is a free vertex: in this case (a, b) is a free edge in G'.

□

Due to Theorem 1, we will refer to a subgraph of G 3-compatible with an $n^{1/2}$-clustering Γ as a *3-spanner associated with* Γ. The right side of Figure 1 shows a 3-spanner associated with the 5-clustering on the left.

If the $n^{1/2}$-clustering is maximal, we can prove that any associated 3-spanner is sparse:

Theorem 2 *A 3-spanner $G' = (V, E')$ associated to a maximal $n^{1/2}$-clustering contains $O(n^{3/2})$ edges.*

Proof. There are at most n cluster edges, since each vertex can be in at most one cluster. There is at most one 3-bridge edge for each possible pair $\langle x, Cl(x_i)\rangle$: since there are at most $n^{1/2}$ clusters, there are at most $n^{3/2}$ 3-bridge edges. We finally bound the number of free edges: since Γ is maximal, we have $|FN(v)| < 2n^{1/2}$ for any vertex v. Hence, the total number of free edges is at most $2 \cdot n^{3/2}$. □

3.2 Clusterings for 5-spanners

Definition 4 *Given an $n^{1/3}$-clustering Γ of $G = (V, E)$, we say that a subgraph $G' = (V, E')$ of G is* 5-compatible *with Γ if E' is the union of the following sets of edges:*

cluster edges: *all edges (x, y) such that y is clustered and $x = center(y)$;*

free edges: *all edges $(x, y) \in E$ such that either x or y is a free vertex;*

5-bridge edges: *for each pair of clusters $Cl(x_i), Cl(x_j)$, with $x_i \neq x_j$, one arbitrary edge $(x, y) \in E$ such that $x \in Cl(x_i)$ and $y \in Cl(x_j)$. We say that edge (x, y)* connects *clusters $Cl(x_i)$ and $Cl(x_j)$.*

We remark that the only difference with Definition 3 lies in the set of bridge edges.

Theorem 3 *Given a graph $G = (V, E)$ and an $n^{1/3}$-clustering Γ, if $G' = (V, E')$ is a subgraph of G 5-compatible with Γ then G' is a 5-spanner of G.*

Proof. We show that for any edge $(a, b) \in E$ there is a path of length at most 5 in G'. There can be only 3 cases, depending on a and b.

- Both a and b belong to the same cluster: in this case one of the following holds:
  - one among a and b is the center of the cluster, thus (a, b) is a cluster edge in G';
  - a third vertex x is the center of the cluster, thus (a, x) and (x, b) are cluster edges in G'.
- Vertices a and b belong to different clusters. Let $a \in Cl(x_i)$ and $b \in Cl(x_j)$: in such a case there must be a 5-bridge edge (x, y), where $x \in Cl(x_i)$ and $y \in Cl(x_j)$. Since cluster edges provide paths of length at most 2 from a to x and from y to b, we have a path of length at most 5 from a to b in G'.
- At least one among a and b is a free vertex: in this case (a, b) is a free edge in G'.

□

Due to Theorem 3, we will refer to a subgraph of G 5-compatible with an $n^{1/3}$-clustering Γ as a *5-spanner associated with* Γ. If the $n^{1/3}$-clustering is maximal, we can prove that any associated 5-spanner is sparse:

Theorem 4 *A 5-spanner $G' = (V, E')$ associated to a maximal $n^{1/3}$-clustering contains $O(n^{4/3})$ edges.*

Proof. There are at most n cluster edges, since each vertex can be in at most one cluster. There is at most one 5-bridge edge for each possible pair of clusters: since there are at most $n^{2/3}$ clusters, there are at most $n^{4/3}$ 5-bridge edges. We finally bound the number of free edges: since Γ is maximal, we have $|FN(v)| < 2 \cdot n^{1/3}$ for any vertex v. Hence, the total number of free edges is at most $2 \cdot n^{4/3}$. □

4 Decremental algorithms for 3- and 5-spanners

The main contribution of this paper is to provide a fully dynamic deterministic algorithm for maintaining 3-spanners and 5-spanners of unweighted graphs.

The construction and maintenance of our spanners is based on the maintenance of a maximal ℓ-clustering. Starting from any ℓ-clustering, a maximal ℓ-clustering can be obtained with a simple greedy algorithm, as illustrated in Figure 2. We observe that the same algorithm can be used for computing a maximal ℓ-clustering from scratch, starting from the initial empty clustering $\Gamma = \emptyset$.

Procedure MaximalCluster
input: graph $G = (V, E)$
ℓ-clustering Γ
output: maximal ℓ-clustering Γ

1. **while** there is a vertex x with $|FN(x)| \geq 2 \cdot \ell$
2. make x a center
3. make $Cl(x) = FN(x)$
4. add $Cl(x)$ to Γ

Figure 2: Procedure MaximalCluster.

The following theorem can be easily proved.

Theorem 5 *Procedure MaximalCluster computes a maximal ℓ-clustering of G.*

Proof. The fact that Procedure MaximalCluster computes an ℓ-clustering can be easily seen by induction on the number of clusters added to Γ. We assume Γ is an ℓ-clustering before applying the procedure. In particular, this is true for the empty clustering. We show now that any time a new cluster $Cl(x)$ is added to Γ all the properties of Definition 2 are maintained:

- Property 1: $Cl(x) = FN(x) \subseteq N(x)$ by definition of free neighborhood;
- Property 2: $Cl(x)$ only contains free vertices, hence it is disjoint from all existing clusters;
- Property 3: all free vertices in $N(x)$ are included in $Cl(x)$;
- Property 4: $Cl(x)$ has size at least $2 \cdot \ell$.

Moreover, Γ is maximal, since at the end there are no vertices having $|FN(x)| \geq 2 \cdot \ell$. □

We show in Subsection 4.1 how to update a maximal $n^{1/2}$-clustering and an associated 3-spanner during a sequence of edge deletions only. Next, in Subsection 4.2, we show how to maintain a 5-spanner during a sequence of edge deletions.

4.1 Maintaining 3-spanners

Procedure MaximalCluster builds an ℓ-clustering Γ by adding one cluster at a time. We now show how to maintain a 3-spanner associated with Γ when new clusters are added, in the case $\ell = n^{1/2}$.

For each vertex x we maintain the following simple data structures, that represent the current spanner plus some auxiliary information.

- The number $|FN(x)|$ of free vertices in $N(x)$.
- For each cluster $Cl(c)$, the list of edges $e = (x, z) \in E$ such that $z \in Cl(c)$. This list represents the candidate edges for connecting x to $Cl(c)$; we may assume that the 3-bridge edge in the spanner connecting x to $Cl(c)$, for each cluster, is the first edge in the list.
- A flag indicating whether x is clustered. If x is clustered:
  - a reference to the cluster containing x;
  - a reference to $center(x)$;
  - the list of all 3-bridge edges (y, x) incident to x, connecting any vertex y to the cluster containing x.
- A flag indicating whether x is a center. If x is a center:
  - the list of vertices in $Cl(x)$. This list implicitly represents all cluster edges of $Cl(x)$.
- The list of all free edges incident to x.

Theorem 6 *It is possible to maintain a 3-spanner associated to a $n^{1/2}$-clustering, under the addition of new clusters $C_1, C_2, \ldots, C_h$ as described in Procedure MaximalCluster, in a total worst-case time $O(\sum_{i=1}^{h} \sum_{y \in C_i} |N(y)|)$.*

Proof. Assume that cluster edges, free edges and 3-bridge edges are correct before adding clusters to Γ.

A vertex x having $|FN(x)| \geq 2n^{1/2}$ can be found in constant time, provided that the set of vertices x having $|FN(x)| \geq 2n^{1/2}$ is maintained throughout.

We now determine the cost of updating the spanner for the different classes of spanner edges. Assume that the new cluster $Cl(x)$ is added to Γ:

(i) **cluster edges:** we add all edges (x, y), with $y \in Cl(x)$. This can be done in $O(|Cl(x)|)$ worst-case time;

(ii) **free edges:** vertices in $Cl(x)$ are no longer free. For each vertex $y \in Cl(x)$ we explore vertices in $N(y)$: for each vertex $z \in N(y)$ we decrement $|FN(z)|$, moreover, if z is clustered we remove (z, y) from the set of free edges. This can be done in $O(\sum_{i=1}^{h} \sum_{y \in C_i} |N(y)|)$ worst-case time;

(iii) **3-bridge edges:** each vertex $v \in V \setminus Cl(x)$ must be connected to $Cl(x)$ via a new 3-bridge edge. For each vertex $y \in Cl(x)$ we scan vertices $z \in N(y)$: if z is not already connected to $Cl(x)$ then (y, z) becomes a 3-bridge edge. This can be done in $O(\sum_{i=1}^{h} \sum_{y \in C_i} |N(y)|)$ worst-case time.

All the above operations can be implemented in a total of $O(\sum_{i=1}^{h} \sum_{y \in C_i} |N(y)|)$ worst-case time. □

Theorem 7 *Given a graph G, Procedure MaximalCluster computes in $O(m+n)$ worst-case time a 3-spanner of G having $O(n^{3/2})$ edges.*

Proof. We apply Procedure MaximalCluster starting from $\Gamma = \emptyset$. At the beginning, all vertices are free, there are no cluster edges and 3-bridge edges, and the set of free edges is E.

By Theorems 1, 2, 5, and 6, we can state that a 3-spanner is computed in $O(m+n)$ worst-case time. □

We now show how to deal with edge deletions. When an edge is deleted from the graph we might have to update the clustering and the associated spanner. We first show how the clustering can be updated, and then describe how to update the associated spanner.

A clustering Γ is affected by the deletion of edge $e = (x, y)$ only if e is a cluster edge; w.l.o.g. assume that x is a center and $y \in Cl(x)$. In this case y can no longer be in $Cl(x)$, due to Property 1 of Definition 2. A more substantial change is due to Property 4 of Definition 2, in the case where, after removing y from $Cl(x)$, this set becomes too small to be a cluster: in this case $Cl(x)$ is removed from Γ. In both cases, in order to preserve Property 3 of Definition 2, we must add y (resp., each vertex $v \in Cl(x)$ in the case $Cl(x)$ is removed from Γ) to some other cluster in Γ, whenever possible. If there are no centers in $N(y)$ (resp., in $N(v)$ for any vertex $v \in Cl(x)$) then y becomes a free vertex. The update algorithm is described by Procedure DeleteEdge, listed in Figure 3.

Procedure DeleteEdge
input: graph $G = (V, E)$
$n^{1/2}$-clustering Γ of $G = (V, E)$
a cluster edge $e = (x, y)$, where $x = center(y)$
output: $n^{1/2}$-clustering Γ of $G = (V, E \setminus \{e\})$

```
1.    if |Cl(x)| > n^{1/2}
2.        remove y from Cl(x)
3.        if y is the center of a cluster in Γ
4.            add y to Cl(y)
5.        else if there exists a center c ∈ N(y) in Γ
6.            add y to Cl(c)
7.    else
8.        // vertex x is no longer a center //
9.        remove Cl(x) from Γ
10.       for each v ∈ Cl(x)
11.           if v is the center of a cluster in Γ
12.               add v to Cl(v)
13.           else if there exists a center c ∈ N(v) in Γ
14.               add v to Cl(c)
```

Figure 3: Deleting a cluster edge

We now show how to update the associated spanner, after the deletion of an edge $e = (x, y)$. We distinguish four different cases, depending on the type of edge being deleted:

e **is not in the spanner:** the spanner $G' = (V, E')$ does not change. Since e is not in the spanner, both x and y must be clustered vertices. Neither $FN(x)$ or $FN(y)$ change their size;

e **is a free edge:** at least one among x and y is free. Edge e is removed from the spanner; the sizes of $FN(x)$ and/or $FN(y)$ are decremented accordingly;

e **is a 3-bridge edge:** w.l.o.g. assume that e connects y to $x \in Cl(z)$ (the case where e also connects x to $y \in Cl(w)$ is dealt with analogously): find another edge f connecting y to a vertex $w \in Cl(z)$ (if it exists), and add f to the set of 3-bridge edges;

e **is a cluster edge:** assume that x is a center and $y \in Cl(x)$, and that the clustering is updated according to Procedure DeleteEdge. The spanner is updated as follows.

- If $|Cl(x)|$ remains at least $n^{1/2}$, $Cl(x)$ is still a cluster, and vertex x is still its center:
  - e is no longer a cluster edge;
  - replace each 3-bridge edge (z, y) connecting a vertex z to $Cl(x)$ via y by a new edge, if possible. To this aim, for each vertex $z \in N(y)$, if (z, y) is a 3-bridge edge we remove (z, y) from the set of 3-bridge edges and search for a new 3-bridge edge (z, w), with $w \in Cl(x)$;
  - if y is added to $Cl(c)$ (where possibly $c = y$):
    - (c, y) becomes a cluster edge (provided that $c \neq y$);
    - in case some vertex w was not connected to $Cl(c)$, because there were no edges $(w, z) \in E$ with $z \in Cl(c)$, but $(w, y) \in E$, it is now possible to connect w to $Cl(c)$ via y. This can be done by detecting, for each $w \in N(y)$, whether w is already connected to $Cl(c)$ and, if not, adding edge (w, y) to the set of 3-bridge edges.
  - in case y is now free, for each $z \in N(y)$ we increase $|FN(z)|$ and add (z, y) to the set of free edges.
- If $|Cl(x)|$ drops below $n^{1/2}$, $Cl(x)$ can no longer be a cluster, and thus vertex x can no longer be a center:
  - remove all 3-bridge edges connecting vertices to $Cl(x)$—provided that the same edge does not connect a vertex to any other cluster;
  - for each $v \in Cl(x)$ we do the following:
    - (x, v) is no longer a cluster edge;
    - if v is added to $Cl(c)$ (where possibly $c = v$):
      - (c, v) becomes a cluster edge (provided that $c \neq v$);
      - in case some vertex w was not connected to $Cl(c)$, because there were no edges $(w, z) \in E$ with $z \in Cl(c)$, but $(w, v) \in E$, it is now possible to connect w to $Cl(c)$ via v. As above, this can be done by detecting, for each $w \in N(v)$, whether w is already connected to $Cl(c)$ and, if not, adding edge (w, v) to the set of 3-bridge edges.
    - in case v is now free, for each $z \in N(v)$ we increase $|FN(z)|$ and add (z, v) to the set of free edges.

This restores a 3-spanner associated to the $n^{1/2}$-clustering Γ. After this, in order to obtain a maximal $n^{1/2}$-clustering, we apply Procedure MaximalCluster.

Theorem 8 *Procedure DeleteEdge updates a $n^{1/2}$-clustering and the associated 3-spanner of G under the deletion of edge (x, y) in*

- $O(|N(x)| + |N(y)|)$ *worst-case time, if no cluster is removed from Γ;*
- $O(\sum_{v \in Cl(x)} |N(v)|)$ *worst-case time, if (x, y) is a cluster edge and cluster $Cl(x)$ is removed from Γ.*

Proof. If e is not in the spanner or it is a free edge, the above algorithm requires constant time. In the case (x, y) is a 3-bridge edge we need $O(|N(x)| + |N(y)|)$ worst-case time for exploring $N(x)$ and/or $N(y)$. If (x, y) is a cluster edge, we distinguish two cases: if $Cl(x)$ is still a cluster, we only explore $N(x)$ and $N(y)$. Otherwise, if $Cl(x)$ is destroyed, we explore the neighborhood of all vertices in $Cl(x)$, in $O(\sum_{v \in Cl(x)} |N(v)|)$ worst-case time. □

4.2 Construction and edge deletion in 5-spanners

In order to build and maintain the 5-spanners, we need to maintain an $n^{1/3}$-clustering. The only change in the data structures consists in keeping track of 5-bridge edges instead of 3-bridge edges. More precisely, for each vertex x we maintain:

- The number $|FN(x)|$ of free vertices in $N(x)$.
- A flag indicating whether x is clustered. If x is clustered:
  - a reference to the cluster containing x;
  - a reference to $center(x)$;
  - the list of all 5-bridge edges (y, x) incident to x, connecting any cluster to the cluster containing x.
- A flag indicating whether x is a center. If x is a center:
  - the list of vertices in $Cl(x)$. This list implicitly represents all cluster edges of $Cl(x)$.
- The list of all free edges incident to x.

Besides, for each pair of clusters $Cl(x_i), Cl(x_j)$, we maintain the set of edges $e = (x, y) \in E$ such that $x \in Cl(x_i)$ and $y \in Cl(x_j)$. These sets represent the candidate edges for connecting pairs of clusters; we may assume that the 5-bridge edge in the spanner connecting $Cl(x_i)$ to $Cl(x_j)$ is the first edge in this set.

Theorem 9 *It is possible to maintain a 5-spanner associated to an $n^{1/3}$-clustering, under the addition of new clusters $C_1, C_2, \ldots, C_h$ as described in Procedure MaximalCluster, in a total worst-case time $O(\sum_{i=1}^{h} \sum_{y \in C_i} |N(y)|)$.*

Proof. The proof proceeds as in Theorem 6, with the exception of bridge edges. Any time a new cluster $Cl(x)$ is added to Γ, at most $n^{2/3}$ new sets of candidate 5-bridges, one for each existing cluster, must be created. For each vertex $v \in Cl(x)$ we examine all $y \in N(v)$. If $y \in Cl(z)$ then edge (v, y) is added to the set of candidate bridges for the pair $Cl(x), Cl(z)$. All the updates can still be done in a total of $O(\sum_{i=1}^{h} \sum_{y \in C_i} |N(y)|)$. □

The following theorem (analogous to Theorem 7) easily follows.

Theorem 10 *Given a graph G, Procedure MaximalCluster computes in $O(m + n)$ worst-case time a 5-spanner of G having $O(n^{4/3})$ edges.*

Let us briefly resume how we deal with the deletion of edge e. If e is not in the spanner, or it is a free edge, nothing changes with respect to the case of 3-spanners.

When e is a 5-bridge edge connecting $Cl(x_i)$ to $Cl(x_j)$, we replace it with one of the candidate 5-bridge edges, if any, in the corresponding set. This can be done in constant worst-case time.

If e is a cluster edge, the clustering is updated as in the case of 3-spanner. The only difference with respect to the case of 3-spanners consists in maintaining both the 5-bridge edges and the candidate 5-bridge edges incident to vertices that enter into or exit from a cluster.

Next result, analogous to Theorem 8, thus follows:

Theorem 11 *Procedure DeleteEdge updates an $n^{1/3}$-clustering and the associated 5-spanner of G under the deletion of edge $e = (x, y)$ in*

- *$O(|N(x)| + |N(y)|)$ worst-case time, if no cluster is removed from Γ;*
- *$O(\sum_{v \in Cl(x)} |N(v)|)$ worst-case time, if (x, y) is a cluster edge and cluster $Cl(x)$ is removed from Γ.*

4.3 Amortized complexity of edge deletions

An edge deletion that does not modify the clustering is performed in $O(n)$ worst-case time. If a vertex is removed from a cluster, the spanner is maintained in $O(n)$ worst-case time (by Theorems 8 and 11), possibly plus the time needed to build a new cluster. Since the size of the new cluster is $O(\ell)$, the new cluster can be built in $O(\ell \cdot n)$ time, due to Theorems 6 and 9.

An edge deletion that destroys a cluster is performed in $O(\ell \cdot n)$ worst-case time (by Theorems 8 and 11), possibly plus the time needed to build the new clusters. Again, each new cluster has size $O(\ell)$, and by Theorems 6 and 9 the time needed is $O(\ell \cdot n)$ for each new cluster. Hence, for each edge deletion we need a total of $O(n)$ time plus $O(\ell \cdot n)$ time for each cluster that appears or disappears from the clustering.

In order to bound the number of clusters that appear or disappear during a sequence of edge deletions, we consider how the cluster sizes can be affected by edge deletions. If the edge deletion does not destroy any cluster, then the size of at most one cluster is decreased by one (this happens when a cluster edge is deleted). Otherwise, only one cluster may be destroyed, and the size of the other clusters does not decrease.

During a sequence of edge deletions, the set of destroyed clusters consists at most of the initial clusters, plus some of the clusters created during the sequence of edge deletions. The initial clusters are at most n/ℓ. The initial size of a cluster C created during the sequence is at least $2 \cdot \ell$, and C is destroyed only if its size decreases to less than ℓ during the update sequence. By the above arguments at least ℓ deletions are needed in order to shrink C from its initial size (at least $2 \cdot \ell$) to ℓ. In summary, if the update sequence has length σ, at most $n/\ell + \sigma/\ell$ clusters may be destroyed overall.

The number of clusters created during the sequence is at most the number of clusters at the end of the sequence plus the number of destroyed clusters, that is at most $2 \cdot n/\ell + \sigma/\ell$. By Theorems 8 and 11, the total cost over the sequence is thus $O(\sigma \cdot n + (n/\ell + \sigma/\ell) \cdot \ell \cdot n) = O(\sigma \cdot n + n^2)$. Hence, we can state the following:

Theorem 12 *A 3-spanner of an unweighted graph can be maintained in $O(\sigma \cdot n + n^2)$ total time over a sequence of σ edge deletions. The spanner has $O(n^{3/2})$ edges. This gives $O(n)$ amortized time per operation over a sequence of $\Omega(n)$ edge deletions.*

Theorem 13 *A 5-spanner of an unweighted graph can be maintained in $O(\sigma \cdot n + n^2)$ total time over a sequence of σ edge deletions. The spanner has $O(n^{4/3})$ edges. This gives $O(n)$ amortized time per operation over a sequence of $\Omega(n)$ edge deletions.*

5 Fully dynamic 3-spanners and 5-spanners for unweighted graphs

To make the decremental algorithms of Section 4 fully dynamic, we deal with edge insertions in a lazy fashion. Inserted edges are kept in a set E'', and our 3-spanner consists of the edges induced by the clustering (see Definition 3) plus the edges in E''. When inserting an edge, we do not

update the clustering and the associated spanner. Only when the size of E'' exceeds the size of the spanner, i.e., $n^{3/2}$ or $n^{4/3}$, a new clustering and the associated spanner are built from scratch using Procedure MaximalCluster starting from the empty clustering, and E'' is set to the empty set. This gives the following theorem:

Theorem 14 *A 3-spanner or a 5-spanner of an unweighted graph can be maintained in $O(\sigma \cdot n + n^2)$ total time over a sequence of σ intermixed edge insertions and edge deletions. This gives $O(n)$ amortized time per operation on a sequence of $\Omega(n)$ edge insertions and edge deletions.*

Proof. If the sequence contains less than $n^{3/2}$ (resp., $n^{4/3}$) edge insertions, then the spanner is never rebuilt from scratch, and the theorem derives from Theorems 12 and 13. Otherwise, we must rebuild from scratch the spanner, taking $O(n^2)$ worst-case time (see Theorems 7 and 10), but this cost can be amortized over a sequence of length $\Omega(n^{3/2})$ (resp., $\Omega(n^{4/3})$), giving an amortized cost of $O(n^{1/2})$ (resp., $\Omega(n^{2/3})$) per operation. Hence, the amortized cost is dominated by the cost of edge deletions, which is $O(n)$ by Theorems 12 and 13. □

6 Fully dynamic t-spanners for general edge costs

In this section we present extensions of algorithm for unweighted graphs to graphs whose edge costs may assume different values. For the sake of presentation, we first show how to deal with d different edge cost values, and then we apply the same technique to deal with general positive edge costs. The extension is described in the case of 3-spanners, its application to 5-spanners being trivial.

In the case of d different edge cost values[1] $c_1, c_2, \ldots, c_d$, with $c_i \geq 0$ for $i = 1, \ldots, d$, we maintain a 3-spanner as the union of $S_1, S_2, \ldots, S_d$, where each S_i is a 3-spanner of the subgraph containing all the edges with cost c_i.

A key observation is the following:

Lemma 2 *Given ℓ subgraphs $G_1 = (V, E_1)$, $G_2 = (V, E_2)$, ..., $G_\ell = (V, E_\ell)$ of a graph $G = (V, E)$ such that $\bigcup_{i=1}^{\ell} E_i = E$, if S_1, S_2, ..., S_ℓ are respectively t-spanners of G_1, G_2, ..., G_ℓ then $S = \bigcup_{i=1}^{\ell} S_i$ is a t-spanner of G.*

Proof. Given any edge $e = (u, v) \in E$, there exists at least one i such that $e \in E_i$. Since S_i is a t-spanner for G_i, by condition (2) in Lemma 1, we have $\mathrm{dist}_{S_i}(u, v) \leq c(e)$. But $S_i \subseteq S$, thus $\mathrm{dist}_S(u, v) \leq \mathrm{dist}_{S_i}(u, v) \leq c(e)$. So, $S = \bigcup_{i=1}^{\ell} S_i$ fulfills condition (2) of Lemma 1, and thus S is a t-spanner of G. □

Based on Lemma 2, we partition our edge set E into d disjoint subsets $E_1, E_2, \ldots, E_d$, where E_i contains all edges $e \in E$ such that $c(e) = c_i$, and apply the same algorithms described in Sections 4 and 5 to each subgraph. Each edge insertion or edge deletion concerns a single spanner S_i.

Let σ be the total number of updates in the sequence, and let σ_i be the number of updates concerning edges with cost c_i. Clearly, $\sigma = \sum_{i=1}^{d} \sigma_i$. For any i, $1 \leq i \leq d$, the fully dynamic maintenance of spanner S_i requires time $O(\sigma_i \cdot n + n^2)$, by Theorem 14. The total running time of the algorithm is therefore $\sum_{i=1}^{d} O\left(\sigma_i \cdot n + n^2\right) = O(\sigma \cdot n + d \cdot n^2)$, thus yielding the following:

Theorem 15 *A 3-spanner (resp., a 5-spanner) of a graph with d different edge costs can be maintained in $O(n)$ amortized time per operation over a sequence of $\Omega(d \cdot n)$ edge insertions and deletions. The spanner has $O(d \cdot n^{3/2})$ (resp., $O(d \cdot n^{4/3})$) edges.*

[1] Note that, in the case of a constant number d of edge cost values the result holds also in presence of edges with cost 0.

The same algorithms can be applied to graphs with general positive edge costs, where the number of different costs can be $\Theta(n^2)$. Let the edge costs in graph $G = (V, E)$ be real numbers in the interval $[1, C]$. We define a new graph G', in which each cost value is rounded to its nearest smaller power of r, with $r > 1$: for each edge $e \in E$ there is an edge in G' with $c'(e) = r^{\lfloor \log_r c(e) \rfloor}$. Costs $c'(\cdot)$ may assume at most $\lceil \log_r C \rceil$ different values.

The following relations hold between edge costs and distances in G and G':

- $c'(e) \leq c(e) < r \cdot c'(e)$;
- $\text{dist}_{S'}(x, y) \leq \text{dist}_S(x, y) < r \cdot \text{dist}_{S'}(x, y)$.

If S' is a t-spanner of G', with respect to edge costs $c'(\cdot)$, and S contains the same edges of S' but with the original costs $c(\cdot)$, then S is a $(t \cdot r)$-spanner of G (with respect to costs $c(\cdot)$). In fact, the above relationships and Lemma 1 allow us to write that for any edge $e = (x, y)$ in E:

$$\text{dist}_S(x, y) < r \cdot \text{dist}_{S'}(x, y) \leq t \cdot r \cdot c'(e) \leq t \cdot r \cdot c(e) \ .$$

Thus, if we replace edge costs by integer powers of r, maintain the 3-spanner on $\log_r C$ edge costs as in Theorem 15, and look back to the original edge costs, we can maintain a $(3 \cdot r)$-spanner of G having $O(n^{3/2} \cdot \log_r C)$ edges . When $r = \frac{t}{3}$, we obtain the following theorems:

Theorem 16 *For any $t > 3$, a t-spanner of a graph with real-valued edge costs in $[1, C]$ can be maintained in $O(n)$ amortized time per operation over a sequence of $\Omega(n \cdot \log_{t/3} C)$ edge insertions and edge deletions. The spanner has $O(n^{3/2} \cdot \log_{t/3} C)$ edges.*

Theorem 17 *For any $t > 5$, a t-spanner of a graph with real-valued edge costs in $[1, C]$ can be maintained in $O(n)$ amortized time per operation over a sequence of $\Omega(n \cdot \log_{t/5} C)$ edge insertions and edge deletions. The spanner has $O(n^{4/3} \cdot \log_{t/5} C)$ edges.*

7 Conclusions

We have presented dynamic algorithms for maintaining 3-spanners and 5-spanners of unweighted graphs under a sequence of updates. The same techniques have been extended to deal with 3-spanners and 5-spanners of graphs with d different edge weights and t-spanners of general weighted graphs for small t. All our algorithms are deterministic and are substantially faster than recomputing a spanner from scratch after each update by static algorithms.

While the size of our 3-spanner and 5-spanner is optimal to within constant factors for constant d, this is not always the case for larger d and for $t > 3$. The fully dynamic maintenance of sparser spanners for general weighted graphs seems thus worth of further investigation.

References

[1] I. Althofer, G. Das, D. P. Dobkin, D. Joseph, and J. Soares. On sparse spanners of weighted graphs. *GEOMETRY: Discrete & Computational Geometry*, 9:81–100, 1993.

[2] B. Awerbuch. Complexity of network synchronization. *Journal of the ACM*, 32(4):804–823, October 1985.

[3] H.-J. Bandelt and A. Dress. Reconstructing the shape of a tree from observed dissimilarity data. *Adv. Appl. Math.*, 7:309–343, 1986.

[4] S. Baswana and S. Sen. A simple linear time algorithm for computing $(2k-1)$-spanner of $O(n^{1+1/k})$ size for weighted graphs. In *30th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 2719 of *LNCS*, pages 384–396, Berlin, 2003. Springer.

[5] L. Cai. NP-completeness of minimum spanner problems. *Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science*, 48(2):187–194, 1994.

[6] L. Cai and J. M. Keil. Degree-bounded spanners. *Parallel Processing Letters*, 3:457–468, 1993.

[7] L. P. Chew. There are planar graphs almost as good as the complete graph. *Journal of Computer and System Sciences*, 39(2):205–219, October 1989.

[8] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms.* MIT Press, 2001.

[9] G. Das and D. Joseph. Which triangulations approximate the complete graph? In H. Djidjev, editor, *Proceedings of the International Symposium on Optimal Algorithms*, volume 401 of *LNCS*, pages 168–192, Berlin, May 29–June 2 1989. Springer.

[10] D. Dobkin, S. J. Friedman, and K. J. Supowit. Delaunay graphs are almost as good as complete graphs. *Discrete Comput. Geom.*, 5:399–407, 1990. See also *28th Symp. Found. Comp. Sci.*, 1987, pp. 20–26.

[11] A. L. Liestman and T. Shermer. Additive graph spanners. *NETWORKS: Networks: An International Journal*, 23:343–364, 1993.

[12] A. L. Liestman and T. Shermer. Grid spanners. *NETWORKS: Networks: An International Journal*, 23:122–133, 1993.

[13] D. Peleg and A. Shäffer. Graph spanners. *J. of Graph Theory*, 13:99–116, 1989.

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

[15] D. Richards and A. L. Liestman. Degree-constrained pyramid spanners. *JPDC: Journal of Parallel and Distributed Computing*, 25:1–6, 1995.

[16] L. Roditty and U. Zwick. Dynamic approximate all-pairs shortest paths in undirected graphs. In IEEE, editor, *Proceedings of the 45th IEEE Conference on Foundations of Computer Science: Rome, Italy, October 17–19, 2004*, pages 499–508, 2004.

[17] R. Wenger. Extremal graphs with no C^4's, C^6's, or C^{10}'s. *J. Combin. Theory Ser. B*, 52:113–116, 1991.

[18] U. Zwick. Personal communication.