Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 10, no. 2, pp. 365–385 (2006)

Small Stretch Spanners on Dynamic Graphs

Giorgio Ausiello

Dipartimento di Informatica e Sistemistica
Sapienza Università di Roma
http://www.dis.uniroma1.it/∼ausiello ausiello@dis.uniroma1.it

Paolo G. Franciosa

Dipartimento di Statistica, Probabilità e Statistiche Applicate
Sapienza Università di Roma
http://www.dis.uniroma1.it/∼pgf paolo.franciosa@uniroma1.it

Giuseppe F. Italiano

Dipartimento di Informatica, Sistemi e Produzione
Università di Roma "Tor Vergata"
http://www.disp.uniroma2.it/users/italiano italiano@disp.uniroma2.it

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-spanner or a 5-spanner under insertions and deletions of edges; on a graph with n vertices each operation is performed in $O(\Delta)$ amortized time over a sequence of $\Omega(n)$ updates, where Δ is the maximum degree of the original graph. 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 within the same amortized time bounds 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.

Article Type	Communicated by	Submitted	Revised
regular paper	G. Liotta	November 2005	June 2006

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

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. A deterministic algorithm for computing a t-spanner of size $O\left(t \cdot n^{1+\frac{2}{t+1}}\right)$ of a weighted graph with n vertices and m edges has been given in [1]; the best known implementation of this algorithm, given in [18], has running time $O(n^{2+2/(t+1)})$. Recently, a randomized algorithm running in $O(m+n)$ time has been given by Baswana and Sen [4]; a derandomization of this algorithm, still running in $O(m+n)$ worst case time, has been proposed in [16].

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 of memory 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. This means that algorithms for computing exact distances or shortest paths could be run on the spanner in main memory, giving approximate distances for the original graph. Applications related to computing distances on very large dense graphs arise for instance in MultiProtocol Label Switching (MPLS) networks, where a connection represents a tunnel between a pair of devices that crosses many physical connections. In this scenario, MPLS networks can be very dense, even close to a complete graph, although the original graph representing physical connections is sparse.

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 graph under edge insertions and/or deletions is known, neither for the unweighted case, and 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 [17] 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 much sparser data structure that still maintains approximate 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 $O(\Delta)$ amortized time per operation, where Δ is the maximum degree of the original graph. The maintained 3-spanner has $O(n^{3/2})$ edges, while the 5-spanner has $O(n^{4/3})$ edges.

Wenger [19] 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. Over a sequence of $\Omega(d \cdot n)$ intermixed edge insertions and deletions the amortized time per operation is still $O(\Delta)$. 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 $O(\Delta)$ 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 $O(\Delta)$ amortized time per operation over a sequence of $\Omega(n \cdot \log_{t/5} C)$ edge insertions and edge deletions.

Our algorithms require $O(m)$ worst-case space (assuming that the original graph is larger than the spanner, i.e., $m = \Omega(n^{3/2})$ for the 3-spanner and $m = \Omega(n^{4/3})$ for the 5-spanner), are deterministic and are substantially faster than recomputing a spanner from scratch. To achieve our results, we derive a deterministic version of the randomized clustering technique of Baswana and Sen [4], and find how to update the clustering under the deletion of edges. 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 deterministic 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 over 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 presented 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 the 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). The *degree* $\delta(x)$ of vertex x is the number of edges incident on x; the maximum degree of vertices in V is denoted by Δ. 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 $0 < i \le \ell$. We say that each edge (v_{i-1}, v_i) is in path π, for $0 < i \le \ell$. The *length of a path* π is 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 $c(\pi) = \mathrm{dist}_G(u,v)$. 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 that 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 $t \ge 1$. A t-*spanner *of G is a graph $S = (V,E')$ with $E' \subseteq E$ such that the following holds:*

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

The following lemma is due to 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:*

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

2.2 Clustering

Definition 2 *Let $x_1, x_2, \ldots, x_k$, with $k \ge 1$, 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 $\ell \ge 1$, Γ is an ℓ-*clustering *of $G = (V,E)$ if the following properties hold:*

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

Each 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 a 5-clustering is shown in 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 $FN(v) = N(v) \setminus \left(\bigcup_{i=1}^{k} Cl(x_i)\right)$.

Due to Properties 2 and 4 in Definition 2, 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.

3 Clusterings and spanners

In this section we show how small stretch spanners of unweighted graphs can be produced from an appropriate 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 an ℓ-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 $x \in Cl(x_i)$, if one exists. We say that edge (x, y)* connects *vertex y to cluster $Cl(x_i)$.*

Theorem 1 *Given a graph $G = (V, E)$ and an ℓ-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 from a to b 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';

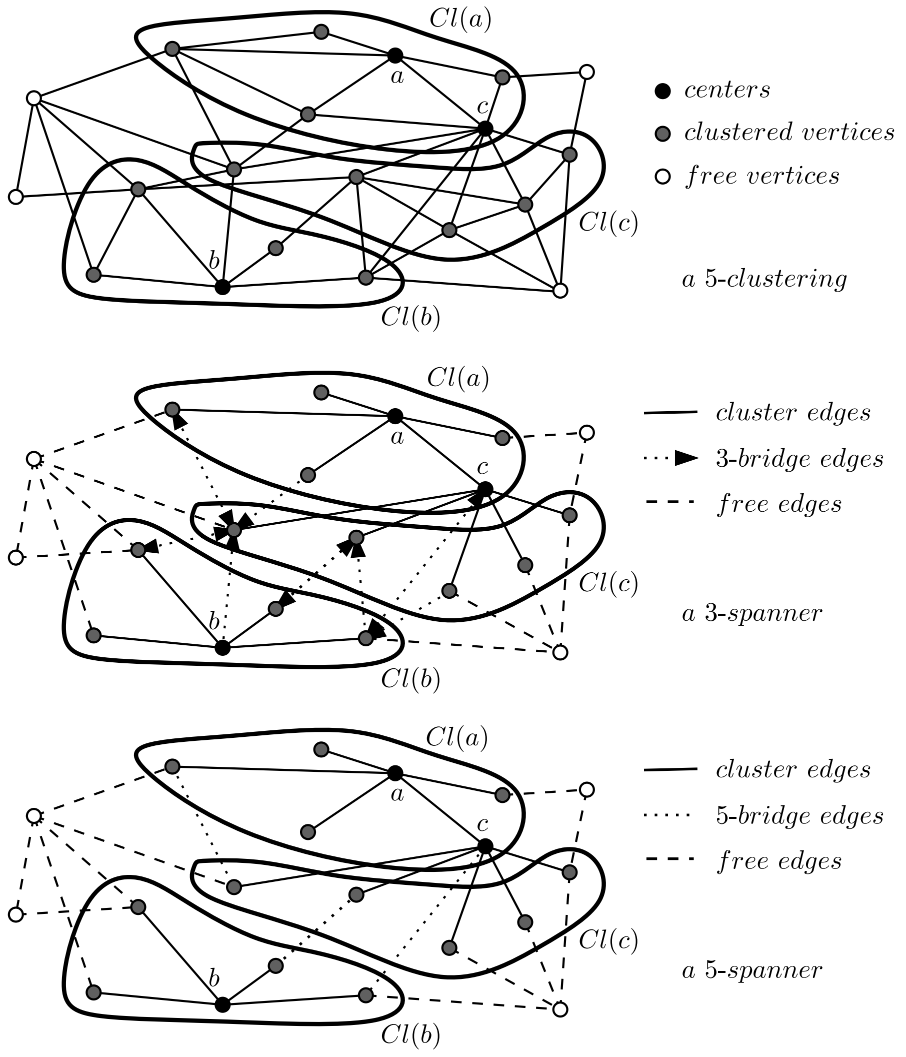

Figure 1: a 5-clustering of a graph and the associated 3-spanner and 5-spanner. In the 3-spanner, 3-bridge edges are represented by arrows from a vertex to a cluster.

  - 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 ℓ-clustering Γ as a *3-spanner associated with* Γ. An example of 3-spanner associated with a 5-clustering of a graph is shown in Figure 1.

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

Corollary 1 *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 $\langle x, Cl(x_i)\rangle$, where $x \in V$ and $Cl(x_i) \in \Gamma$: 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. Let us partition the set of free edges into two sets: FF containing edges with two free endpoints, and FC containing edges between a free vertex and a clustered vertex. Since Γ is maximal, each free vertex has at most $2 \cdot n^{1/2}$ free adjacent vertices, and thus $|FF| \leq n \cdot 2n^{1/2}$.

We count edges in FC by looking at each clustered vertex v: since Γ is maximal, there can be at most $2 \cdot n^{1/2}$ free edges incident to v, and thus $|FC| \leq n \cdot 2n^{1/2}$. □

3.2 Clusterings for 5-spanners

Definition 4 *Given an ℓ-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 $y \in Cl(x_j)$, if one exists. 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 2 *Given a graph $G = (V, E)$ and an ℓ-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'.

$\square$

Due to Theorem 2, we will refer to a subgraph of G 5-compatible with an ℓ-clustering Γ as a *5-spanner associated with* Γ. If we choose $\ell = n^{1/3}$ and the $n^{1/3}$-clustering is maximal, we can prove that any associated 5-spanner is sparse:

Corollary 2 *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. Let us partition the set of free edges into two sets: FF containing edges with two free endpoints, and FC containing edges between a free vertex and a clustered vertex. Since Γ is maximal, each free vertex has at most $2 \cdot n^{1/3}$ free adjacent vertices, and thus $|FF| \leq n \cdot 2n^{1/3}$.

We count edges in FC by looking at each clustered vertex v: since Γ is maximal, there can be at most $2 \cdot n^{1/3}$ free edges incident to v, and thus $|FC| \leq n \cdot 2n^{1/3}$. $\square$

4 Construction and edge deletion in 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.

In this section we first show how to build a maximal ℓ-clustering by means of Procedure MaximalCluster (Figure 2). Starting from any ℓ-clustering, Procedure MaximalCluster obtains a maximal ℓ-clustering with a simple greedy approach. When the starting clustering is empty, the same procedure may be used to compute a maximal ℓ-clustering from scratch.

After that, we discuss how to update a clustering under edge deletions: it may happen that, when an edge is deleted, the updated clustering is no longer maximal. In this case procedure MaximalCluster is applied again. Finally, we show how to modify 3-spanners and 5-spanners when the clustering is updated under an edge deletion.

```
Procedure MaximalCluster
input: graph $G = (V, E)$
       $\ell$-clustering $\Gamma$
output: maximal $\ell$-clustering $\Gamma$
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 $\Gamma$
```

Figure 2: Procedure MaximalCluster.

Lemma 2 *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 now show 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 the definition of the 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 Section 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 Section 4.2, we show how to maintain a 5-spanner during a sequence of edge deletions.

4.1 Construction and edge deletion in 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 $\ell = n^{1/2}$.

We maintain the following simple data structures, which represent the current spanner plus some auxiliary information.

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 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 edges incident to x.
- The list of all free edges incident to x.

Moreover, we maintain:

- A doubly linked list containing all vertices x having $|FN(x)| \geq 2 \cdot n^{1/2}$.
- A matrix of candidate 3-bridge edges: more precisely, for each pair (x, C), where C is a cluster and $x \notin C$ is a vertex, we maintain a list $CB(x, C)$ containing all edges (x, y) in the graph, with $y \in C$. The first edge in $CB(x, C)$ is the 3-bridge edge connecting x to C in the spanner.

  We associate to each cluster C a *timestamp* time(C), denoting the time at which cluster C has been created. A timestamp time$(CB(x, C))$ is also associated to each list $CB(x, C)$, denoting the time at which $CB(x, C)$ has been created.

Each cluster is identified by an integer label in the range $[1, \lceil n^{1/2} \rceil]$, which is used as an index to access the matrix of candidate bridge lists. Cluster labels are reused: when a cluster is created, it gets one of the free labels, and when a cluster is destroyed, its label becomes free.

When a cluster C having label y is destroyed, for the sake of efficiency, we do not explicitly destroy its lists $CB(x, y)$. When a new cluster C' is created, and gets y as its label, we simply create the new lists $CB(x, y)$ with the appropriate timestamp, possibly overwriting some previous list.

Let C' be the current cluster having label y: if $\text{time}(CB(x, y)) < \text{time}(C')$, then the list $CB(x, y)$ refers to a previously destroyed cluster C, and thus it is considered empty; otherwise, if $\text{time}(CB(x, y)) \geq \text{time}(C')$, then the list $CB(x, y)$ refers to C' and thus it is considered valid. In the sequel, by *empty* list we mean a list either with no entries or with mismatching timestamps.

Lemma 3 *It is possible to maintain a 3-spanner associated with 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 of $O(\sum_{i=1}^{h} \sum_{y \in C_i} |N(y)|)$ worst-case time.*

Proof: Assume that cluster edges, free edges and candidate 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. Any time $|FN(v)|$ crosses the value $2n^{1/2}$, v is inserted into/deleted from the doubly linked list.

We now determine the cost of updating the spanner for the different classes of spanner edges. Assume that a new cluster $Cl(x) = FN(x)$ 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 by scanning the list of free edges incident to x. These edges are removed from the list of free edges;

(ii) **free edges:** vertices in $Cl(x)$ are no longer free. For each vertex $y \in Cl(x) \setminus \{x\}$ 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 by scanning the list of free edges incident to each $y \in Cl(x) \setminus \{x\}$;

(iii) **(candidate) 3-bridge edges:** the lists of candidate 3-bridge edges can be updated by scanning the list of edges incident to each $y \in Cl(x)$. Since cluster $Cl(x)$ has been added to Γ, a new list $CB(z, Cl(x))$ must be created for each $z \in N(y)$, $y \in Cl(x)$. The first edge in each non-empty list is a 3-bridge edge. This can be done in $O(\sum_{y \in Cl(x)} |N(y)|)$ worst-case time by scanning the lists of edges incident to each vertex $y \in Cl(x)$. Note that, if we had to create explicitly an empty list $CB(v, Cl(x))$ for each $v \notin N(y)$ the cost would have been $\Omega(n)$; this is avoided by the timestamping technique described above.

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

Lemma 4 *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 Theorem 1, Corollary 1, and Lemmas 2 and 3, 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; without loss of generality 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, and y is removed from $Cl(x)$. A more substantial change is required to preserve 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 DeleteClusterEdge, listed in Figure 3.

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. If x and y belong to different clusters C_x and C_y, then e is removed from the lists of candidate 3-bridge edges $CB(x, C_y)$ and $CB(y, C_x)$.

e **is a free edge:** at least one among x and y is free. Edge e is removed from the lists of free edges incident to x and y. The sizes of $FN(x)$ and/or $FN(y)$ are decremented accordingly. If either x or y is clustered, then e is removed from the appropriate list of candidate 3-bridge edges.

e **is a 3-bridge edge:** without loss of generality, we 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): another edge f connecting y to $Cl(z)$ (if it exists) can be found in $CB(y, Cl(z))$;

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 DeleteClusterEdge. The spanner is updated as follows.

```
Procedure DeleteClusterEdge
input: graph $G = (V, E)$
       $n^{1/2}$-clustering $\Gamma$ of $G = (V, E)$
       a cluster edge $e = (x, y)$, where $x = center(y)$
output: $n^{1/2}$-clustering $\Gamma$ 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 $\Gamma$
4.          add $y$ to $Cl(y)$
5.      else if there exists a center $c \in N(y)$ in $\Gamma$
6.          add $y$ to $Cl(c)$
7.  else
8.      // vertex $x$ is no longer a center //
9.      remove $Cl(x)$ from $\Gamma$
10.     for each $v \in Cl(x)$
11.         if $v$ is the center of a cluster in $\Gamma$
12.             add $v$ to $Cl(v)$
13.         else if there exists a center $c \in N(v)$ in $\Gamma$
14.             add $v$ to $Cl(c)$
```

Figure 3: Deleting a cluster edge

- 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, and y is no longer in $Cl(x)$. We check whether there are candidate 3-bridge edges (y, z), with $z \in Cl(x)$, by scanning the list of edges incident to y and insert them into $CB(y, Cl(x))$.
  - replace each 3-bridge edge (z, y) connecting a vertex z to $Cl(x)$ via y by a new bridge, if one exists. To this aim, for each vertex $z \in N(y)$, we remove (z, y) from $CB(z, Cl(x)))$. The next edge in $CB(z, Cl(x)))$, if it is not empty, is the new 3-bridge edge;
  - if y is added to $Cl(c)$ (where possibly $c = y$):
    * (c, y) becomes a cluster edge (provided that $c \neq y$);
    * each edge (y, z) is examined and, if z belongs to some cluster, (y, z) is inserted into the appropriate candidate 3-bridge edges list;
    * for each $w \in N(y)$, we add edge (w, y) to $CB(w, Cl(c))$. If this list was empty (i.e., w was not connected to $Cl(c)$, because there were no edges $(w, z) \in E$ with $z \in Cl(c)$), edge (w, y) becomes the new 3-bridge edge connecting w to $Cl(c)$.
  - 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:
    - ∗ we remove edge (x, v) from the set of cluster edges;
    - ∗ if v is added to $Cl(c)$ (where possibly $c = v$):
      - · (c, v) becomes a cluster edge (provided that $c \neq v$);
      - · for each $w \in N(v)$, we add edge (w, v) to $CB(w, Cl(c))$. If this list was empty — i.e., w was not connected to $Cl(c)$, because there were no edges $(w, z) \in E$ with $z \in Cl(c)$, edge (w, v) becomes the new 3-bridge edge connecting v to $Cl(c)$.
    - ∗ 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.

Lemma 5 *A $n^{1/2}$-clustering of G is updated by Procedure DeleteClusterEdge under the deletion of edge (x, y), and the associated 3-spanner is updated accordingly, 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. □

We remark that Procedure DeleteClusterEdge does not produce a maximal $n^{1/2}$-clustering. To achieve this, Procedure MaximalCluster is applied as a final step.

4.2 Construction and edge deletion in 5-spanners

In order to build and maintain 5-spanners, we need to maintain a $n^{1/3}$-clustering. With respect to Section 4.1, 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 edges incident to x.
- The list of all free edges incident to x.

Moreover, we maintain:

- A doubly linked list containing all vertices x having $|FN(x)| \geq 2 \cdot n^{1/3}$.
- A matrix of candidate 5-bridge edges: more precisely, for each pair of clusters (C_i, C_j), we maintain a list $CB(C_i, C_j)$ containing all edges (x, y) in the graph, with $x \in C_i$ and $y \in C_j$. The first edge in $CB(C_i, C_j)$ is the 5-bridge edge connecting C_i to C_j in the spanner.

  We associate to each cluster C a *timestamp* time(C), denoting the time at which cluster C has been created. A timestamp time$(CB(C_i, C_j))$ is also associated to each list $CB(C_i, C_j)$, denoting the time at which $CB(C_i, C_j)$ has been created.

Each cluster is identified by an integer label in the range $[1, \lceil n^{1/3} \rceil]$, which is used as an index to access the matrix of candidate bridge lists. Cluster labels and timestamps are used exactly as in the case of 3-spanners.

Lemma 6 *It is possible to maintain a 5-spanner associated with a $n^{1/3}$-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 Lemma 3, 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 $O(\sum_{i=1}^{h} \sum_{y \in C_i} |N(y)|)$. $\square$

The following lemma (analogous to Lemma 4) easily follows.

Lemma 7 *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 summarize 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.

The next result, analogous to Lemma 5, thus follows:

Lemma 8 *A $n^{1/3}$-clustering of G is updated by Procedure DeleteClusterEdge under the deletion of edge (x, y), and the associated 5-spanner is updated accordingly, 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 $O(\Delta)$ worst-case time. If no cluster is destroyed by the update, the spanner is maintained in $O(\Delta)$ worst-case time (by Lemmas 5 and 8), plus possibly the time needed to build a new cluster. The new cluster C can be built in $O(\sum_{y \in C} \delta(y))$ time, due to Lemmas 3 and 6.

An edge deletion that destroys cluster C is performed in $O(\sum_{y \in C} \delta(y))$ worst-case time (by Lemmas 5 and 8), plus possibly the time needed to build the new clusters. Since a cluster that is destroyed after an edge deletion has exactly ℓ vertices, a new cluster centered in vertex v can contain at most those ℓ vertices plus the vertices in $FN(v)$, that were no more than $2 \cdot \ell - 1$. Thus, each new cluster has size $O(\ell)$, and by Lemmas 3 and 6 the time needed is $O(\ell \cdot \Delta)$ for each new cluster. Hence, for each edge deletion we need a total of $O(\Delta)$ time plus $O(\ell \cdot \Delta)$ 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 Lemmas 5 and 8, the total cost over the sequence is thus $O(\sigma \cdot \Delta + (n/\ell + \sigma/\ell) \cdot \ell \cdot \Delta) = O((n + \sigma) \cdot \Delta)$. Hence, we can state the following:

Theorem 3 *A 3-spanner (resp., a 5-spanner) of an unweighted graph can be maintained in $O((n+\sigma)\cdot\Delta)$ total time over a sequence of σ edge deletions. The spanner has $O(n^{3/2})$ edges (resp., $O(n^{4/3})$ edges). This gives $O(\Delta)$ 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 spanners consists of the edges induced by the clustering (see Definitions 3 and 4) 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 4 *A 3-spanner and a 5-spanner of an unweighted graph can be maintained in $O((n+\sigma)\cdot\Delta)$ total time over a sequence of σ intermixed edge insertions and edge deletions. This gives $O(\Delta)$ amortized time per operation over a sequence of $\Omega(n)$ edge insertions and edge deletions.*

Proof: If the sequence contains less than $n^{3/2}$ edge insertions (resp., $n^{4/3}$ for the 5-spanner), then the spanner is never rebuilt from scratch, and the theorem derives from Theorem 3. Otherwise, we must rebuild from scratch the spanner, taking $O(m + n)$ worst-case time (see Lemmas 4 and 7), but this cost can be amortized over a sequence of length $\Omega(n^{3/2})$ (resp., $\Omega(n^{4/3})$ for the 5-spanner), giving an amortized cost of $O(m/n^{3/2})$ per operation (resp., $O(m/n^{4/3})$ for the 5-spanner). Since $\Delta \geq m/n \geq m/n^{3/2}$ (resp., $\Delta \geq m/n \geq m/n^{4/3}$), the amortized time per operation is dominated by edge deletions (see Theorem 3), giving the thesis. □

6 Fully dynamic t-spanners for general edge costs

In this section we present extensions of the 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 similar.

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 9 *Given ℓ subgraphs $G_1 = (V, E_1)$, $G_2 = (V, E_2)$, $\ldots$, $G_\ell = (V, E_\ell)$ of a graph $G = (V, E)$ such that $\bigcup_{i=1}^{\ell} E_i = E$, if S_1, S_2, $\ldots$, S_ℓ are respectively t-spanners of G_1, G_2, $\ldots$, 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. $\square$

Based on Lemma 9, 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 3-spanner S_i requires time $O((n + \sigma_i) \cdot \Delta + m \cdot \sigma_i / n^{3/2})$, by Theorem 4. The total running time of the algorithm is therefore $\sum_{i=1}^{d} O\left((n + \sigma_i) \cdot \Delta + m \cdot \sigma_i / n^{3/2}\right) = O((d \cdot n + \sigma) \cdot \Delta)$, thus yielding the following:

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

The same algorithms can be applied to graphs with general positive edge costs, where the number of different costs can be $\Theta(m)$. Let the edge costs in graph $G = (V, E)$ be real numbers in the interval $[1, C]$. We choose a real value $r > 1$, and we define a new graph G', in which each cost value is rounded to

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

its nearest smaller power of r: 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':

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

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:

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

Thus, if we replace edge costs by integer powers of r, maintain the spanner on $\log_r C$ edge costs as in Theorem 5, and look back to the original edge costs, we can maintain a $(3 \cdot r)$-spanner (resp. a $(5 \cdot r)$-spanner) of G having $O(n^{3/2} \cdot \log_r C)$ edges (resp. $O(n^{4/3} \cdot \log_r C)$ edges). When $r = \frac{t}{3}$ (resp. $r = \frac{t}{5}$), we obtain the following theorem:

Theorem 6 *For any $t > 3$ (resp., $t > 5$), a t-spanner of a graph with real-valued edge costs in $[1, C]$ can be maintained in $O(\Delta)$ amortized time per operation over a sequence of $\Omega(n \cdot \log_{t/3} C)$ (resp., $\Omega(n \cdot \log_{t/5} C)$) edge insertions and edge deletions. The spanner has $O(n^{3/2} \cdot \log_{t/3} C)$ (resp., $O(n^{4/3} \cdot \log_{t/5} C)$) edges.*

7 Conclusions

We have presented dynamic algorithms for maintaining both 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. Unfortunately, our technique does not seem easy to extend to stretch factors other than 3 or 5.

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

Acknowledgements

We are grateful to the anonymous referees for their careful reading and precious suggestions.

References

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

[2] B. Awerbuch. Complexity of network synchronization. *Journal of the ACM*, 32(4):804–823, Oct. 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, Oct. 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 & Computational Geometry*, 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: An International Journal*, 23:343–364, 1993.

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

[13] D. Peleg and A. Shäffer. Graph spanners. *Journal 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, Aug. 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, M. Thorup, and U. Zwick. Deterministic constructions of approximate distance oracles and spanners. In *Proceedings of the The 32nd International Colloquium on Automata, Languages and Programming: Lisbon, Portugal, July 11–15, 2005*, pages 261–272, 2005.

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

[18] L. Roditty and U. Zwick. On dynamic shortest paths problems. In *Proceedings of the 12th Annual European Symposium on Algorithms (ESA-04)*, pages 580–591, 2004.

[19] R. Wenger. Extremal graphs with no C^4's, C^6's, or C^{10}'s. *Journal of Combinatorial Theory Series B*, 52:113–116, 1991.