Alcom-FT Technical Report Series
ALCOMFT-TR-03-23

Fully dynamic maintenance of t-spanners on general graphs*

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

April 4, 2003

Abstract

We give a deterministic algorithm for maintaining a t-spanner of a general weighted undirected graph under a sequence of edge insertions and deletions. In worst-case sequences our algorithm requires $\widetilde{O}\left(t \cdot n^2 \cdot \log C\right)$ [1] amortized time per update, where n is the number of vertices and edge costs are reals in $[1, C]$. The spanner has $O\left(t \cdot n^{1+\frac{4}{t+2}} \cdot \log C\right)$ edges. For random update sequences, the average cost per update is improved to $\widetilde{O}\left(t \cdot n^{1+\frac{4}{t+2}} \cdot \log C\right)$.

1 Introduction

Graph spanners arise in many applications, including communication networks, computational biology, computational geometry, distributed computing, and robotics ([1, 2, 3, 8, 9, 10, 12, 14, 17, 18, 19, 20, 21]).

Intuitively, a spanner is a subgraph of a given graph that preserves approximate distances between all pairs of vertices. More formally, 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.

The interest toward spanners derives from the fact that it is always possible to compute *sparse* spanners in polynomial time. The previously best known

*Partially supported by the Future and Emerging Technologies programme of the European Union under contract number IST-1999-14186 (ALCOM-FT) and the Italian MIUR Project ALINWEB.

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

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

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

[1] Throughout the paper, we assume that $\widetilde{O}(f(n)) = O(f(n) \cdot \mathrm{polylog}(n))$.

bound for computing a t-spanner of a graph with n vertices and m edges was given in [22], and requires $O(t \cdot m \cdot n^{\frac{2}{t+1}})$ expected time. Very recently, Baswana and Sen announced a linear time randomized algorithm [4] for this problem.

To the best of our knowledge, no dynamic algorithm for maintaining a t-spanner of a general weighted graph under edge insertion and deletions has been proposed in the literature, while this dynamic problem has been studied for special classes of graphs (planar graphs, Euclidean graphs, etc.).

In this paper, we present a deterministic algorithm for maintaining a t-spanner of a general real weighted undirected graph under a sequence of edge insertions and deletions. In worst-case sequences our algorithm requires $\widetilde{O}\left(t \cdot n^2 \cdot \log C\right)$ amortized time per update, where n is the number of vertices, and edge costs are reals in $[1, C]$. For random sequences, the average cost per update becomes $\widetilde{O}\left(t \cdot n^{1+\frac{4}{t+2}} \cdot \log C\right)$. The spanner we maintain has $O\left(t \cdot n^{1+\frac{4}{t+2}} \cdot \log C\right)$ edges.

Our algorithm is a dynamization of a simple greedy method, and uses dynamic all-pairs shortest paths [13]. It uses simple data structures, and thus seems amenable to practical implementations.

The paper is organized as follows. After giving some preliminary definitions in Section 2, we review in Section 3 a simple static greedy algorithm. Next, our dynamic algorithm is presented in Section 4, first dealing with the simpler case of unit edge costs in Section 4.1. The technique is then extended to graphs with d different edge costs in Section 4.2 and to general real edge costs in Section 4.3.

2 Definitions

We assume the standard graph terminology as contained for instance in [11]. Let $G = (V, E)$ be an undirected graph, with V being the set of vertices and $E \subseteq V \times V$ being the set of edges. A graph $G' = (V', E')$ is a *subgraph* of graph $G = (V, E)$ if $V' \subseteq V$ and $E' \subseteq E$.

Throughout the paper, we denote by n the number of vertices and by m the number of edges in a graph G. We assume that the graph is weighted, i.e., there is a (positive) real-valued cost $c(e) > 0$ associated with each edge $e \in E$.

Given two vertices $u, v \in V$, a *path* π *in* $G = (V, E)$ *from vertex* u *to vertex* v consists of a sequence of vertices $u = v_0, v_1, \ldots, v_\ell = v$ such that $(v_{i-1}, v_i) \in E$, for $1 \le i \le \ell$. We say that edge (v_{i-1}, v_i) is in path π. A path $v_0, v_1, \ldots, v_\ell$ such that $v_0 = v_\ell$ is referred to as a *cycle*. The *cost of a path* π is the total sum of the costs of edges in π:

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

The *length of a path* π is given by the total number of edges in π.

The *distance* $dist_G(u, v)$ *from* u *to* v *in* G is given by the minimum cost of a path 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 $c(\pi) = \mathrm{dist}_G(u, v)$.

The *girth* of a graph G is the length of the shortest cycle in G.

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 has been proved in [19]:

Lemma 1 [19] *A subgraph $S = (V, E')$ of $G = (V, E)$, is a t-spanner of G if and only if the following holds:*

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

3 A greedy algorithm

It is well known that a t-spanner of a weighted undirected graph can be built by the simple greedy algorithm given in Figure 1.

```
Procedure GreedySpanner
input: real t ≥ 1 , graph G = (V, E);
output: a t-spanner of G
E' ← ∅
S ← (V, E')
foreach edge e = (x, y) ∈ E, sorted by non-decreasing c(e) do
    if dist_S(x, y) > t · c(e) then
        E' ← E' ∪ {e}
endfor
return S
```

Figure 1: Procedure GreedySpanner.

Lemma 2 *Procedure GreedySpanner builds a t-spanner S of G.*

Proof (Sketch). For each edge $e \in E$, either e was added to S or there was a path in S of cost at most $t \cdot c(e)$ connecting the endpoints of e. Since edges are never deleted from S, the lemma follows from Lemma 1. $\square$

The greedy algorithm does not necessarily build a minimum t-spanner, i.e. a spanner with minimum number of edges. This is not surprising, as finding a minimum t-spanner is NP-hard even in restricted cases (see [7, 8, 9, 19, 23]), and it also is hard to approximate [16]. However, we here prove that Procedure GreedySpanner always leads to t-spanners that are sparse.

First of all we show that the girth of the t-spanner is lower bounded by a function of t.

Lemma 3 *Let S be a t-spanner built by Procedure GreedySpanner. Then all the cycles in S have at least $\lfloor t \rfloor + 2$ edges.*

Proof (Sketch). The proof proceeds by contradiction. Assume that S contains a cycle C having less than $\lfloor t \rfloor + 2$ edges. Let $e_{\max} = (u, v)$ be the last edge in C considered by the greedy algorithm. Since edges are considered by non-decreasing cost, all edges in C have cost less than or equal to $c(e_{\max})$.

This implies that, when $e_{\max}$ is considered, all edges in $C \setminus \{e_{\max}\}$ are already in S, and give a path from u to v containing at most $\lfloor t \rfloor$ edges, thus having cost at most $\lfloor t \rfloor \cdot c(e_{\max})$. Thus, the condition on line 4. of Procedure GreedySpanner would be violated, and edge $e_{\max}$ would not be added to S, clearly a contradiction. □

Note that Lemmas 2 and 3 apply even if the greedy algorithm considers path lengths in place of distances, i.e., if it does not take into account edge costs. To be more precise, we can still examine edges by non-decreasing costs, but we may decide whether an edge has to be added to the spanner on the basis of the *unweighted* distance, i.e., we include edge (x, y) in the spanner if and only if there is no path from x to y containing at most $\lfloor t \rfloor$ edges.

The following result ensures the sparseness of hight girth graphs:

Theorem 4 [5, 6, 15] *A graph with more than $90kn^{1+\frac{1}{k}}$ edges contains necessarily a $2k$-cycle.*

Combining Theorem 4 with Lemma 3 yields the following:

Lemma 5 *The number of edges of the t-spanner computed by algorithm GreedySpanner is*

$$O\left(t \cdot n^{1+\frac{2}{t+1}}\right) .$$

4 Dynamic t-spanners

The main result of this paper is a fully dynamic deterministic algorithm for maintaining a t-spanner. In worst-case sequences, our algorithm achieves the following amortized bounds:

Theorem 6 *A t-spanner of a graph with real-valued edge costs in $[1, C]$ can be maintained in $\widetilde{O}\left(t \cdot n^2 \cdot \log C\right)$ amortized time per operation over a sequence of $\Omega\left(n^{1+\frac{4}{t+2}}\right)$ edge insertions and edge deletions. The spanner has*

$$O\left(t \cdot n^{1+\frac{4}{t+2}} \cdot \log C\right)$$

edges.

For random sequences, our algorithm achieves the following average bounds:

Theorem 7 *If the sequence of updates is random, i.e., if the edge to be deleted or inserted is selected uniformly at random, then the average cost per update is*

$$\widetilde{O}\left(t \cdot n^{1+\frac{4}{t+2}} \cdot \log C\right) .$$

For the sake of presentation, we first show how to handle the simple case of unit cost graphs. Next, we tackle the maintenance of spanners in graphs with d different edge costs. Finally, we show the algorithm for general non-negative real cost values.

4.1 Maintaining t-spanners for unit cost graphs

In the case of unit costs, t-spanners with large girth show a minimality property that does not always hold in the general case.

Lemma 8 *If S is a subgraph of a graph G with unit costs, and S has girth at least $\lfloor t \rfloor + 2$, then no proper subgraph of S is a t-spanner of G.*

Proof (Sketch). Let $e = (x, y)$ be any edge in S. By hypothesis, e does not belong to a cycle of length less than $\lfloor t \rfloor + 2$, hence $\text{dist}_{S \setminus \{e\}}(x, y) > \lfloor t \rfloor + 1 > t$. Thus, by Lemma 1, $S \setminus \{e\}$ cannot be a t-spanner of G. □

We now give two procedures for updating a t-spanner under the insertion or deletion of an edge.

Procedure UnitInsert
input: real $t \geq 1$, a t-spanner $S = (V, E')$ of $G = (V, E)$, an edge $e = (x, y) \notin E$;
output: a t-spanner (V, E') of $(V, E \cup \{e\})$

1. **if** $\text{dist}_S(x, y) > t$
2. $\quad E' \leftarrow E' \cup \{e\}$
3. $\quad$ update the distance on S between any pair of vertices

Figure 2: Procedure UnitInsert.

Procedures UnitInsert and UnitDelete in Figures 2 and 3 are based on a simple consideration. In the case of unit costs, algorithm GreedySpanner can consider edges in any order; moreover, if S is the t-spanner computed by GreedySpanner when edges are considered according to a given permutation π_1, the same t-spanner results from first considering all the edges in S and then all the other edges.

In practice, Procedures UnitInsert and UnitDelete mimic the greedy algorithm. In the case of the insertion of a new edge e, since all edges have the same cost, we may assume that GreedySpanner considers e as the last edge.

Consider now the deletion of an edge e. If e is not part of the spanner, then no change needs to be done. Otherwise, if the edge e is part of the spanner,

Procedure UnitDelete
input: real $t \geq 1$, a t-spanner $S = (V, E')$ of $G = (V, E)$, an edge $e = (x, y) \in E$;
output: a t-spanner of $(V, E \setminus \{e\})$

1. **if** $e \in E'$
2.   delete e from S
3.   **foreach** $f = (x, y) \in E \setminus E'$ **do**
4.   **if** $\text{dist}_S(x, y) > t$ **then**
5. $E' \leftarrow E' \cup \{f\}$
6.   update the distance on S between any pair of vertices
7.   **endfor**

Figure 3: Procedure UnitDelete.

we may suppose that the greedy algorithm considers first all the edges in the spanner, e being the last of them, and then all the other edges. Hence, the only change in the behavior of GreedySpanner concerns non-spanner edges, that after the deletion of e may become part of the spanner. This proves the following:

Theorem 9 *Procedures UnitInsert and UnitDelete correctly updates the t-spanner, and the t-spanner has girth at least $\lfloor t \rfloor + 2$.*

By Lemma 8 and Theorem 9, we can state that the dynamic algorithm for unit costs maintains a minimal t-spanner.

Theorem 10 *A t-spanner of a graph with unit costs can be maintained in $\widetilde{O}\left(t \cdot n^2\right)$ amortized time per operation on a sequence of $\Omega(n^{1+\frac{2}{t+1}})$ edge insertions and edge deletions.*

Proof (Sketch). We maintain $\text{dist}_S(u, v)$, for each pair of vertices u, v, using the method proposed in [13], that performs updates in $\widetilde{O}(n^2)$ amortized time per operation and distance queries in $O(1)$ worst case time.

Let us first consider the insertion of an edge e: we can test in $O(1)$ time whether e has to be added to the spanner. In this case, the distance between any pair of vertices is updated in $\widetilde{O}(n^2)$.

Deletion of an edge e, if it is not in the spanner, is performed in constant time as well. If e is in E', we update all pair distances, still requiring $\widetilde{O}(n^2)$ amortized time, then we check for $O(n^2)$ edges in $E \setminus E'$ to be possibly added to S. For each edge added to the spanner we again update all pair distances.

Each operation would require $\widetilde{O}(n^2)$ amortized time, with the only exception of the deletion of a spanner edge, which causes an all pair distance update for each edge that enters in the spanner due to the deletion (lines 3–7 in Procedure UnitDelete).

We need now to bound the total number ρ of edges added to S in line 5. of Procedure UnitDelete, while performing the whole sequence of operations. Let

m_0 be the number of edges in the spanner before performing the sequence of updates. The number of spanner edges m_f at the end of the sequence is

$$m_f = m_0 + \sigma_i - \sigma_d + \rho \ ,$$

where σ_i is the number if edge insertions fulfilling the test in line 1 of Procedure UnitInsert, and σ_d is the number if edge deletions fulfilling the test in line 1 of Procedure UnitDelete. We have thus

$$\rho = m_f - m_0 - \sigma_i + \sigma_d \leq m_f + \sigma_d \ ,$$

and, by Theorems 9 and 4,

$$\rho = O\left(t \cdot n^{1+\frac{2}{t+1}} + \sigma_d\right) \ .$$

The overall worst case time is thus

$$\widetilde{O}\left(n^2 \cdot (\sigma_i + \sigma_d) + n^2 \cdot \rho\right) = \widetilde{O}\left(n^2 \cdot (\sigma_i + \sigma_d) + t \cdot n^{3+\frac{2}{t+1}}\right) \ ,$$

giving the theorem. □

4.2 Graphs with d different costs

In this section we present an extension of the unit cost procedure to graphs whose edge costs may assume d different values $\{c_1, c_2, \ldots, c_d\}$, with $c_i \geq 0$ for $i = 1, \ldots d$. In the next section we show that the same approach also applies to graphs with general non-negative costs.

Procedure EdgeInsert
input: real $t \geq 1$, a t-spanner $S = (V, E')$ of $G = (V, E)$, an edge $e = (x, y) \notin E$;
output: a t-spanner of $(V, E \cup \{e\})$

1. **if** $\text{dist}_S(x, y) > t \cdot c(e)$
2. $\quad E' \leftarrow E' \cup \{e\}$
3. $\quad$ update the distance on S between any pair of vertices

Figure 4: Procedure EdgeInsert.

Procedures EdgeInsert and EdgeDelete, respectively in Figures 4 and 5, are simple extensions of Procedures UnitInsert and UnitDelete. We can prove that both procedures correctly maintain a t-spanner S.

Theorem 11 *Procedure EdgeInsert correctly maintains a t-spanner.*

Proof (Sketch). The inserted edge $e = (x, y)$ is not included in the t-spanner only if, without e, $\text{dist}_S(x, y) \leq t \cdot c(e)$. In this case the condition in Lemma 1 is fulfilled. □

Procedure EdgeDelete
input: real $t \geq 1$, a t-spanner $S = (V, E')$ of $G = (V, E)$, an edge $e = (x, y) \in E$;
output: a t-spanner of $(V, E \setminus \{e\})$

```
if e ∈ S
    delete e from S
    foreach f = (x, y) ∈ E \ E' do
        if dist_S(x, y) > t · c(f) then
            E' ← E' ∪ {f}
            update the distance on S between any pair of vertices
    endfor
```

Figure 5: Procedure EdgeDelete.

Theorem 12 *Procedure EdgeDelete correctly maintains a t-spanner.*

Proof (Sketch). After deleting an edge, condition 2 in Lemma 1 is checked for all edges in $E \setminus E'$. Hence, at the end of the loop condition 2 holds for all $e \in E$. □

Moreover, if we consider the subgraph S_i of S given by all edges in S whose cost is w_i, we can show that all the cycles in each S_i have at least $\lfloor t \rfloor + 2$ edges, hence the spanner given by the above procedures in the union of d graphs, each having girth at least $\lfloor t \rfloor + 2$.

Theorem 13 *The t-spanner maintained by Procedures EdgeInsert and EdgeDelete has $O\left(d \cdot t \cdot n^{1+\frac{2}{t+1}}\right)$ edges.*

Proof (Sketch). Assume the theorem holds before performing an edge insertion or deletion. In both cases, an edge f is added to E' only if $\text{dist}_S(x, y) > t \cdot c(f)$. Consider the subgraph composed by all the spanner edges with cost $c(f)$: the above condition implies that in this subgraph we never introduce cycles containing less than $\lfloor t \rfloor + 2$ edges. The t-spanner maintained is thus contained in the union of d graphs, each having girth at least $\lfloor t \rfloor + 2$, where d is the number of different cost values. The theorem now follows directly from Theorem 4. □

The sparseness of the t-spanner implies the time bounds stated in Theorem 10, with an extra factor d.

Theorem 14 *A t-spanner of a graph with d different edge costs can be maintained in $\widetilde{O}\left(d \cdot t \cdot n^2\right)$ amortized time per operation on a sequence of $\Omega(n^{1+\frac{2}{t+1}})$ edge insertions and edge deletions.*

4.3 Real costs

It is interesting to note that, in the case of d different edge costs treated in the previous section, the partition of the t-spanner into d subgraph is only introduced in order to bound the number of edges in the spanner, but the procedures just maintain a single t-spanner built on the whole graph.

Exactly the same procedures EdgeInsert and EdgeDelete can be applied to graphs with general non-negative costs, where the number of different costs can be $\Theta(n^2)$.

Let the edge costs in graph $G = (V, E)$ be reals in the interval $[1, C]$. We define a new graph G', in which cost values are replaced by integer powers of 2: for each edge $e \in E$ there is an edge in G' with $c'(e) = 2^{\lfloor \log_2 c(e) \rfloor}$. Cost $c'(e)$ may have at most $\log_2 C$ different values.

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

- $c'(e) \leq c(e) < 2 \cdot c'(e)$;
- $\mathrm{dist}_{S'}(x, y) \leq \mathrm{dist}_S(x, y) < 2 \cdot \mathrm{dist}_{S'}(x, y)$.

A t-spanner of G' actually is also a $(2 \cdot t)$-spanner of G. In fact, if S' is a t-spanner of G', the above relationships allows us to write that for any edge $e = (x, y)$ in E :

$$\mathrm{dist}_S(x, y) < 2 \cdot \mathrm{dist}_{S'}(x, y) \leq 2 \cdot t \cdot c'(e) \leq 2 \cdot t \cdot c(e) \ .$$

Thus, if we replace edge costs by integer powers of 2, apply procedures EdgeInsert and EdgeDelete using a $t/2$ stretch factor, and look back to the original edge costs, we obtain a t-spanner of G having

$$O\left(t \cdot n^{1+\frac{4}{t+2}} \cdot \log_2 C\right)$$

edges.

Rephrasing Theorem 14, applied to $\log C$ edge cost values, we can state that a t-spanner can be maintained in $\widetilde{O}\left(t \cdot n^2 \cdot \log C\right)$ amortized time per operation on a sequence of $\Omega(n^{1+\frac{4}{t+2}})$ edge insertions and edge deletions, thus proving Theorem 6.

Theorem 7 derives from the observation that our algorithm performs some work only if the spanner needs to be updated. Since t-spanners contain a small fraction of the graph edges, under random sequences of deletions and insertions the spanner will not be modified too often. A formal proof of Theorem 7 will be given in the full paper.

References

[1] Althofer, Das, Dobkin, Joseph, and Soares. On sparse spanners of weighted graphs. *GEOMETRY: Discrete & Computational Geometry*, 9, 1993.

[2] Baruch 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)*, 2003. To appear.

[5] B. Bollobás. *Extremal graph theory.* Academic Press, London, 1978.

[6] J. A. Bondy and M. Simonovits. Cycles of even length in graphs. *J. Combin. Theory Ser. B*, 16:97–105, 1974.

[7] Ulrik Brandes and Dagmar Handke. *NP*-completeness results for minimum planar spanners. *Discrete Mathematics and Theoretical Computer Science*, 3(1):1–10, 1998.

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

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

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

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

[12] 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.

[13] C. Demetrescu and G. F. Italiano. A new approach to dynamic all pairs shortest paths. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC-03)*, New York, 2003. ACM Press. to appear.

[14] 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.

[15] R. J. Faudree and M. Simonovits. On a class of degenerate extremal graph problems. *Combinatorica*, 3:83–93, 1983.

[16] G. Kortsarz. On the hardness of approximating spanners. *Algorithmica*, 30(3):432–450, 2001.

[17] Liestman and Shermer. Additive graph spanners. *NETWORKS: Networks: An International Journal*, 23, 1993.

[18] Liestman and Shermer. Grid spanners. *NETWORKS: Networks: An International Journal*, 23, 1993.

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

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

[21] Richards and Liestman. Degree-constrained pyramid spanners. *JPDC: Journal of Parallel and Distributed Computing*, 25, 1995.

[22] Mikkel Thorup and Uri Zwick. Approximate distance oracles. In ACM, editor, *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing: Hersonissos, Crete, Greece, July 6–8, 2001*, pages 183–192, New York, NY, USA, 2001. ACM Press.

[23] G. Venkatesan, U. Rotics, M. S. Madanlal, J. A. Makowsky, and C. Pandu Rangan. Restrictions of minimum spanner problems. *Information and Computation*, 136(2):143–164, 1997.