Generating Sparse Spanners For Weighted Graphs [1]

Report for Advanced Algorithms and Data Structures Seminar

Tierry Hörmann

March 31, 2018

1 Introduction

1.1 Overview

In this report we present and discuss the results of the paper [1]. First we introduce some definitions and notions which are often used in the paper together with the main goal, which was reached by the paper. Afterwards we present the main algorithm, which the paper is all about. In section 3 we show an analysis of the algorithm showing his practicality by providing and proving two important theorems. In section 4 and 5 we then show further interesting results gained with the presented algorithm and conclude by giving some critical thoughts about the paper.

In the associated talk to this report, only sections 1, 2 and 3 will be covered, where proofs in section 3 will sometimes only be sketched and subsection 3.3 will only be mentioned briefly. Section 4 will be omitted in the talk, since they don't seem to be the main focus of the paper.

1.2 Definitions

We begin by providing a few definitions which will be used further on.

Definition 1 (weight)**.** The weight of a graph G, denoted by $\mathit{Weight}(G)$, indicates the sum of all edge weights of G.

Definition 2 (length)**.** The weight of a path P is called the length of P and is denoted by $\mathit{Length}(P)$. The weight of a single edge e can also be denoted by $\mathit{Length}(e)$.

Definition 3 (size)**.** The size of a graph $G = (V, E)$, denoted by $\mathit{Size}(G)$, indicates the number of edges $|E|$ of G

Definition 4 (t-spanner)**.** Let $G = (V, E)$ be a connected graph with positive weighted edges. A subgraph $G' = (V, E')$ is called a *t-spanner*, if for every pair of vertices $[u, v]$, the shortest path between u and v in G' is at most t times longer, than the shortest path in G. The parameter t is then called the *stretch factor* of G'.

Figure 1 shows an example of a 3-spanner of a graph. To recognize that it is indeed a 3-spanner, remark that every thin edge is representable by a path of fat edges, which is at most 3 times longer than the thin edge.
A graph G is said to be sparse in weight, if $\mathit{Weight}(G)$ is small, and sparse in size accordingly.
Also we will further on use the common abbrevation of m as the number of edges, and n as the number of vertices of a graph.

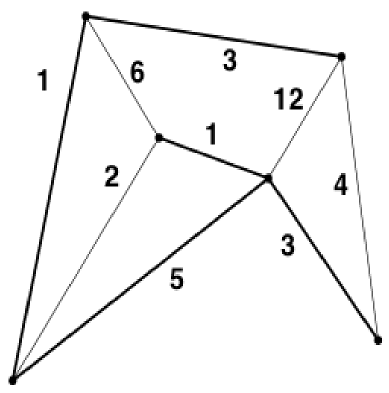

Figure 1: A graph where the fat edges indicate a possible 3-spanner of the full graph.

1.3 Goal

Our goal is to provide an algorithm which is able to generate a t-spanner out of a graph G, whose sparseness in weight comes arbitrarily close to the sparseness of its minimum spanning tree. In addition we would like the algorithm to produce a result that is sparse in size as well. This means, that we would like the algorithm to provide a spanner whose size is arbitrarily close to the size of a spanning tree of G, which is $n-1$.

A valid question is now, what could we use such an algorithm for?
For certain use-cases, general graphs can provide way more information than actually needed, or even contain a lot of redundancy. Such an algorithm can then eliminate the additional, useless information for a small price. Let's consider, as an example, a computer network, for which we can assume that nodes won't fail.
Let's now say we have a routing algorithm, which requires a full graph representation of the network to be stored on every node. For routing, a node then has to calculate the shortest path from the source to the destination. In this scenario we will then most probably store edges, which are never or very rarely used, or which could be replaced with a path with a similarly small length. This means that we store edges, which we don't really need for this use case. An algorithm as described above could eliminate such edges at low cost (i.e. the routing results won't be much worse than before). This means, that routing could be more performant, by for example extending Dijkstra's algorithm [3] with Fibonacci heaps [4] to improve time complexity from $O(n^2)$ to $O(m + n \log n)$ (which for a standard graph might not be better, since $m \in O(n^2)$ in general). Also we would need less storage, indeed close to storage linear in n.

2 Algorithm

We now provide an algorithm, which reaches the required goal. The next section will then provide an analysis of the algorithm, which shows, that the goal is really reached. The following algorithm takes as input a weighted graph G and a positive parameter r and outputs a graph $G' \subseteq G$, where G' is a r-spanner of G.

The above algorithm is slightly different than the one provided in the paper, because it has a different condition in the if statement. This is, because in the paper it is not mentioned what would happen in the case, if there is no shortest path between the two selected nodes in G'. However further on in the paper it gets pretty clear, what is meant to happen (especially during the comparison with Kruskal's algorithm [5]).

If we compare algorithm 1 to Kruskal's minimum spanning tree algorithm [5], we recognize, that it is basically a generalized minimum spanning tree algorithm, since for infinite r it behaves exactly as Kruskal's algorithm. We will later use this observation when comparing G' to the minimum spanning tree of G ($MST(G)$).

Algorithm 1 Spanner

```
function SPANNER(G = (V, E), r)
    Sort $E$ by nondecreasing weight
    $G' \leftarrow (V, \emptyset)$
    for $e = [u, v] \in E$ do
        Compute $P(u, v)$, the shortest path from $u$ to $v$ in $G'$
        if $P(u, v)$ does not exist or $r \cdot \mathit{Length}(e) < \mathit{Length}(P(u, v))$ then
            $G' \leftarrow G' \cup \{e\}$
        end if
    end for
    return $G'$
end function
```

3 Analysis

3.1 Useful Lemmas

For our analysis of the above algorithm, we first provide some useful lemmas, which are easy to prove and which already show the direction we will go. We will from now on use G' to refer to the output of the algorithm.

First we prove, that algorithm 1 indeed outputs what we want, i.e. a r-spanner of G.

Lemma 3.1. G' is a r-spanner of G

Proof. Let $e = [u, v] \in G - G'$ be an arbitrary edge, which is in G, but not in G'. When e is examined by the algorithm, the if condition must fail, this means, that there exists a path $P(u, v)$ from u to v, whose length is at most r times longer than e. Since the algorithm does not remove edges from G', the resulting graph also includes $P(u, v)$.
Now consider an arbitrary shortest path in G. From the above result we see, that we can substitute every edge in this path that is missing in G' by a path that is at most r times longer, and therefore the full path is at most r times longer. ☐

Now we show, that there is a lower bound for the size of a cycle in G', which is already a step in the direction of showing sparseness in size of G'.

Lemma 3.2. Let C be any simple cycle in G', then $\mathit{Size}(C) > r + 1$

Proof. Assume that G' contains a cycle with $\mathit{Size}(C) \leq r + 1$. Let $e = [u, v]$ be the last edge (one of the largest) of C, that was examined by algorithm 1. When e is examined by the algorithm, there is already a path $P(u, v)$ in G' from u to v with a size of at most r, e.g. the remainder of C. Since every edge in $P(u, v)$ is smaller or equal to e, we know, that $\mathit{Length}(P(u, v)) \leq r \cdot \mathit{Length}(e)$, hence e should not have been added to G', which is a contradiction. ☐

The next lemma comes very straight forward from the if condition in algorithm 1.

Lemma 3.3. Let C be any simple cycle in G' and e be an arbitrary edge in C, then $\mathit{Length}(C - e) > r \cdot \mathit{Length}(e)$.

Proof. Assume that G' contains a cycle C for which there exists an edge e in C, such that $\mathit{Length}(C - \{e\}) \leq r \cdot \mathit{Length}(e)$, i.e. it violates the above condition. Clearly the largest edge e_m in C applies now for e. When e_m was examined by algorithm 1, again all other edges of C must already have been added to G', which means that there is a path $P(u, v) = C - \{e\}$ with $\mathit{Length}(P(u, v)) \leq r \cdot \mathit{Length}(e_m)$. Hence e_m should not have been added to G', which is a contradiction. ☐

To later compare G' with $MST(G)$, we now prove the following lemma:

Lemma 3.4. $MST(G) \subseteq G'$

Proof. Let's indicate by G'_i the graph G' after the i'th iteration of the loop of the algorithm, where $G'_0 = \emptyset$ and $G'_m = G'$.
Now let's indicate by M_i the graph after the i'th iteration of Kruskal's algorithm, where $M_0 = \emptyset$ and $M_m = MST(G)$. By induction we now can show, that for any n, $M_n \subseteq G'_n$, hence $M_n \subseteq G'_n$.
Base case: We obviously have $M_0 = \emptyset \subseteq \emptyset = G'_0$.
Induction hypothesis: We can assume, that for n fixed: $M_n \subseteq G'_n$.
Step case ($n \to n+1$): We can assume without loss of generality, that an edge $e = [u, v]$ was added to M_n after the next iteration (otherwise we trivially get our result via the induction hypothesis, since $M_n = M_{n+1}$). If we now take a look at the if condition of the algorithm, we see that for e to be added, there must not be any path between u and v in M_n. This obviously also has to be the case for G'_n. Hence e is also added to G'_n after the next iteration. With our induction hypothesis we therefore get $M_{n+1} = M_n \cup \{e\} \subseteq G'_n \cup \{e\} = G'_{n+1}$. □

3.2 Main Theorems

Now we come to the main part of the analysis. The paper provides two main theorems which summarize the sparseness in weight and in size of the t-spanner resulting from the algorithm:

Theorem 3.5. Given a n-vertex graph G and a $t \geq 1$, there is a polynomially constructible $(2t+1)$-spanner G' such that

1. *Size*$(G') < n \cdot \lceil n^{1/t} \rceil$
2. *Weight*$(G') <$ *Weight*$(MST(G)) \cdot (1 + n/2t)$

Theorem 3.6. Given a n-vertex *planar* graph G and a $t \geq 1/2$, there is a polynomially constructible $(2t+1)$-spanner G' such that

1. *Size*$(G') \leq (n-2) \cdot (1 + 2/\lfloor 2t \rfloor)$
2. *Weight*$(G') <$ *Weight*$(MST(G)) \cdot (1 + 1/t)$

3.2.1 Further Lemmas

For proving our two theorems from above, we need a few additional Lemmas where we will use some properties of planar graphs. For completion we therefore provide the following two definitions:

Definition 5 (face)**.** After drawing a planar graph G in a plane, we call a subset of the plane, which is disjoint from G and only bounded by edges in G a *face* of G.

Definition 6 (face size)**.** The *size* of a face is the number of encountered edges while traversing the face, where repetitions of edges are allowed while traversing.

We might now ask, when do we need to repeat edges? Figure 2 shows an example of such a graph.

The first Lemma gives a bound on the size of a planar graph, given a lower bound on the face size of the graph. This will help us in proving the size bound of Theorem 3.6.

Lemma 3.7. If all faces of a planar graph G have sizes $\geq r$, then *Size*$(G) \leq (n-2) \cdot (1 + 2/(r-2))$.

Proof. Euler's formula for planar graphs states that $n - m + f = 2$, where f is the number of faces in the graph. If we traverse every face in the graph and mark the edges encountered, every edge gets marked twice. Since the number of marked edges during the traversal of one face is at least r, we get $r \cdot f \leq 2m$. With Euler's formula we hence get $m \leq (n-2) \cdot (1 + 2/(r-2))$. □

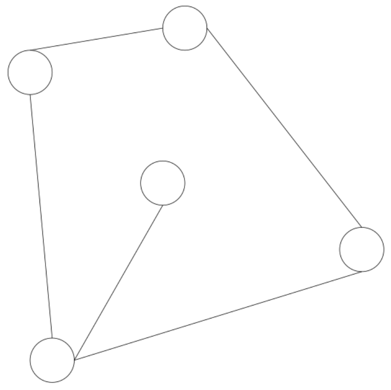

Figure 2: An example of a planar graph, where an edge has to be counted twice when traversing the bounded face

For the next lemma we need another definition:

Definition 7 (girth)**.** The *girth* of a graph is the size of its smallest simple cycle.

From extremal graph theory ([2] Theorem 3.7, Chapter III) we can derive the following lemma, which provides a upper bound on the size of a graph, which will be helpful when proving theorem 3.5.

Lemma 3.8. Let G be a graph with girth $> r$. Then $\mathit{Size}(G) < n \cdot \lceil n^{2/(r-2)} \rceil$.

3.2.2 Proofs

We now can finally prove the two theorems. In the paper there are some typos, especially in the proof of theorem 3.6, which might confuse the reader. One typo even makes the size bound of the two theorems slightly worse.

Corollary 3.8.1 (polynomial time)**.** Algorithm 1 has a polynomial runtime.

Proof. First of all recognize, that the algorithm takes time $O(m \log m)$ to sort its edges in the first step. Afterwards it iterates the loop m times, each time executing a shortest path algorithm which takes $O(m + n \log n)$ time. Hence we get a total runtime of $O(m \log m + m \cdot (m + n \log n))$ which is clearly polynomial. □

Proof of theorem 3.6. Let $t \geq 1/2$. We run algorithm 1 on a planar graph G and $r = 2t+1$ and receive output G'. By lemma 3.1, G' has a stretch factor of $2t+1$. By lemma 3.2, the girth of G' is $> 2t+2 \geq \lfloor 2t+3 \rfloor$. Hence the minimum face size is also $\geq \lfloor 2t+3 \rfloor$. By lemma 3.7 we now get: $\mathit{Size}(G') \leq (n-2) \cdot (1 + 2/\lfloor 2t+1 \rfloor) \leq (n-2) \cdot (1 + 2/\lfloor 2t \rfloor)$, which proves the size bound of theorem 3.6.

For our weight bound let's describe the following polygon emerging by $MST(G)$: The polygon contains the same vertices as $MST(G)$. Its border consists of edges of $MST(G)$ (every edge is included twice on the border) and results by traversing $MST(G)$. This polygon has no area and a perimeter of $2 \cdot \mathit{Weight}(MST(G))$. Now we will "grow" this polygon in iterations, such that it finally includes the full graph G' by adding a new edge from $G' - MST(G)$ in every iteration. One iteration now consists of selecting an edge $e = [u, v]$ from $G' - MST(G)$ that wasn't already added and which builds a face with $P(u, v)$ that is adjacent to the current polygon, where $P(u, v)$ is part of the current polygon. We then remove $P(u, v)$ from the polygon and instead add e. Figure 3 shows an example of such a polygon after the 4th iteration. Let now the length of the polygon at iteration i be W_i with $W_0 = 2 \cdot \mathit{Weight}(MST(G))$ and T_i be the total length of edges in $G' - MST(G)$ that were added until iteration i with $T_0 = 0$. By lemma 3.3 we now get:

$$\mathit{Length}(e) \cdot (2t+1) < \mathit{Length}(P(u, v))$$

This means that:

$$W_{i+1} = W_i - \mathit{Length}(P(u, v)) + \mathit{Length}(e) < W_i - \mathit{Length}(e) \cdot (2t+1) + \mathit{Length}(e) = W_i - 2t \cdot \mathit{Length}(e)$$

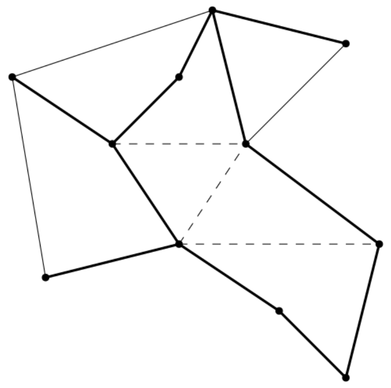

Figure 3: An example for the polygon descibed in the proof of theorem 3.6 after 4 iterations. The fat edges indicate the polygon, while the dashed edges mark the edges that were once part of the polygon (and are now inside the polygon) and the normal edges mark the edges that will be added to the polygon in further iterations.

and $T_{i+1} = T_i + \mathit{Length}(e)$.
Let's denote by e_i the edge added to the polygon at iteration i. We therefore get:

$$W_i < W_0 - 2t \cdot \sum_{j=1}^{i} \mathit{Length}(e_j) = W_0 - 2t \cdot T_i$$

By reformulating we get:

$$T_i < (W_0 - W_i)/2t = (2 \cdot \mathit{Weight}(MST(G)) - W_i)/2t < \mathit{Weight}(MST(G))/t$$

So every T_i, especially also the last one, is smaller than $\mathit{Weight}(MST(G))/t$. Thus

$$\mathit{Weight}(G') < \mathit{Weight}(MST(G)) + \mathit{Weight}(MST(G))/t = \mathit{Weight}(MST(G)) \cdot (1 + 1/t)$$

. □

Proof of theorem 3.5. Let $t \geq 1$. We run algorithm 1 on a graph $G = (V, E)$ and $r = 2t + 1$ and receive output G'. By lemma 3.1, $G' = (V, E')$ has a stretch factor of $\leq 2t + 1$. And by lemma 3.2, the girth of the output is $> 2t + 2$. Thus, by lemma 3.8, $\mathit{Size}(G') < n \cdot n \cdot \lceil n^{1/t} \rceil$

For the weight bound we may proceed somehow similarly as above. By lemma 3.4, $MST(G)$ is contained in G'. For each vertex v, consider the graph $G_v = MST(G) \cup E_v$ where $E_v = \{[u, v] \in E'; u \in V \wedge [u, v] \notin MST(G)\}$ (the edges in E' incident to v, but not in $MST(G)$. Obviously G_v cannot contain a subdivision of the complete graph K_5, nor the complete bipartite graph $K_{3,3}$ and thus must be planar. If we apply a similar method as in the previous proof, we can show, that $\mathit{Weight}(E_v) < \mathit{Weight}(MST(G))/t$. Therefore we get $\sum_{v \in V} \mathit{Weight}(E_v) < \mathit{Weight}(MST(G)) \cdot n/t$, where each edge in $G' - MST(G)$ has now been counted twice in the sum. We hence get $\mathit{Weight}(G') < \mathit{Weight}(MST(G)) \cdot (1 + n/2t)$. □

3.3 How good is this?

In the previous subsection we have provided upper bounds for weight and size, for general and planar graphs, which can be achieved by algorithm 1. The question now arises, how optimal these results are, i.e. what are lower bounds for weight and size of a t-spanner?

Let's now consider graphs with arbitrary positive weights. From [6] we get the following theorem:

Theorem 3.9. For every $t \geq 1$, there exist infinitely many graphs with unit edge weights, such that every $(2t+1)$-spanner requires $\Omega(n^{1+1/(2t+3)})$ edges.

This means, that the size bound in theorem 3.5 is tight up to a constant factor. Since during the proof of theorem 3.5 there was a small mistake in the paper, there is even a tighter bound of $n \cdot \lceil n^{2/(2t+1)} \rceil \leq$ for the algorithm (see above), which however still "only" is tight up to a constant factor in the exponent.

With theorem 3.9 we can get a lower bound on the weights of spanners:

Theorem 3.10. For every $t \geq 1$, there exist infinitely many graphs G such that every $(2t+1)$-spanner has weight $\Omega(\mathit{Weight}(MST(G)) \cdot n^{1/(2t+3)})$.

Therefore the weight bound in theorem 3.5 is not as tight as maybe wished.

Now let's consider planar graphs with arbitrary positive weights.

Theorem 3.11. For infinitely many n and t, there exists a planar graph G with unit edge weights, such that every $(2t+1)$-spanner requires $\Omega(n \cdot (1 + 1/t))$ edges, and has weight $\Omega(\mathit{Weight}(MST(G)) \cdot (1 + 1/t))$.

Proof. For infinitely many n and t it is possible to construct a planar n-vertex graph with unit edge weights, such that

1. each face is a regular $(2t+4)$-gon
2. the girth is $\geq 2t+4$

as follows:
Let $k \geq 2$ and $n = k(t+1)+2)$. Select 2 nodes u and v and create k chains containing $t+1$ nodes out of the remaining $k(t+1)$ nodes. Now connect u to one end of every chain and v to the other end and we get our graph. With Euler's formula we can now derive, that the size of such a graph must be $\Omega(n \cdot (1+1/t))$. It is easy to see, that every proper subgraph will have a stretch factor of at least $2t+3$. This means, that a $(2t+1)$-spanner must be equal to the previous graph and has therefore the same size. We can also see, that the weight condition holds, since a minimum spanning tree has at least $n-1$ edges. So with unit weights we get the same result for the weight of a $(2t+1)$-spanner: $\Omega(n \cdot (1+1/t)) = \Omega(\mathit{Weight}(MST(G)) \cdot (1+1/t))$ □

This shows, that the results of theorem 3.6 are tight.

4 Further graph theoretical results

The paper now provides further results for (generalized) spanners, especially further lower bounds. We will here present these results, but not always provide details about the proofs, since it doesn't seem as if those results are the main focus of the paper and this report should be a summary of it. Also even in the paper some theorems are provided without proofs, so there we obviously also won't state any.

First we again provide some definitions:

Definition 8 (distance)**.** The length of the shortest path between two vertices u and v in a graph G is called *distance* and is denoted by $\mathit{Dist}(u, v, G)$.

Definition 9 (compressor)**.** Let g be a function that assigns a string of s bits to each unweighted graph (with n vertices). g is called a (n, t, s)-*compressor*, if the following is true:
For all pairs of graphs G_1, G_2, if $g(G_1) = g(G_2)$, then for all pairs of vertices u, v: $\mathit{Dist}(u, v, G_1) \leq t \cdot \mathit{Dist}(u, v, G_2)$.

We can understand a compressor as a function that maps graphs to (named) groups. Such a group only contains graphs, which are similar, i.e. the distance between two arbitrary vertices are similar up to a certain parameter in two graphs of the same group.

Definition 10 (generalized t-spanner)**.** Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs with $V_1 \subseteq V_2$. G_2 is a generalized t-spanner of G_1, if for all $u, v \in V_1$, $Dist(u, v, G_1) \leq Dist(u, v, G_2) \leq t \cdot Dist(u, v, G_1)$.

So a generalized t-spanner might add new vertices and edges to a graph, but may not shorten any path. This ability allows, that the size (and weight) of certain graphs can be reduced drastically. In the general case however, generalized spanners sometimes cannot be much smaller than normal (simple) spanners, which is stated in the next theorem.

An other approach for generalizing spanners works as follows:

Definition 11 (t-approximators)**.** Let $t \geq 1$. $G' = (V, E')$ is called a *t-approximator* of $G = (V, E)$ if, for all $u, v \in V, 1/t \cdot Dist(u, v, G) \leq Dist(u, v, G') \leq t \cdot Dist(u, v, G)$.

Theorem 4.1. For infinitely many n and t, there exists a graph G with unit edge weights such that every generalized t-spanner has size $\Omega(1/(\log n) \cdot n^{1+1/(t+2)})$.

To prove this theorem we first provide some lemmas and corollaries:

Given a t, the following lemma provides a lower bound on the number of groups of a compressor.

Lemma 4.2. For a (n, t, s)-compressor to exist, $s \geq 1/4 \cdot n^{1+1/(t+2)}$.

Proof. From [2] we know, that for infinitely many n and t, there exist graphs with girth $\geq t + 2$ and size $\geq 1/4 \cdot n^{1+1/(t+2)}$. Let $G = (V, E)$ be such a graph, and let G_1 and G_2 be two different subgraphs of G. Now consider two vertices $u, v \in V$, which don't have the same distance in G_1 and G_2 and are adjacent to each other in either G_1 or G_2. Note that such vertices must exist, since G_1 is different to G_2. Because the girth is $\geq t + 2$, the distance from u to v must be $t + 1$ in one of the two subgraphs, and 1 in the other one. Therefore G_1 and G_2 are in different groups and the number of groups is $\geq 2^m$, thus $s \geq m$ (m as always being the edge count of G). $\square$

Corollary 4.2.1. If there exists a collection of t-spanners, one for each graph, of size k, then there exists a $(n, t, \lceil \log k \rceil)$-compressor.

Corollary 4.2.2. Every graph can be encoded by $2 \cdot \lceil \log n \rceil \cdot m$ bits.

Proof (sketch) of theorem 4.1. Let's first prove the theorem for unweighted spanners:
Since we are interested in sparse generalized spanners, we require, that a spanner has at most $n + 2 \cdot n(n-1)/2 = n^2$ vertices, i.e. they should introduce not more vertices than twice the number of edges of a complete graph. Now consider any collection of t-spanners where every graph has a t-spanner of this collection. Let m be the size of the largest spanner in this set. Each spanner can be encoded in $\leq 4 \cdot \lceil \log n \rceil \cdot m$ bits. By corollary 4.2.1, there exists a $(n, t, 4 \cdot \lceil \log n \rceil \cdot m)$-compressor. With lemma 4.2 we get $4 \cdot \lceil \log n \rceil \cdot m \geq 1/4 \cdot n^{1+1/(t+2)}$ and when solving for m we get our result.
For weighted spanners we try to encode the edge weights, which obviously might in general not be possible, since they can be arbitrarily large or require arbitrarily many precision bits. It however can be shown, that by ignoring large weights and rounding off other weights, we can encode spanners with slightly larger stretch factors. $\square$

Now follow some theorems without proofs, which the paper provides as well:

Theorem 4.3. For infinitely many n and t, there exists a graph G with unit edge weights, such that every t-approximator has size $\Omega(1/(\log n) \cdot n^{1+1/(t+2)})$.

Definition 12 (proper edge)**.** A edge is called *proper*, if it is never longer than any path between its end vertices.

Theorem 4.4. For infinitely many n and t, there exists a complete graph G with proper edge weights, such that every $(2t-1)$-spanner has size $\Omega(1/t \cdot n^{1+1/(2t+1)})$.

Theorem 4.5. Denote by $K(V)$ the complete euclidean graph. Let $t \geq 1/2$ Let G' be the output of algorithm 1 for the Delaunay triangulation over V in the $|| \cdot ||_2$ norm and for $r = 2t + 1$. Then G' is a $2.42 \cdot (2t + 1)$-spanner of $K(V)$, such that

1. $\mathit{Size}(G') \leq (n-2) \cdot (1 + 2/\lfloor 2t \rfloor)$
2. $\mathit{Weight}(G') < \mathit{Weight}(MST(K(V))) \cdot (1 + 1/t)$.

Now we provide a new way to construct a spanner in a multidimensional euclidean graph.
Let $G = (V, E)$ with points in V being in R^d with norm $|| \cdot ||$.
Let $G' = (V, \emptyset)$ be the spanner to be constructed.
Let $\delta > 0$ be a fixed angle and C_i a open cone with angle δ and fixed focus (same for every cone), where $C_1, \ldots, C_{s(\delta)}$ provides a covering of R^d.
For every $v \in V$, consider the covering of R^d by the open cones $C_1 + v, \ldots, C_{s(\delta)} + v$, where $C + v$ represents a translation of the open cone C to the new origin v, i.e. the focus of $C + v$ is v. For every $C_i + v$, let u be the vertex in the cone such that $||u - v||$ is minimized. Add $[u, v]$ to G' and call u the ith neighbor of v.

Lemma 4.6. For every $\epsilon < 1/(2 + \sqrt{2}$, there is an angle $\delta(\epsilon) > 0$ such that $\mathit{Dist}(v, u, G') < ||v - u||/(1 - \epsilon(2 + \sqrt{2}))$.

Theorem 4.7. For every $t > 0$, dimension d, and norm $|| \cdot ||$ of R^d, there exists a constant $c(t, d, || \cdot ||)$ such that every finite set V has a t-spanner with size of at most $c \cdot n$.

5 Conclusion

The paper provides a very simple algorithm to construct sparse t-spanners in both weight and size. The results look very promising, as the sparseness is very tight to the lower bounds.

The paper only very briefly mentions runtime, probably because it is very straightforward. Therefore we provided some runtime analysis in this report.

Some errors in the paper resulted in wrong proofs, especially for the proofs of the main theorems, which can be very confusing to the reader.

References

[1] Ingo Althoefer, Gautam Das, David P.Dobkin, and Deborah Joseph. Generating sparse spanners for weighted graphs. pages 26–37, 07 1990.

[2] Bela Bollobas. *Extremal Graph Theory.* Academic Press, 1978.

[3] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numer. Math.*, 1(1):269–271, December 1959.

[4] Michael L. Fredman and Robert Endre Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM*, 34(3):596–615, July 1987.

[5] Joseph B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, February 1956.

[6] David Peleg and Alejandro A. Schäffer. Graph spanners. *Journal of Graph Theory*, 13:99–116, 1989.