

Distribution-Sensitive Construction of the Greedy Spanner

Sander P. A. Alewijnse¹ · Quirijn W. Bouts¹ ·
Alex P. ten Brink¹ · Kevin Buchin¹

Received: 5 February 2015 / Accepted: 27 April 2016 / Published online: 6 May 2016
© The Author(s) 2016. This article is published with open access at Springerlink.com

Abstract The greedy spanner is the highest quality geometric spanner (in e.g. edge count and weight, both in theory and practice) known to be computable in polynomial time. Unfortunately, all known algorithms for computing it on n points take $\Omega(n^2)$ time, limiting its applicability on large data sets. We propose a novel algorithm design which uses the observation that for many point sets, the greedy spanner has many ‘short’ edges that can be determined locally and usually quickly. To find the usually few remaining ‘long’ edges, we use a combination of already determined local information and the well-separated pair decomposition. We give experimental results showing large to massive performance increases over the state-of-the-art on nearly all tests and real-life data sets. On the theoretical side we prove a near-linear expected time bound on uniform point sets and a near-quadratic worst-case bound. Our bound for point sets drawn uniformly and independently at random in a square follows from a local characterization of t -spanners we give on such point sets. We give a geometric property that holds with high probability, which in turn implies that if an edge set on these points has t -paths between pairs of points ‘close’ to each other, then it has t -paths between all pairs of points. This characterization gives an $O(n \log^2 n \log^2 \log n)$ expected time bound on our greedy spanner algorithm, making it the first subquadratic time algorithm for this problem on any interesting class of points. We also use this characterization to give an $O((n + |E|) \log^2 n \log \log n)$ expected time algorithm on uniformly distributed

Quirijn W. Bouts and Kevin Buchin are supported by the Netherlands Organisation for Scientific Research (NWO) under Project Nos. 639.023.208 and 612.001.207 respectively.

✉ Quirijn W. Bouts
q.w.bouts@tue.nl

✉ Kevin Buchin
k.a.buchin@tue.nl

¹ Eindhoven University of Technology, Eindhoven, The Netherlands

points that determines whether E is a t -spanner, making it the first subquadratic time algorithm for this problem that does not make assumptions on E .

Keywords Computational geometry · Spanners · Greedy spanner · Uniform points

1 Introduction

A *Euclidean graph* on a set of n points in the Euclidean plane is a weighted graph with geometric distances as edge weights. If a shortest route in the graph is at most t times longer than the direct geometric distance between its endpoints, we say these endpoints *have a t -path*: a Euclidean graph is a t -spanner if all pairs of points have t -paths. For any $t > 1$, we can efficiently find a t -spanner with $O\left(\frac{n}{t-1}\right)$ edges in the Euclidean plane [20]. These ‘approximations’ thus have very few edges compared to the complete Euclidean graph, while approximately maintaining distances. This makes them a useful tool in many areas.

Bounded degree spanners are used in wireless network design [14], where for example points of high degree tend to have problems with interference. By using such a bounded degree spanner the problem of interference is reduced while the connectivity is maintained. A considerable amount of research has been done on the topic of spanners [15, 20] since they were introduced in network design [21] and in geometry [10]. Spanners have been used as components in various geometric and distributed algorithms.

Many different construction methods exist for t -spanners, where t can be parameterized to an arbitrary value greater than 1, each having different advantages and disadvantages. An in-depth treatise of these spanners can be found in the book [20]. We focus on the greedy spanner, which is defined as the graph resulting from repeatedly adding the edge between the closest pair of points which do not have a t -path yet. The result is a very sparse graph with asymptotically optimal edge count, degree and weight. On uniform point sets and for $t = 2$, one of its closest well-known competitors with respect to these three properties is the Θ -graph. It has about ten times as many edges, twenty times higher total weight and six times higher maximum degree on uniformly distributed point sets in practice. Figure 1 clearly shows the contrast between these two spanners. Unfortunately, all known algorithms computing the greedy spanner use $\Omega(n^2)$ time [3], making the spanner impractical to compute on large point sets.

We observed that on real-world examples, the greedy spanner contains mostly short edges with at most a few longer edges. Whether an edge is included in the greedy spanner depends only on the points and edges in an ellipse with its endpoints as foci and with eccentricity $1/t$. For pairs of points close to each other, this is a small area that tends to contain few other points. We will show that you can easily find these short edges using a bucketing scheme, giving a speedup on such point sets.

For the ‘long’ edges, we consider the ‘long’ well-separated pairs from a well-separated pair decomposition (WSPD) [9]: it is known that the greedy spanner contains at most one edge per well-separated pair in this decomposition [3]. We first compute information from the ‘short’ edges, attempting to find witnesses that show that certain ‘long’ well-separated pairs will not contain greedy spanner edges. This information

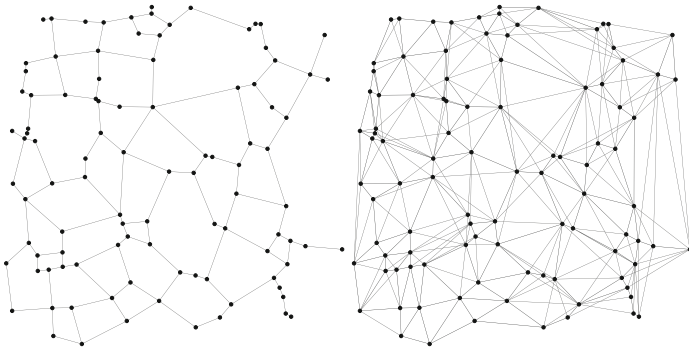


Fig. 1 The *left* rendering shows the greedy spanner on 100 points distributed uniformly in a *square* with $t = 2$. The *right* rendering shows the Θ -graph on the same points with $k = 6$ for which it has been proven that it achieves a dilation of 2

is represented by *path-hyperbola*. We then perform a standard algorithm [3] on the (hopefully only few) well-separated pairs for which we cannot find such a witness.

We present experimental results showing that the algorithm above works very well on many data sets, ranging from real-world data sets to sets which are generated according to different distributions. Speedups vary from an (apparently) linear factor to a constant factor. In particular, on a uniformly distributed point set with 300,000 points, our new algorithm needs 19 minutes to compute the greedy spanner for $t = 2$, while the currently fastest linear space algorithm, WSPD-Greedy [3], needs 17 hours on the same set. The only other linear space algorithm, lazy-greedy [7], even needs 55 hours on this set. The quadratic space algorithms already use gigabytes of memory at 10,000 points making a time comparison on large sets impossible. See [7] for a comparison of WSPD-Greedy, lazy-greedy and the main quadratic space algorithms.

We show that our algorithm has a near-quadratic worst-case time bound. We give formal evidence for the algorithm's good behavior observed in experiments on realistic point sets (which are often reasonably spread out) by analyzing its performance on point sets distributed uniformly and independently at random in a square (or 'uniformly distributed points' for short).

Euclidean graphs are frequently analyzed on uniformly distributed points, both concerning theoretical properties and experimental evaluation of structures and algorithms. One can find examples in computational geometry [8, 18], combinatorial optimization [25, 28] and the analysis of ad-hoc networks [22, 27].

Various spanner constructions have been analyzed on uniformly distributed point sets [1, 6, 12, 24, 26]. Some of these constructions are a t -spanner for fixed t , others are parameterizable with arbitrary $t > 1$. Relatively sharp bounds have been obtained on various qualities of these spanners. This gives insight into the behavior of these constructions in situations arguably closer to realistic point sets than worst case situations.

The spanner constructions studied in these analyses have a 'local' characterization: for example, Gabriel graphs connect u, v if the disk having uv as its diameter contains no points other than u and v . For graphs with such a local characterization there are well-developed techniques to analyze them on uniformly distributed points [11]. In

this paper, however, we look at the ‘global’ property t -spanner and the greedy spanner, a graph for which the existence of an edge may depend on all other points. Previous analysis techniques do not directly apply on such properties.

We consider points distributed uniformly and independently at random in a $\sqrt{n} \times \sqrt{n}$ square. We use a $\sqrt{n} \times \sqrt{n}$ scale for our square so that if a part of this square has area A , then $O(A)$ points lie in it in expectation. We only consider the case of the Euclidean plane—our results may generalize to higher dimensions, but we did not explore this. In this introduction, when stating bounds, we assume t is a constant.

We prove that such uniformly distributed point sets are, with high probability, configured in such a way that for any edge set E , if there are t -paths between points at most $O(\log n)$ away from each other, then there are t -paths between all points. In particular, we show that we can construct a ‘witness’ of this configuration in $O(n \log^2 n \log \log n)$ expected time if it exists, thus allowing our algorithms to always give the correct answer.

This result easily implies that with high probability the greedy spanner has no long edges (longer than $O(\log n)$) and furthermore that the ‘proof’ phase of our algorithm will find the witnesses for this if it exists. As the grid strategy works well on uniformly distributed point sets, we obtain an $O(n \log^2 n \log^2 \log n)$ expected time bound on our algorithm, thus giving the first subquadratic algorithm to compute the greedy spanner on any interesting class of point sets.

Another application of our result is a method to test whether a Euclidean graph $G = (P, E)$ is a t -spanner on uniformly distributed points running in $O((n + |E|) \log^2 n \log \log n)$ expected time. This gives us the first subquadratic time algorithm for testing t -spanner on any interesting class of points. Various algorithms are known for specific graphs on arbitrary points, but not for arbitrary graphs on specific sets of points. Hellweg et al. [16] give a Monte Carlo algorithm for bounded degree graphs that distinguishes between being a t -spanner and being far away from a spanner. For specific graph classes the minimum t can be computed [2, 13], and for general graphs this t can be approximated [19].

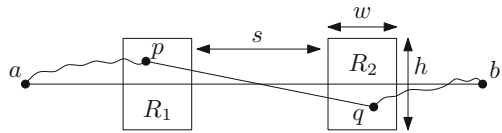
The rest of the paper is organized as follows. In Sect. 2 we introduce *bridgedness* and give a geometric lemma that will help us obtain our results. In Sect. 3 we show uniform point sets are locally- $O(\log n)$ -bridged with high probability. In Sect. 4 we give several fast algorithms that use this result. Finally, in Sect. 5 we present experimental results for our greedy spanner algorithm.

2 Bridging Points

In this section we will introduce the concept of λ -bridgedness for point sets. We will later use this concept in our characterization of t -spanners on uniformly distributed point sets. We prove two geometric lemmas that will help us with the result of Sect. 3.

Let P be a finite set of points in \mathbb{R}^2 , let $n = |P|$, and let $t \in \mathbb{R}$ be the intended dilation ($t > 1$). Let $G = (P, E)$ be a graph on P whose edges are weighted with the Euclidean distance between its endpoints. For two points $u, v \in P$, we denote the Euclidean distance between u and v by $|uv|$, and the network distance in G by $\delta_G(u, v)$ (or just $\delta(u, v)$ if G is clear from the context). We say a pair of points (u, v)

Fig. 2 (p, q) bridges (a, b)



has a t -path if $\delta(u, v) \leq t \cdot |uv|$. If all pairs of points have a t -path, the graph is called a t -spanner.

Let $a, b, p, q \in P$ be pairwise different points. We say that the pair (p, q) bridges the pair (a, b) if $t \cdot |ap| + |pq| + t \cdot |qb| \leq t \cdot |ab|$. Bridging points guarantee a t -path for (a, b) if (p, q) is an edge and the pairs (a, p) and (q, b) already have t -paths. Note that $|ap|, |qb| < |ab|$ as a consequence.

We say that (p, q) is mandatory if the ellipse with foci p and q and eccentricity $1/t$ including its border contains no points in P other than p and q . Any t -path between p and q must fully lie within this ellipse, so a mandatory (p, q) will be in E for any t -spanner E .

Let $\lambda \in \mathbb{R}^+$. We say that a point $a \in P$ is λ -bridged if for all $b \in P$ with $|ab| > \lambda$, there exist some mandatory pair of points $(p, q), p, q \in P$, bridging (a, b) . We say that the point set P is λ -bridged if all points in P are λ -bridged. We say a point $a \in P$ is locally- λ -bridged if it is λ -bridged using only mandatory bridging pairs of points with distance most λ from a . A point set P is locally- λ -bridged if all points in P are locally- λ -bridged. Lemma 1 shows the usefulness of this concept.

Lemma 1 *Let P be a set of points that is λ -bridged. For any Euclidean graph $G = (P, E)$ it holds that G is a t -spanner if and only if all pairs of points $(a, b), a, b \in P$, with $|ab| \leq \lambda$ have a t -path in G .*

Proof Follows by induction over all pairs of points (a, b) with $|ab|$ ascending and earlier observations. □

As a warm-up exercise, we now develop a sufficient geometric condition for bridging pairs of points.

Lemma 2 *Suppose we are given points $a, b \in P$, rectangles R_1 and R_2 and $t > 1$, such that (as per Fig. 2): R_1 and R_2 lie in between a and b , have a side parallel to ab , have their centers on line segment ab , both have width w and height h , are separated by $s \geq \frac{t+1}{t-1}h$ and R_1 lies closer to a than R_2 .*

Then, for any $p, q \in P$ with p lying in R_1 and q lying in R_2 , (p, q) bridges (a, b) .

Proof Let u be the projection of p onto the line piece ab , and v the same for q . We have $|uv| \geq s \geq \frac{t+1}{t-1}h$, so $h \leq \frac{t-1}{t+1}|uv|$, which leads to the lemma using the triangle inequality as follows:

$$\begin{aligned} t|ap| + |pq| + t|qb| &\leq t(|au| + |up|) + |pq| + t(|qv| + |vb|) \\ &\leq t\left(|au| + \frac{1}{2}h\right) + (|uv| + h) + t\left(\frac{1}{2}h + |vb|\right) \\ &= t(|au| + |vb|) + (t + 1)h + |uv| \end{aligned}$$

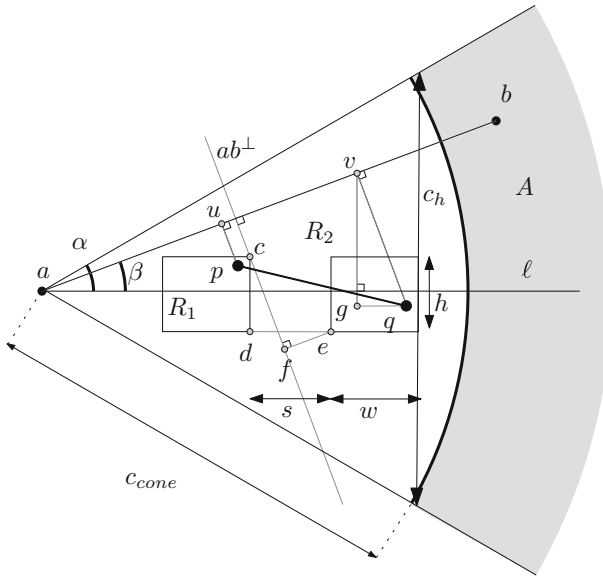


Fig. 3 The situation of Lemma 3

$$\begin{aligned}
 &\leq t(|au| + |vb|) + (t + 1)\frac{t - 1}{t + 1}|uv| + |uv| \\
 &= t(|au| + |uv| + |vb|) \\
 &= t|ab|
 \end{aligned}$$

□

We will now prove a stronger statement than Lemma 2 (albeit with a slightly stricter bound on s) using a similar proof method. We will use this statement to prove the main Theorem 1. Let $a, p, q \in P$ be pairwise different points and let region $A \subseteq \mathbb{R}^2$ with $a, p, q \notin A$. We say that the pair (p, q) bridges (a, A) if for every point $b \in P$ with $b \in A$ we have that (p, q) bridges (a, b) .

Lemma 3 Assume we are given $a \in P$, a line ℓ through a , an angle $\alpha \leq \pi/4$, rectangles R_1 and R_2 and $t > 1$, such that (as per Fig. 3): R_1 and R_2 have width w and height h , are separated by s , have a side parallel to ℓ , have their centers on ℓ , R_1 lies between a on the left and R_2 on the right and b lies to the right of the right side of R_2 .

Define c_w as the distance from a to the right side of R_2 . For the cone with apex a , angle 2α and bisector ℓ , we define A as the area that is at least $c_{cone} = c_w + h/2$ away from a and c_h as the length of the line piece perpendicular to ℓ going through the right side of R_1 that start and ends at the border of the cone. If $s \sin(\arctan(\frac{s}{h}) - \alpha) \geq \frac{\sqrt{2t+1}\sqrt{2}}{t-1}(c_h + h)$, then for any $p, q \in P$ with p lying in R_1 and q lying in R_2 , (p, q) bridges (a, A) . Note that this is true regardless of whether R_1 and R_2 lie fully within the cone.

Proof We will first give some definitions, then prove two bounds and use these to make the final derivation.

We now give some definitions resulting in the situation found in Fig. 3. Let $b \in A$ and $b \in P$. Let u be the projection of p onto the line piece ab , and v the same for q . Let c be the corner of R_1 lying closest to b . Let d be the corner of R_1 that is the mirror of c in ℓ , and let e be the corner of R_2 closest to d . Let ab^\perp be the line going through c that is perpendicular to the line ab . Let f be the projection of e onto ab^\perp .

We continue to prove two bounds: we wish to show that $(t + 1)\sqrt{2}(c_h + h) \leq (t - 1)|uv|$ and that $|up|, |vq| \leq \frac{\sqrt{2}}{2}(h + c_h)$.

We first bound $|uv|$ from below, by using that $|uv| \geq |ef|$. Note that $|cd| = h$, $|de| = s$, so $\angle dce = \arctan(\frac{s}{h})$. Define $\beta = \angle fcd$ (note that β is the angle between ab and ℓ). We therefore have $\angle fce = \arctan(\frac{s}{h}) - \beta$. Using that $|ce| = \sqrt{h^2 + s^2}$, we get $|ef| = |ce| \sin(\angle fce) = \sqrt{h^2 + s^2} \sin(\arctan(\frac{s}{h}) - \beta)$.

We therefore have $|uv| \geq \sqrt{h^2 + s^2} \sin(\arctan(\frac{s}{h}) - \beta) \geq s \sin(\arctan(\frac{s}{h}) - \alpha)$. By the assumption in the lemma, we get $|uv| \geq \frac{\sqrt{2}t + \sqrt{2}}{t - 1}(c_h + h)$, which implies $(t + 1)\sqrt{2}(c_h + h) \leq (t - 1)|uv|$. This proves our first bound.

Next, we prove that $|up|, |vq| \leq \frac{\sqrt{2}}{2}(h + c_h)$: we prove the case for $|vq|$, the case for $|up|$ is analogous. Let g be the point on the line perpendicular to ℓ going through v which is placed such that $\angle vgq$ is right. As v lies inside the cone and within distance c_{cone} from a , its distance to ℓ is at most $c_h/2$. Combined with the fact that g lies at most $h/2$ from ℓ we have $|vg| \leq \frac{h + c_h}{2}$. Next we observe that $|vq| \cos(\beta) = |vg|$, and since $\beta \leq \pi/4$ and therefore $\cos(\beta) \geq \frac{1}{\sqrt{2}}$ we get $|vq| \frac{1}{\sqrt{2}} \leq \frac{h + c_h}{2}$. Dividing both sides by $\frac{1}{\sqrt{2}}$ gives us our second bound. Using these bounds we can prove the lemma as follows.

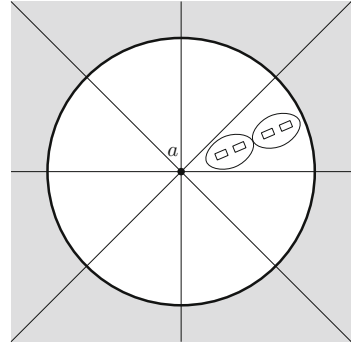
$$\begin{aligned}
 t|ap| + |pq| + t|qb| &\leq t(|au| + |up|) + |pq| + t(|qv| + |vb|) \\
 &= t(|au| + |vb|) + |pq| + t(|qv| + |up|) \\
 &\leq t(|au| + |vb|) + |pu| + |uv| + |vq| + t(|up| + |qv|) \\
 &= t(|au| + |vb|) + |uv| + (t + 1)(|up| + |qv|) \\
 &\leq t(|au| + |vb|) + |uv| + (t + 1)\sqrt{2}(h + c_h) \\
 &\leq t(|au| + |vb|) + |uv| + (t - 1)|uv| \\
 &= t(|au| + |vb|) + t|uv| \\
 &= t|ab|
 \end{aligned}$$

We now have the tool needed for the main result.

3 Uniform Point Sets

Theorem 1 *There exists c_t dependent only on t such that for every $c > 0$, if P is a set of points uniformly and independently distributed at random in a $\sqrt{n} \times \sqrt{n}$ square*

Fig. 4 Covering the plane with cones



and n is large enough, then with probability at least $1 - n^{-c}$, P is locally- $(c \cdot c_t \log n)$ -bridged.

We first give a high level overview of the proof. In Sect. 3.2 the full proof is given.

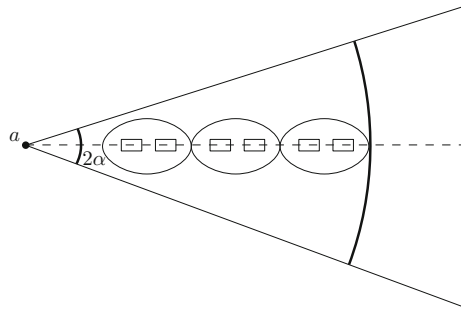
3.1 The Main Idea of the Proof

We need to prove that every point in P is locally- $(c \cdot c_t \log n)$ -bridged simultaneously with high probability. We show that every point individually is locally- $(c \cdot c_t \log n)$ -bridged with sufficiently high probability that a simple union bound shows that it will happen to all points simultaneously with high probability. We will use Lemma 3 to achieve this. For ease of presentation, we assume t is a constant in this proof sketch.

The rectangles in Lemma 3 can be chosen to have a roughly constant chance of containing a point and, as we will show, we can fulfill the other requirements of this lemma, the resulting pair of points bridges a relatively large part of \mathbb{R}^2 . In fact, we need only $\lceil \pi/\alpha \rceil$ cones to cover the area we wish to cover, as depicted in Fig. 4. We show the likely existence of a pair of mandatory points that bridges a single cone and use a union bound to show such pairs are likely to exist for all cones simultaneously. Note that this union bound allows us to ignore all dependency issues between cones.

We will place $O(\log n)$ pairs of rectangles in every cone as depicted in Fig. 5. If any pair of boxes ends up containing a point per box, these two points will satisfy the requirements for Lemma 3. We just need this pair of points to be mandatory, and therefore consider an ellipse around such a pair of boxes (defined in terms of the boxes, not the points, for easy analysis), such that if this ellipse is empty apart from these two points, these points must be mandatory. Using careful analysis, the chance that a pair of boxes contains one point per box and the ellipse contains no more points (an event we will call a ‘success’) is at least some constant p (dependent only on t). We need only one success per cone and the events are nearly independent (the ellipses don’t overlap), so the chance that we get at least one success is at least (roughly) $p^{O(\log n)} = n^{-O(f(t))}$ (with $f(t)$ a function depending only on t), which then shows the theorem.

Fig. 5 Rectangle configuration in a cone (not to scale)



3.2 The Full Proof

Note that we will often introduce a constant (say, the height h of R_1), give it a value (say $h = 1$) but still refer to the name of the variable later for clarity (so h instead of 1).

3.2.1 Positioning the Cones

Let c be given as per the theorem. Let $k = (c + 2) \frac{32}{10} e^{\frac{133r^4}{(t-1)^2}}$. Let $S'_{major} = \frac{9t^2 + 4t}{2(t-1)}$ (we derive this number later). Let c_{max} be the space needed to put all ellipses in a cone next to each other, i.e. $c_{max} = k \log n \cdot S'_{major}$. We partition the disk with radius $c_{cone} = c_{max} + 1/2$ around every point a into m cones, as depicted in Fig. 4. We choose m and then $\alpha = \frac{\pi}{m}$ such that the height of the cone at the start of the grey area as per Lemma 3 is 1. Because k is so large, we have that $m \geq 19$ - we don't need any more specifics on m or α other than this bound.

We will prove that P is very likely to be locally- $2c_{cone}$ -bridged, as per the theorem. We will do this by showing that every cone contains a mandatory pair of points that together bridge the gray area in Fig. 4. This gives us the following bound:

$$2c_{cone} = 1 + 2k \log n \cdot S'_{major} = 1 + 2(c + 2) \frac{32}{10} e^{\frac{133r^4}{(t-1)^2}} \log n \frac{9t^2 + 4t}{2(t-1)}$$

Recall that we place rectangles and ellipses in the area of cones colored white in Fig. 4. We first note that if a point lies very close to the edge of the $\sqrt{n} \times \sqrt{n}$ square in which our points lie, then for some of the cones we place, this white area may end up lying partially outside of this square, and so it is less likely that points end up in the rectangles we place there. We will deal with these cones now, and assume for the rest of the proof that these white areas lie fully within the square.

We divide the area around the origin a of such a boundary cone in three regions: A_{near} , $A_{between}$ and A_{far} based on their distance to a . A_{near} is the white area with distance at most c_{cone} from a that partially overlaps the square. A_{far} is the area more than $2c_{cone}$ away from a and $A_{between}$ lies between A_{near} and A_{far} . We want to show that A_{far} is bridged by some local pair of points. Note that the angle of the cones will end up being small (at most $\pi / 19$). We look at the (part of the) boundary between

A_{near} and $A_{between}$ that is outside the square. There are three cases: this part of the boundary consists of two pieces (the cone then points to a corner of the square), the part of the boundary is a single piece that touches a cone edge or the part is a single piece that does not touch the cone edge. For most of these cases, it is easy to see that A_{far} is completely outside the square: if the boundary has two pieces, the cone points to the corner of the square and so A_{far} is outside the square. If it has a single piece not touching the cone edge, it must be pointing (almost) perpendicularly to a square edge, and then A_{far} is outside the square again.

Now consider the case of the boundary touching the cone edge: in this case, the cone is (somewhat) parallel to a square edge, and we attempt to rotate the cone around its origin away from the square edge, until A'_{near} (defined the same as A_{near} but for the rotated cone) is completely within the square. There are again two cases: if the cone hits a different square edge during this rotation, then we note that we again must be near a corner of a square and again that A_{far} is not within the square. If it does not, then we note that A'_{far} (defined the same as A_{far} but for the rotated cone) now encompasses the part of A_{far} that is fully inside the square, and that A'_{near} is fully inside the square. We therefore simply assume we use this rotated cone for the rest of the proof, as this will obtain the same result.

3.2.2 Boxes in Cones

We first describe the rectangle and ellipse placement and show that this geometric setup will indeed allow us to invoke Lemma 3, which is needed to achieve local bridgedness. We also analyze the areas of the rectangles and ellipses involved, which will be useful later.

We place $k \log n$ rectangles R_1 and R_2 in every cone as per Lemma 3 and as depicted in Fig. 5. Every rectangle has width $w = 1$, height $h = 1$, and R_1 and R_2 are placed $s = \max\left(3\frac{t+1}{t-1}, 6\right)$ apart. The rectangles are aligned with the bisector ℓ of the cones. Neighboring pairs of rectangles are placed at a pitch of S'_{major} (so two neighboring left rectangles have a distance of S'_{major} between their centers). Let $A = w \cdot h = 1$ be the area of R_1 and R_2 .

We surround the rectangles by an ellipse \mathcal{E} with foci d and e and eccentricity $1/t$ such that the centers of R_1 and R_2 lie on de and d and e are placed at a distance $X = \frac{h+2w+th}{2(t-1)} = \frac{3+t}{2(t-1)}$ from R_1 and R_2 respectively.

We first show that both rectangles lie entirely in \mathcal{E} . Clearly, if the top left corner of R_1 (which we will call a for this proof) is in \mathcal{E} then both rectangles lie in \mathcal{E} due to their symmetric placement. Hence all we have to argue is that $|da| + |ae| \leq t|de|$. Using the triangle inequality we have $|da| \leq X + 0.5h$ and $|ae| \leq 0.5h + w + s + w + X$ which leads us to $h + 2w + 2X + s \leq t|de|$. As $|de| = 2X + s + 2w$ this clearly holds for our $w = h = 1$ if $t \geq 1.5$. To see why this holds for any $t > 1$ we rewrite the inequality to $(2w + 2X + s) + h \leq (t2w + 2tX + s) + (t - 1)s$. Clearly this inequality holds if $h \leq (t - 1)s$. Recall that $s = \max\left(3\frac{t+1}{t-1}, 6\right)$, which (for $t < 1.5$) we can simplify to $s = 3\frac{t+1}{t-1}$. Let $t = 1 + \epsilon$, with $0 < \epsilon < 0.5$, we have

$(t - 1)s = \epsilon s = \epsilon 3 \frac{(1+\epsilon)+1}{(1+\epsilon)-1} = 6 + 3\epsilon$. Since we picked $h = 1$ the inequality holds which proves that R_1 lies in \mathcal{E} .

We now show that if our ellipse \mathcal{E} is empty apart from two points (one inside R_1 and the other in R_2), then this pair of points is mandatory. We note that if $a \in R_1$ and $b \in R_2$, then any point f lying in the ellipse with foci a and b and eccentricity $1/t$ also lies in \mathcal{E} as follows: from $|af| + |fb| \leq t|ab| \leq t(h + 2w + s)$ we conclude $|df| + |fe| \leq |da| + |af| + |fb| + |be| \leq 2X + h + 2w + t(h + 2w + s) = t(2X + 2w + s) = t|de|$ (by filling in X and simplifying), and so $f \in \mathcal{E}$. However, we assumed that the ellipse was empty apart from a and b , hence such an f does not exist implying that (a, b) is mandatory.

The semi-major axis of the ellipse is given by:

$$\begin{aligned} S_{major} &= \frac{t|de|}{2} = \frac{t(2X + 2w + s)}{2} \\ &= \frac{t \left(\frac{3+t}{t-1} + 2 + \max \left(3 \frac{t+1}{t-1}, 6 \right) \right)}{2} \\ &= t \left(\frac{3+t}{2(t-1)} + 1 + \max \left(\frac{3t+3}{2(t-1)}, 3 \right) \right) \\ &= t \left(\max \left(\frac{3t+3}{2(t-1)} + \frac{3+t}{2(t-1)} + 1, \frac{3+t}{2(t-1)} + 4 \right) \right) \\ &= t \left(\max \left(\frac{6t+4}{2(t-1)}, \frac{9t-5}{2(t-1)} \right) \right) \\ &= \max \left(\frac{6t^2+4t}{2(t-1)}, \frac{9t^2-5t}{2(t-1)} \right) \\ &\leq \frac{9t^2+4t}{2(t-1)} = S'_{major} \end{aligned}$$

The semi-minor axis can be expressed in terms of S_{major} and the eccentricity:

$$\begin{aligned} S_{minor} &= S_{major} \sqrt{1 - \left(\frac{1}{t}\right)^2} \\ &\leq S'_{major} \sqrt{1 - \left(\frac{1}{t}\right)^2} \end{aligned}$$

This allows us to calculate the area of the ellipse:

$$\begin{aligned} |\mathcal{E}| &= \pi S_{major} S_{minor} \\ &\leq \pi S'_{major} S_{minor} \\ &= \pi \left(\frac{9t^2+4t}{2(t-1)} \right)^2 \sqrt{1 - \left(\frac{1}{t}\right)^2} \end{aligned}$$

$$\begin{aligned}
&\leq \pi \left(\frac{9t^2 + 4t}{2(t-1)} \right)^2 \\
&= \pi \left(\frac{81t^4 + 72t^3 + 16t^2}{4(t-1)^2} \right) \\
&\leq \frac{\pi}{4} \left(\frac{(81 + 72 + 16)t^4}{(t-1)^2} \right) \\
&\leq \frac{133t^4}{(t-1)^2}
\end{aligned}$$

If exactly one point ends up in R_1 , and exactly one point ends up in R_2 and no other point ends up in \mathcal{E} (making the pair of points mandatory), then we say that this pair of rectangles is a *success*. We now show that the pair of points corresponding to a success would fit the conditions of Lemma 3 and would therefore bridge the cone we are considering.

First, consider the requirement $s \sin(\arctan(\frac{s}{h}) - \alpha) \geq \frac{\sqrt{2t+\sqrt{2}}}{t-1}(c_h + h)$. Filling in known values, we get $s \sin(\arctan(s) - \alpha) \geq 2\frac{\sqrt{2t+\sqrt{2}}}{t-1}$. Now, by $m = 19$, we get $\alpha = \pi/m \leq 1/6$. Also, $s \geq 6$ by definition, so we get $\sin(\arctan(6) - 1/6) \geq 0.943$. Finally, $2\frac{\sqrt{2t+\sqrt{2}}}{t-1} = 2\sqrt{2}\frac{t+1}{t-1} = 2\sqrt{2}\frac{s}{3}$. Given that $0.943 \geq 2\sqrt{2}\frac{1}{3} (\approx 0.9428)$, we find that the requirement is met. The Lemma also requires that the angle of the cones is at most $\pi/4$, which follows from $m \geq 8$.

We have now shown that our geometric setup is able to prove local bridgeness by fulfilling the conditions of Lemma 3.

3.2.3 Probability of Success

We will show next that we will have at least one success for a given cone with high probability. Using the union bound we can then ignore all dependency issues between cones, such as ellipses overlapping with neighboring cones, and yet still obtain that all cones will simultaneously have at least one success.

We consider the chance that no ellipses yield a success for a given cone. Unfortunately, the chance that a given ellipse is a failure or not is dependent on whether other ellipses are failures or not, so we cannot simply consider every ellipse in isolation. For example, an ellipse can be a failure because *all* n points ended up in that ellipse, thus making the chance that any other ellipse is a success equal to 0.

We first narrow down the number of points that end up in an ellipse. The expected number of points ending up in an ellipse is $\mu = |\mathcal{E}|$. By a Chernoff bound, the chance that we have more than $\mu(1 + \delta)$ points in an ellipse is (with high probability) at most $e^{-\frac{\delta|\mathcal{E}|}{3}}$. We fill in $\delta = \ln n \frac{3k+3}{|\mathcal{E}|}$, giving at most $|\mathcal{E}| + \ln n(3k+3)$ points in an ellipse. Since we eventually want this to happen for all ellipses simultaneously, we apply the union bound and multiply by $k \log n$ (and using that $k \log n \geq n$ for sufficiently large n), giving:

$$\begin{aligned}
 k \log ne^{-\frac{\delta|\mathcal{E}|}{3}} &= k \log ne^{-\frac{3k+3}{|\mathcal{E}|} \ln n|\mathcal{E}|} \\
 &= k \log ne^{-(k+1) \ln n} \\
 &= k \log nn^{-(k+1)} \\
 &\leq n^{-k}
 \end{aligned}$$

At the ‘cost’ of an extra factor n^{-k} , we will now assume any ellipse in the cone contains at most $|\mathcal{E}| + \ln n(3k + 3)$ points, regardless of whether any ellipse in that cone is a success or not.

We number the ellipses from 1 to $k \log n$ and consider the chance P (ellipses 1 through $k \log n$ are failures). By definition of conditional probability, this equals P (ellipse $k \log n$ is a failure | ellipses 1 through $k \log n - 1$ are failures) · P (ellipses 1 through $k \log n - 1$ are failures). Applying this definition $k \log n$ times in total, this chance equals $\prod_{i=1}^{k \log n} P$ (ellipse i is a failure | ellipses 1 through i are failures). Let $p_i = P$ (ellipse i is a failure | ellipses 1 through i are failures). We are effectively interested in the chance $\prod_{i=1}^{k \log n} p_i$.

We now look to the $1 - p_i$ chances (so the chance of a success). We will be able to give a lower bound for all these chances with a similar formula structure, which we use to obtain a single lower bound for all these chances by filling in the ‘worst case’ value for every variation point in these lower bounds, to get a simple end result. First, as a warm-up, let’s calculate p_1 , the chance that the first ellipse is a failure regardless of whether the other ellipses are failures. Since both rectangles lie entirely in \mathcal{E} , we have the following:

$$1 - p_1 = \binom{n}{2} \left(\frac{|A|}{n}\right)^2 \left(1 - \frac{|\mathcal{E}|}{n}\right)^{n-2}$$

This equation is a straightforward application of probability theory. For a success, we must have exactly one point end up in both rectangles, and all other points must end up outside the ellipse. Since the locations of the points are independent (as opposed to the number of points falling in particular areas), we obtain a factor $\left(\frac{|A|}{n}\right)^2$ for the chance that two points end up in the square, and a factor $\left(1 - \frac{|\mathcal{E}|}{n}\right)^{n-2}$ for the chance that all other points end up outside the ellipse. The factor $\binom{n}{2} = n(n - 1)/2$ accounts for the number of ways we can pick two points to land in the rectangles.

Now, to bound $1 - p_i$, we take the same basic formula, but we now consider the more general $1 - p_i$ case in which all ellipses before it have produced failures, and limit ourselves to an upper bound. In particular, we essentially look at the ‘worst case’ for every factor in the $1 - p_1$ formula.

The factor $\binom{n}{2}$ now becomes $\binom{n-k \log n(|\mathcal{E}| + \ln n(3k+3))}{2}$ since the reason that all the $k \log n$ previous ellipses (actually at most $i - 1$ ellipses, but we bound it for simplicity) have produced a failure may be that they were filled with the maximum of $|\mathcal{E}| + \ln n(3k + 3)$ points.

The factor $\left(\frac{|A|}{n}\right)^2$ stays the same. Although the $\frac{1}{n}$ factor here can conceivably be larger (a failure in another ellipse may have been caused by an empty rectangle, which would imply that the chance that a points falls into a different rectangle increases a tiny bit because it can no longer fall in that empty rectangle), it will definitely not be smaller for any $1 - p_i$ and any reason for the failure of the previous ellipses.

We change the factor $\left(1 - \frac{|\mathcal{E}|}{n}\right)^{n-2}$ to $\left(1 - \frac{|\mathcal{E}|}{n-k \log n|\mathcal{E}|}\right)^n$. As $\left(1 - \frac{|\mathcal{E}|}{n-k \log n|\mathcal{E}|}\right) < 1$, setting the exponent to n only decreases the chance, which is allowed. The nominator in the fraction becomes smaller as we may be in the case where the reason that all previous ellipses failed is that they were all empty, which influences where these points may end up instead.

Using the previous arguments and using the well-known inequality $(1 + \frac{x}{n})^n \geq e^x(1 - \frac{x^2}{n})$, we now bound our expression bounding p_i . We use that $n - \frac{\log^2 n(|\mathcal{E}| + \ln n(3k+3))}{\ln 2} \geq \frac{n}{\sqrt{2}} + 1$ and $\frac{n}{k \log n} - |\mathcal{E}| \geq n/2$ for sufficiently large n .

$$\begin{aligned} 1 - p_i &\geq \binom{n - k \log n(|\mathcal{E}| + \ln n(3k + 3))}{2} \left(\frac{|A|}{n}\right)^2 \left(1 - \frac{|\mathcal{E}|}{n - k \log n|\mathcal{E}|}\right)^n \\ &\geq \binom{\frac{n}{\sqrt{2}} + 1}{2} \left(\frac{|A|}{n}\right)^2 \left(1 - \frac{|\mathcal{E}|}{n - k \log n|\mathcal{E}|}\right)^n \\ &\geq \frac{n^2}{4} \left(\frac{|A|}{n}\right)^2 \left(1 - \frac{|\mathcal{E}|}{n - k \log n|\mathcal{E}|}\right)^n \\ &= \frac{|A|^2}{4} \left(1 - \frac{|\mathcal{E}|}{n - k \log n|\mathcal{E}|}\right)^n \\ &= \frac{|A|^2}{4} \left(1 - \frac{|\mathcal{E}|}{n - k \log n|\mathcal{E}|}\right)^{n-k \log n|\mathcal{E}|} \left(1 - \frac{|\mathcal{E}|}{n - k \log n|\mathcal{E}|}\right)^{k \log n|\mathcal{E}|} \\ &\geq \frac{|A|^2}{4} e^{-|\mathcal{E}|} \left(1 + \frac{|\mathcal{E}|^2}{n - k \log n|\mathcal{E}|}\right) \left(1 - \frac{|\mathcal{E}|}{n - k \log n|\mathcal{E}|}\right)^{k \log n|\mathcal{E}|} \end{aligned}$$

As $\left(\frac{|\mathcal{E}|^2}{n-k \log n|\mathcal{E}|}\right)$ goes to 0 and we can bound $\left(1 - \frac{|\mathcal{E}|}{n-k \log n|\mathcal{E}|}\right)^{k \log n|\mathcal{E}|} \geq \frac{1}{2}$ for large enough n we have

$$\begin{aligned} 1 - p_i &\geq \frac{|A|^2}{4} e^{-|\mathcal{E}|} \left(1 - \frac{|\mathcal{E}|}{n - k \log n|\mathcal{E}|}\right)^{k \log n|\mathcal{E}|} \\ &\geq \frac{|A|^2}{4} e^{-|\mathcal{E}|} \frac{1}{2} \\ &\geq \frac{1}{8} |A|^2 e^{-|\mathcal{E}|} \end{aligned}$$

Using the union bound to couple the p_i chances with the events that not too many points end up in ellipses, we get that the chance that all ellipses in a cone produce failures is at most

$$\begin{aligned}
 n^{-k} + p_i^{k \log n} &\leq n^{-k} + \left(1 - \frac{1}{8}|A|^2 e^{-|\mathcal{E}|}\right)^{k \log n} \\
 &= n^{-k} + n^{k \log\left(1 - \frac{1}{8}|A|^2 e^{-|\mathcal{E}|}\right)}
 \end{aligned}$$

We concentrate on the logarithm, and fill in the values for $|A|$ and $|\mathcal{E}|$:

$$\log\left(1 - \frac{1}{8}e^{-\frac{133r^4}{(t-1)^2}}\right) \leq -\frac{1}{8}e^{-\frac{133r^4}{(t-1)^2}}$$

Going back to the bound:

$$n^{-k} + n^{-k\frac{1}{8}e^{-\frac{133r^4}{(t-1)^2}}} \leq n^{-k\frac{10}{32}e^{-\frac{133r^4}{(t-1)^2}}}$$

3.2.4 Conclusion

We now apply the union bound again, to ensure that we have a success for all cones simultaneously. Using that $m \log n \leq n$ holds for sufficiently large n , we have that the chance that one of the cones of any point produces only failures is therefore at most the following:

$$\begin{aligned}
 (mn \log n)n^{-k\frac{10}{32}e^{-\frac{133r^4}{(t-1)^2}}} &\leq n^{2-k\frac{10}{32}e^{-\frac{133r^4}{(t-1)^2}}} \\
 &\leq n^{-c}
 \end{aligned}$$

The last step follows from our definition of k . The theorem is now proven. □

4 Algorithms

We first introduce three tools used in the results below. Let c and c_t be as in Theorem 1 throughout this section. The first tool is that we can divide the input into a $\frac{\sqrt{n}}{c \cdot c_t \log n} \times \frac{\sqrt{n}}{c \cdot c_t \log n}$ grid in $O(n \log n)$ time, with every cell containing in expectation $O((c \cdot c_t \log n)^2)$ points.

The second tool is the ‘local’ Dijkstra algorithm. It determines for all points at most λ away from a source point s whether it has a t -path to s and if so, their network distance. It differs from the standard Dijkstra algorithm in that it only adds the points to the queue if they are at most λt away from the source s and only considers the edges E_s that have such a point as either endpoint. Using the grid to quickly find these points this can be done in $O((\lambda^2 + |E_s|) \log \lambda)$ expected time.

The third tool is called *path-hyperbola*. It is an area given by an origin point $u \in P$, a focus $v \in P$ and an edge set E . When using G to denote the graph $G = (P, E)$ we define the path hyperbola as $PH(u, v, E) = \{a \in \mathbb{R}^2 \mid \delta_G(u, v) + t \cdot |va| \leq t \cdot |ua|\}$. Obviously, if (p, q) bridges (a, b) , then $b \in PH(a, q, E)$ for every edge set E with

t -paths for pairs of points (u, v) with $|uv| \leq |ab|$, making path-hyperbola at least as powerful as bridging points for guaranteeing t -paths.

If we perform a local Dijkstra on s , we either find pairs of points without t -path, or we find a set of network distances that induce a set of path-hyperbola. If s is locally- λ -bridged, the union of path-hyperbola will be a superset of the area more than λ away from s . This union can be computed in $O(\lambda^2 \log \lambda)$ expected time: using polar coordinates, the union corresponds to a lower envelope. Since the hyperbolas pairwise intersect at most twice, this envelope has linear complexity and can be computed in $O(n \log n)$ time [4, 23]. This gives an efficient test of t -paths from s to all other points as least as accurate as local- λ -bridgedness.

4.1 Testing t -Spanners

The first application of Theorem 1 and our tools is a faster algorithm to test if a Euclidean graph is a t -spanner on uniformly distributed point sets. To the best of our knowledge, this leads to the first subquadratic algorithm for this problem on any interesting class of point sets not making assumptions on E .

Theorem 2 *There is an algorithm that, given a point set P whose points are uniformly distributed in a $\sqrt{n} \times \sqrt{n}$ square and a Euclidean graph $G = (P, E)$ on P , checks if G is a t -spanner using $O((n + |E|)(c_t \log n)^2 \log(c_t \log n))$ expected time, where c_t is a constant dependent only on t .*

Proof Applying our three tools with $\lambda = c \cdot c_t \log n$ almost immediately gives us the desired result: we run a local Dijkstra for every point, maintaining the union of the path hyperbola. If we find any pair of points without t -path, we return that the input is not a t -spanner. If some union of path-hyperbola for a point s does not cover the area more than λ away from s , we perform an $O(n^2 \log n)$ test for t -spannerness, and otherwise we return that the input is a t -spanner, which happens with high probability by Theorem 1. This algorithm therefore uses $O((n + |E|)(c_t \log n)^2 \log(c_t \log n))$ expected time.

4.2 Greedy Spanner

Algorithm *GreedySpannerOriginal*(V, t)

1. $E \leftarrow \emptyset$
2. **for** every pair of distinct points (u, v) in ascending order of $|uv|$
3. **do if** $\delta_{(V, E)}(u, v) > t \cdot |uv|$
4. **then** add (u, v) to E
5. **return** E

Consider the original greedy algorithm above as introduced in [17]. The graph returned by this algorithm is called the *greedy spanner* on V for t and it is obviously a t -spanner, but the algorithm has an $O(n^3 \log n)$ running time. (Note that more complex algorithms to compute the same spanner in near quadratic time exist). We make the following observation:

Lemma 4 *If P is λ -bridged, then the greedy spanner on P does not have edges longer than λ .*

Proof After ensuring t -paths for all (u, v) with $|uv| \leq \lambda$ the algorithm will not add more edges as all (u, v) with $|uv| > \lambda$ have t -paths by Lemma 1. \square

We can combine Lemma 4 with Theorem 1 to quickly compute the greedy spanner on uniform point sets. We first give a preliminary algorithm which we then employ in two greedy spanner algorithms.

Theorem 3 *For every $\lambda > 0$, there is an algorithm that, given a point set P whose points are uniformly distributed in a $\sqrt{n} \times \sqrt{n}$ square, computes in $O(n \log n + n\lambda^2 \log^2 \lambda)$ expected time the edges of length at most λ in the greedy spanner on P with dilation t .*

Proof We use the algorithm introduced in [3]. This algorithm is able to compute the greedy spanner in linear space by initially working on well-separated pairs instead of single edges. It first computes the well-separated pair decomposition and sorts the linear number of pairs instead of the quadratic number edges. It then considers the well-separated pairs by length sorting the small set of edges in similar sized pairs to ensure that the correct edge is added. For a well-separated pair $\{A_i, B_i\}$ we use $\min(A_i, B_i)$ and to denote the distance between the closest pair of points a, b with $a \in A_i$ and $b \in B_i$. Our algorithm also initially considers well-separated pairs except we keep Lemma 4 in mind and use our local Dijkstra instead of a normal Dijkstra and only consider well-separated pairs $\{A_i, B_i\}$ with $\min(A_i, B_i) \leq \lambda$. Using the analysis in [3] and using that the greedy spanner has degree $O(1)$, we conclude that if m is the number of considered well-separated pairs, the running time of our modified algorithm is $O(n \log n + \lambda^2 \log \lambda \sum_{i=1}^m \min(|A_i|, |B_i|))$. We therefore need to bound

$$\sum_{i=1}^m \min(|A_i|, |B_i|) \leq \sum_{i=1}^m (|A_i| + |B_i|) = \sum_{a \in P} |\{\{A_i, B_i\} \mid a \in A_i \vee a \in B_i\}|.$$

For any $l \in \mathbb{R}$, a point p can only be in $O(1)$ well-separated pairs of length at most a constant factor higher or lower than l [9, Lemma4.6.1]. We can therefore partition the well-separated pairs containing p into $O(1)$ -sized sets of similar length. As the minimal length per set differs by at least a constant factor, we conclude $|\{\{A_i, B_i\} \mid a \in A_i \vee a \in B_i\}| = O\left(\log \frac{\max_i\{A_i, B_i\}}{\min_i\{A_i, B_i\}}\right)$ (here $\min(\{A_i, B_i\})$ and $\max(\{A_i, B_i\})$ denote the minimum and maximum distance respectively between a pair of points in the set $\{A_i, B_i\}$). This last expression is $O(\log \lambda)$ in expectation on uniform point sets, giving an expected running time of $O(n \log n + n\lambda^2 \log^2 \lambda)$. \square

Note that we could have adapted the algorithm from [5], but this algorithm sorts all potential edges, resulting in an expected $O(n \log n \lambda^2 \log \lambda)$ running time, which is slower when filling in $\lambda = O(\log n)$.

Combining Lemma 4, Theorems 1 and 3 (with $\lambda = c \cdot c_t \log n$) gives:

Corollary 1 *There is an algorithm that, given a point set P whose points are uniformly distributed in a $\sqrt{n} \times \sqrt{n}$ square, computes in $O(n(c_t \log n)^2 \log^2(c_t \log n))$ expected time a graph on P which is with high probability the greedy t -spanner (with c_t is a constant dependent only on t).*

4.3 The Full Distribution-Sensitive Algorithm

The algorithm from Theorem 3 is the first phase of our distribution-sensitive algorithm. We now present the second and third phase that ensure that all long edges are also computed. Note that it is relatively easy to prove Theorem 4 by chaining Theorem 3 and Theorem 2 and invoking an existing quadratic algorithm if the t -spanner test fails, but our aim is to provide an algorithm that ‘gradually’ slows down as the input is less uniformly distributed instead of tanking as soon as the property does not hold.

The second phase of our algorithm gathers path-hyperbola as described at the start of this section. We then consider the well-separated pairs that did not get considered in the first stage of the algorithm and try to prove for them that they will not produce a greedy spanner edge. For the remaining pairs, we employ the algorithm of [3] in the third phase of our algorithm to find the remaining greedy spanner edges.

If for a point $u \in A_i$, the bounding box B_i is covered by the union of path-hyperbola computed for u (testing this takes $O(\log n)$ time), then we say u is *discounted* with respect to $\{A_i, B_i\}$. If all $u \in A_i$ are discounted, then $\{A_i, B_i\}$ will not contain a greedy spanner edge and we say $\{A_i, B_i\}$ is *discounted*. This can be computed in $O(\log n \sum_{i=1}^m (|A_i| + |B_i|)) = O(n \log n \log \lambda)$ expected time (The $\log \lambda$ factor follows from the same argument as used in the proof of Theorem 3).

In the third phase we perform the algorithm from [3], with small differences. We ignore pairs that have been discounted in the previous phase, and we do not perform a Dijkstra operation on points which have been discounted with respect to that pair as well. By Theorem 1, all pairs are discounted with high probability and hence this phase takes constant time in expectation on uniform point sets.

In practice, using a λ lower than predicted by Theorem 1 will suffice and be faster. From experiments we observe that $\lambda = \frac{\log n}{\sqrt[4]{t-1} \log \log n}$ is the ‘right’ bound for the length of the longest edge in the greedy spanner. Using $1.1 \cdot \lambda$ the initial phase nearly always finds all edges, with the second phase usually discounting 99.7% of the pairs and 95% of the points in non-discounted pairs, with the second phase taking about 20% of the time of the first. Using $1.5 \cdot \lambda$, all pairs are typically discounted.

Theorem 4 *There is an algorithm that, given t and a point set P whose points are uniformly distributed in a $\sqrt{n} \times \sqrt{n}$ square, computes in $O(n(c_t \log n)^2 \log^2(c_t \log n))$ expected time its greedy spanner, with c_t a constant dependent only on t . The algorithm uses $O(n^2 \log^2 n)$ time on arbitrary P .*

5 Experimental Results

We have experimentally compared the performance of our algorithm and the two other linear space algorithms, WSPD-Greedy [3] and lazy-greedy [7]. For the comparison

we used point sets whose size ranged from 500 to 128,000 points. On small point sets the WSPD-Greedy algorithm has a running time comparable to the major quadratic space algorithms. Since running these quadratic space algorithms on more than 10,000 points quickly becomes unfeasible we did not include them in our experiments. For a detailed comparison between the major quadratic space algorithms and WSPD-Greedy we refer to [3]. Note that we have verified that all our implemented algorithms give the same output.

Throughout this section we will refer to our algorithm as “bucketing” in the graphs. We generated point sets according to several distributions. We have recorded space usage and running time (wall clock time). The results are averages over several runs where new point sets were generated each time. We included graphs for the uniform point set and for a clustered point set as these represent the best and worst cases respectively for our algorithm (with respect to our set of tests). To generate the clustered point set we used the same method as [3,7], that is, for n points, it consists of \sqrt{n} uniformly distributed point sets of \sqrt{n} uniformly distributed points.

5.1 Environment

The algorithms have been implemented in C++. The random generator used was the Mersenne Twister PRNG—we have used a C++ port by J. Bedaux of the C code by the designers of the algorithm, M. Matsumoto and T. Nishimura. We have implemented all other necessary data structures and algorithms not already in the `std` ourselves. The implementations do not use parallelism and run on a single thread.

Our experiments have been run on a server using an Intel Xeon E5530 CPU (2.40 GHz) and 8 GB (1600 MHz) RAM. It runs the Debian 7 OS and we compiled for 64 bits using G++ 4.7.2 with the `-O3` option.

5.2 Dependence on Instance Size

We have compared running time and space usage of the different linear space algorithms for different values of n . We plotted the results using $t = 2$ on both uniform (Fig. 6) and clustered points (Fig. 7).

The space usage of our algorithm appears to be a constant factor less than that of WSPD-Greedy. Its running time on uniformly distributed points is (nearly) linear making it a massive improvement over WSPD-Greedy. This allows us to calculate greedy spanners on such point sets in a matter of minutes where WSPD-Greedy would need hours or even days for bigger instances. When comparing to lazy-greedy we see that our memory requirements are bigger. The reduced memory requirements of lazy-greedy come at a cost of a significant performance hit, making the gap in running times even more extreme.

The clustered point set is a bad case for our algorithm since the greedy spanner will contain a considerable amount of very long edges between clusters. Nevertheless, the algorithm still beats both other algorithms by quite a margin. Our experiments on clustered data with smaller t values (up to $t = 1.1$ as plotted in Fig. 8) show that the performance of the algorithms gets more similar as t decreases. On point sets drawn

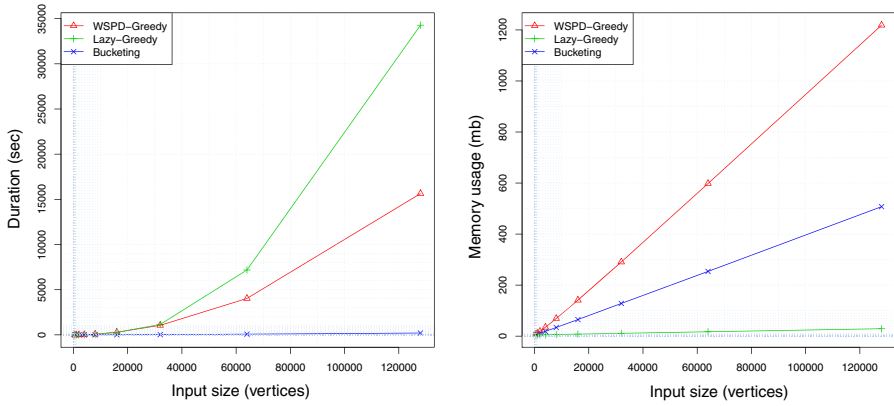


Fig. 6 The *left plot* the running time of our algorithm (bucketing) and the other linear space algorithms for $t = 2$ on variously sized uniformly distributed instances. The *right plot* the memory usage on the same data

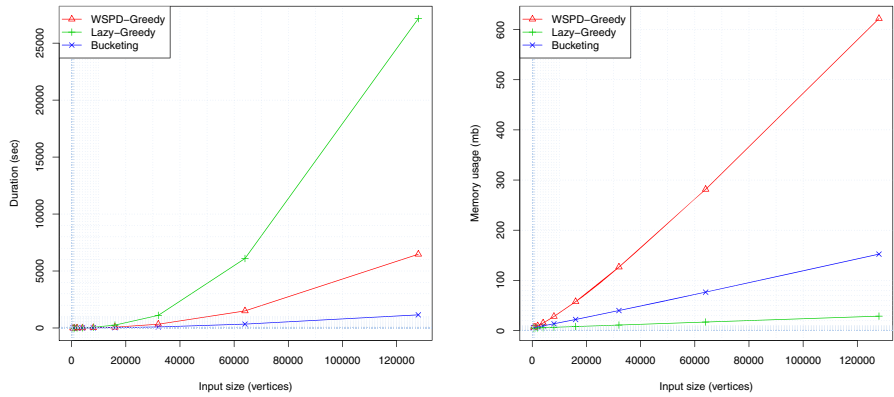


Fig. 7 The *left plot* the running time of the linear space algorithms for $t = 2$ on variously sized clustered instances. The *right plot* the memory usage on the same data

using either a uniform or normal distribution our algorithm massively outperforms both algorithms for both small and large t . Since the performance on point sets drawn from a normal distribution was very similar to the performance on uniform point sets we only plotted the latter (Figs. 6 and 9). Interestingly, the lazy-greedy algorithm becomes our closest competitor in the low t case on the non-clustered point sets. This is a result of the huge number of well-separated pairs on such point sets which are all processed by the WSPD-Greedy algorithm.

5.3 Real Data

Aside from generated instances we also experimented on some real point sets from the TSPLIB.¹ The performance of our algorithm on these real datasets seems to be close

¹ <http://comopt.ifl.uni-heidelberg.de/software/TSPLIB95/>.

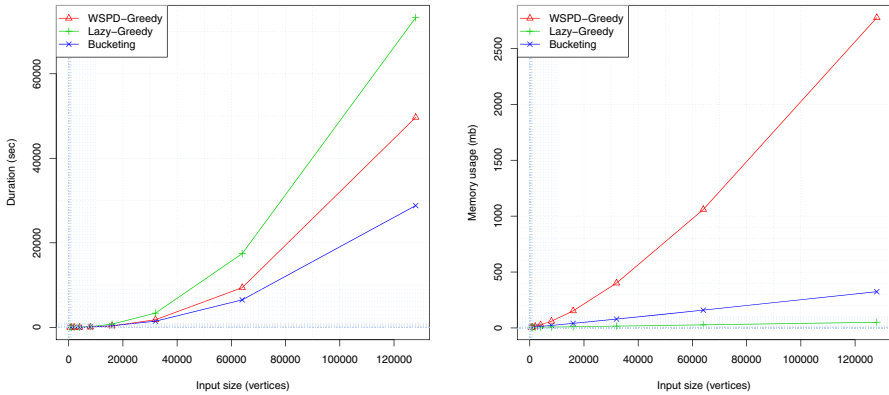


Fig. 8 The *left plot* the running time of our algorithm (bucketing) and WSPD-Greedy for $t = 1.1$ variously sized clustered instances. The *right plot* the memory usage on the same data

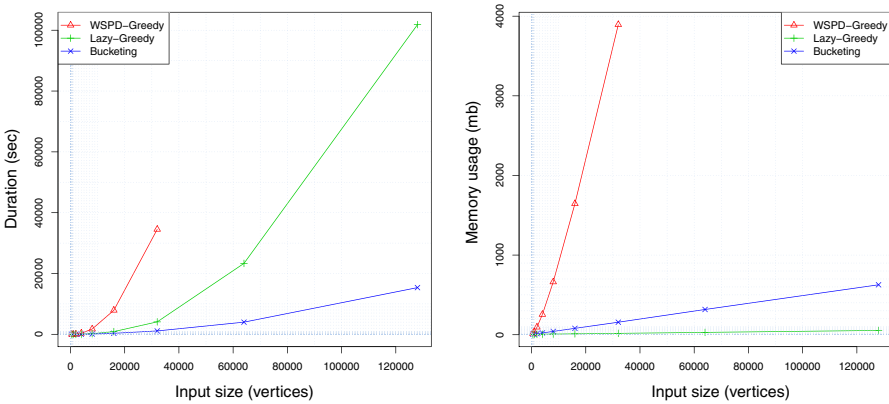


Fig. 9 The *left plot* shows the running time of our algorithm (bucketing) and WSPD-Greedy for $t = 1.1$ variously sized uniformly distributed instances. The *right plot* shows the memory usage on the same data

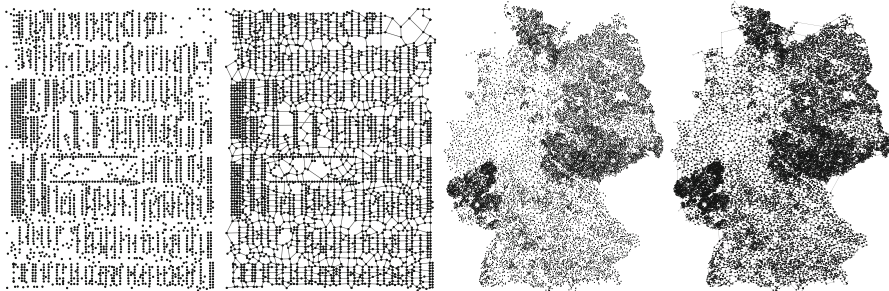


Fig. 10 Real point sets from the TSPLIB and their greedy spanners using $t = 2$. *Left* A PCB instance of 3,038 points. *Right* Cities in Germany, 15,112 points

to the uniform point sets. Figure 10 shows two point sets and their greedy spanners. For the PCB the computation using our algorithm took on average about 2 s for $t = 2$ and 11 s for $t = 1.1$. The same computations using WSPD-Greedy took 12 and 203 s

respectively and lazy-greedy needed 8.4 and 28 s. The bigger Germany instance took 21 and 147 s to compute using our algorithm while WSPD-Greedy needed 274 and 7486 s and lazy-greedy needed 255 and 795 s for $t = 2$ and $t = 1.1$. This is a factor 50 improvement for the low t case over WSPD-Greedy which suffers from the huge number of well-separated pairs at these low t values. The different approach taken by lazy-greedy makes it more efficient than WSPD-Greedy for low t , but it is still more than a factor 5 slower than our algorithm.

6 Conclusion

We have introduced a distribution-sensitive algorithm for computing the greedy spanner. Experiments show large improvements in both time and space for most data sets, while results are never worse than the state-of-the-art. The performance gap in many cases increases further for lower t . To explain these results, we have analyzed the algorithm on uniformly distributed point sets.

To this end, we have introduced the concept of *bridgedness* and have shown that point sets that are uniformly distributed in a $\sqrt{n} \times \sqrt{n}$ square are $O(\log n)$ -bridged with high probability. This implies that ‘ t -spannerness’ is a ‘local’ property on these point sets: a Euclidean graph is a t -spanner if and only if all pairs of ‘close-by’ points have t -paths. This locality shows that our algorithm runs in near-linear expected time on these point sets and yields a near-linear time algorithm for testing whether an edge set is a t -spanner on these point sets.

We leave several questions open that may be answered in future work. First, in our experiments, we have observed that the length of the longest edge of the greedy spanner on uniform point set tends towards $\log n / \log \log n / (t - 1)^{1/4}$, leaving a gap to our upper bound; similarly, our bridgedness bound may also be improvable. Second, it would be interesting to find an even faster algorithm on uniform point sets. Third, it would be interesting to see if our results generalize to higher dimensions. Lastly, there is still no general subquadratic time algorithm for the greedy spanner. One could consider our algorithm to be a divide and conquer algorithm where the conquer step may be very slow and possibly susceptible to improvement.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

References

1. Abam, M.A., de Berg, M., Farshi, M., Gudmundsson, J.: Region-fault tolerant geometric spanners. *Discrete Comput. Geom.* **41**(4), 556–582 (2009)
2. Agarwal, P.K., Klein, R., Knauer, C., Langerman, S., Morin, P., Sharir, M., Soss, M.: Computing the detour and spanning ratio of paths, trees, and cycles in 2D and 3D. *Discrete Comput. Geom.* **39**(1), 17–37 (2008)
3. Alewijnse, S.P.A., Bouts, Q.W., ten Brink, A.P., Buchin, K.: Computing the greedy spanner in linear space. *Algorithmica* **73**(3), 589–606 (2015)

4. Atallah, M.: Some dynamic computational geometry problems. *Comput. Math. Appl.* **11**, 1171–1181 (1985)
5. Bose, P., Carmi, P., Farshi, M., Maheshwari, A., Smid, M.: Computing the greedy spanner in near-quadratic time. *Algorithmica* **58**(3), 711–729 (2010)
6. Bose, P., Devroye, L., Evans, W., Kirkpatrick, D.: On the spanning ratio of Gabriel graphs and beta-skeletons. *SIAM J. Discrete Math.* **20**(2), 412–427 (2006)
7. Bouts, Q.W., ten Brink, A.P., Buchin, K.: A framework for computing the greedy spanner. In: Proceedings of 30th Symposium Computational Geometry, pp. 11–19. ACM (2014)
8. Buchin, K.: Constructing Delaunay triangulations along space-filling curves. In: Proceedings of 17th Annual European Symposium Algorithms (ESA), pp. 119–130. Springer (2009)
9. Callahan, P.B.: Dealing with Higher Dimensions: The Well-Separated Pair Decomposition and Its Applications. PhD thesis, Johns Hopkins University, Baltimore (1995)
10. Chew, L.P.: There are planar graphs almost as good as the complete graph. *J. Comput. Syst. Sci.* **39**(2), 205–219 (1989)
11. Devroye, L.: On the expected size of some graphs in computational geometry. *Comput. Math. Appl.* **15**, 53–64 (1988)
12. Devroye, L., Gudmundsson, J., Morin, P.: On the expected maximum degree of Gabriel and Yao graphs. *Adv. Appl. Probab.* **41**(4), 1123–1140 (2009)
13. Eppstein, D., Wortman, K.A.: Minimum dilation stars. *Comput. Geom.* **37**(1), 27–37 (2007)
14. Gao, J., Guibas, L.J., Hershberger, J., Zhang, L., Zhu, A.: Geometric spanners for routing in mobile networks. *IEEE J. Sel. Areas Commun.* **23**(1), 174–185 (2005)
15. Gudmundsson, J., Knauer, C.: Dilation and detours in geometric networks. In: Gonzales, T. (ed.) *Handbook on Approximation Algorithms and Metaheuristics*, pp. 52–1–52–16. Chapman & Hall/CRC, Boca Raton (2006)
16. Hellweg, F., Schmidt, M., Sohler, C.: Testing Euclidean spanners. In: Goldreich, O. (ed.) *Property Testing*, 6390 of LNCS, pp. 306–311. Springer, Berlin (2011)
17. Keil, J.M.: Approximating the complete Euclidean graph. In: Proceedings of 1st Scandinavian Workshop on Algorithm Theory (SWAT), volume 318 of LNCS, pp. 208–213. Springer (1988)
18. Mücke, E.P., Saias, I., Zhu, B.: Fast randomized point location without preprocessing in two- and three-dimensional Delaunay triangulations. In: Proceedings of 12th Symposium on Computer Geom. pp. 274–283. ACM (1996)
19. Narasimhan, G., Smid, M.: Approximating the stretch factor of Euclidean graphs. *SIAM J. Comput.* **30**(3), 978–989 (2000)
20. Narasimhan, G., Smid, M.: *Geometric Spanner Networks*. Cambridge University Press, New York (2007)
21. Peleg, D., Schäffer, A.A.: Graph spanners. *J. Graph Theory* **13**(1), 99–116 (1989)
22. Santi, P.: Topology control in wireless ad hoc and sensor networks. *ACM Comput. Surv.: CSUR* **37**(2), 164–194 (2005)
23. Sharir, M., Agarwal, P.: *Davenport–Schinzel Sequences and their Geometric Applications*. Cambridge University Press, Cambridge (1995)
24. Shpungin, H., Segal, M.: Near-optimal multicriteria spanner constructions in wireless ad hoc networks. *IEEE/ACM Trans. Netw.* **18**(6), 1963–1976 (2010)
25. Steele, J.M.: *Probability Theory and Combinatorial Optimization*, volume 69 of CBMS-NSF Regional Conference Series in Applied Mathematics. SIAM (1997)
26. Wang, Y., Li, X.-Y.: Efficient Delaunay-based localized routing for wireless sensor networks. *Int. J. Commun. Syst.* **20**(7), 767–789 (2007)
27. Xue, F., Kumar, P.R.: The number of neighbors needed for connectivity of wireless networks. *Wireless Netw.* **10**(2), 169–181 (2004)
28. Yukich, J.E.: Probability theory of classical Euclidean optimization problems. In: *Lecture Notes in Mathematics*, vol. 1675. Springer Berlin Heidelberg (1998)