The CSI Journal on
Computer Science and Engineering
Vol. 14, No. 1, 2016
Pages 6-18
Regular Paper

Improving Space and Time Complexity of the Gap-Greedy Spanner Algorithm

Davood Bakhshesh Mohammad Farshi

Department of Mathematical Sciences, Yazd University, Yazd, Iran

Abstract

Let S be a set of n points in $\mathbb{R}^d$ and G be a geometric graph with vertex set S. For a fixed real number $t > 1$, G is called a t-spanner of S if, for all pairs of vertices $u, v \in S$, the shortest path in G between u and v is no longer than $t|uv|$, where $|uv|$ is the Euclidean distance between u and v. In this paper, we present an algorithm to construct the gap-greedy spanner that takes $O(n^2)$ time and $O(n^2)$ space, and another variant of this algorithm that takes $O(n^3)$ time and $O(n)$ space. We also improve the (theoretical) dilation of the gap-greedy spanner. Moreover, we present some other $O(n^3)$ time algorithms to construct the gap-greedy spanner and compare their time complexity with the gap-greedy algorithm in practice. Furthermore, we experimentally compare the running time and some geometric properties of the gap-greedy spanner with the greedy spanner.

Keywords: Geometric Spanners, Gap-Greedy Spanner, Well-Separated Pair Decomposition, Greedy Spanner.

1. Introduction

Let S be a set of n points in $\mathbb{R}^d$ and let $t > 1$ be a real number. A geometric graph is an edge-weighted graph on $S \subseteq \mathbb{R}^d$ such that the weight of each edge is the Euclidean distance between its endpoints. A geometric graph G with vertex set S is called a t-spanner of S, if for each two points p and q in S, there exists a path Q in G between p and q whose length is at most t times $|pq|$, the Euclidean distance between p and q. The length of a path is defined to be the sum of the lengths, or weights, of all edges on the path. The path Q is called at-spanner path (or t-path) between p and q. We denote the length of a path Q by $|Q|$. The stretch factor (or dilation) of G is the smallest value of t for which G is a t-spanner. Analogously, a directed t-spanner of S is a directed geometric graph G with vertex set S such that for each two points p and q in S there exists a directed t-path from p to q. Spanners were introduced by Peleg and Schäffer [1] in the scope of distributed computing and, then, by Chew [2] in the scope of computational geometry. Spanners are applicable in many scopes such as graph theory, network topology design, distributed systems and robotics. We refer the reader to [3, 4, 5, 6, 7] for reading about the t-spanners and their applications. Also, one can see the major algorithms for efficient construction of spanners in the book by Narasimhan and Smid [6].

One of the basic spanner algorithms in the greedy-class of the geometric spanner algorithms is the gap-greedy algorithm that was introduced by Arya and Smid [8]. The algorithm is based on the gap property which was introduced by Chandra et al. [3]. Let $w > 0$ be a real number, and let E be a set of directed edges in $\mathbb{R}^d$, the set E satisfies the w-gap property if for any two distinct (directed) edges (p, q) and (r, s) in E, we have $|pr| > w \times min(|pq|, |rs|)$,and the set E satisfies the strong w-gap property if for any two (directed) distinct edges (p, q) and (r, s) in E, we have $|pr| > w \times min(|pq|, |rs|)$ and $|qs| > w \times min(|pq|, |rs|)$. The gap-property is useful for bounding the weight of graphs, where the weight of any edge set E, denoted by $wt(E)$, is the sum the Euclidean distance of all the edges in E. Chandra et al. [3] proved that if the edge set of (directed) geometric graph $G = (S, E)$ satisfies the gapproperty, then $wt(E) = O(\log n) \cdot wt(MST(S))$, where

Algorithm 1.1: GAPGREEDY(S, θ, w)

```
/* This algorithm takes as input a set S of n points in the plane, and two real
   numbers θ and w such that 0 < θ < π/4, 0 ⩽ w < (cos θ − sin θ)/2. The algorithm
   returns a directed t-spanner G = (S, E), for t = 1/(cos θ − sin θ − 2w).   */
Sort the 2(n choose 2) ordered pairs of distinct points in non-decreasing order of their distances (ties are
broken arbitrarily), and store them in a list L;
E := ∅;
foreach ordered pair (p, q) ∈ L do
    add := true;
    foreach edge (r, s) ∈ E do
        /* Let a⃗ be a vector from the origin to the point a. We define angle(pq, rs) to
           be the angle between the vectors q⃗ − p⃗ and s⃗ − r⃗.                  */
        if angle (pq, rs) ⩽ θ then
            add := add ∧(|pr| > w|rs|) ∧ (|qs| > w|rs|);
        end
    end
    if add = true then
        E := E ∪ (p, q);
    end
end
return G = (S, E);
```

Table 1. Construction time, memory usage and dilation for the gap-greedy spanner

Gap-Greedy Spanner				
		Construction Time	Memory Usage	Dilation
Previous work [8]		$O(n^3)$	$O(n^2)$	$\frac{1}{\cos\theta - \sin\theta - 2w}$
This Paper	First Algorithm	$O(n^3)$	$O(n)$	$\frac{1}{1 - 2\sin\left(\frac{\theta}{2}\right) - 2w}$
	Second Algorithm	$O(n^2)$	$O(n^2)$	

wt(MST(S)) is the weight of a minimum spanning tree on SThey also proved that if the edge set of a graph $G = (S, E)$ satisfies the strong w-gap property, then for every point v in S, the indegree and outdegree of v is at most one in G. In 1997, Arya and Smid [8] proposed the gap-greedy algorithm which constructs a spanner for every point set in the plane, that is called gap-greedy spanner, with constant small dilation whose edge set can be divided into a constant number of subsets such that each subset satisfies the strong gap property, see Algorithm 1.1. Hence, using the Chandra et al.'s results, it can be shown [8] that each vertex of the gap-greedy spanner generated by GAPGREEDY (S, θ, w)(see Algorithm 1.1) has degree at most $2\lceil 2\pi/\theta \rceil$. Moreover, if $w > 0$, then the weight of the gap-greedy spanner is less than $\lceil 2\pi/\theta \rceil(1 + 2/w)\log n$ times wt(MST(S)). Let θ and w be two real numbers such that $0 < \theta < \pi/4$ and $0 < w < (cos\theta - sin\theta)/2$. It can be proven that the graph$G = (S, E)$ generated by GAPGREEDY (S, θ, w) is a t-spanner for S, with $t = 1/(\cos\theta - \sin\theta - 2w)$ [8]. It is easy to see that the algorithm GAPGREEDY (S, θ, w) has the time complexity $O(n^3)$ and the space complexity $O(n^2)$. To the best of our knowledge, there is no $o(n^3)$ time algorithm for constructing the gap-greedy spanner. Arya and Smid [8] proposed an algorithm that called modified gap-greedy algorithm which for every set of n points in the plane, constructs a geometric spanner with (theoretical) properties similar to the gap-greedy spanner in $O(n\log^2 n)$ time using $O(n\log^2 n)$ space. The output of the modified gap-greedy algorithm, somehow, approximates the gap-greedy spanner. We emphasize that the output of the modified gap-greedy algorithm is not necessarily same as the output of gap-greedy algorithm.

Another interesting spanner is the greedy spanner (or path-greedy spanner), proposed by Althöfer et al. [9]. It has been shown that the weight, edge count and maximal degree of the greedy spanner are asymptotically optimal; see [6, 10]. To the best of our knowledge, the best known algorithm to construct the greedy spanner for a set of n points in $\mathbb{R}^d$ takes $O(n^2\log n)$ time; see [11].

Although the theoretical total weight of the greedy spanner is better than the theoretical total weight of the gap-greedy spanner, to the best of our knowledge, there is still no algorithm taking $o(n^2\log n)$ time to construct the greedy spanner. In this paper, we present an algorithm to construct the gap-greedy spanner takes $O(n^2)$ time using $O(n^2)$ space, and another variant of this algorithm that takes $O(n^3)$ time in linear space. To the best of our knowledge, this is the first algorithm to construct the gap-greedy spanner in $o(n^3)$ time. Furthermore, we improve the (theoretical) dilation $1/(\cos\theta - \sin\theta - 2w)$ of the gap-greedy spanner to $1/(1 - 2\sin(\theta/2) - 2w)$ (see table 1). It is notable that by this improvement on the dilation, the angle θ can be considered as a real number with $0 < \theta < \pi/3$ instead of$0 < \theta < \pi/4$. We can easily extend these results to higher dimensions. Note that throughout this paper, we will use the notation and terminology of [12]. In this paper, we also present three other cubic time algorithms to construct the gap-greedy spanner and compare their time complexity with Algorithm 1 in practice. Moreover, we experimentally compare the running time and some geometric properties of the gap-greedy spanner such as size (edge count), maximum degree, weight, with the greedy spanner. We hoped that we can find some point sets in the plane such that the geometric properties of the gap-greedy spanner and greedy spanner of the point sets are almost the same and the running time of computing the gap-greedy spanner is faster than the greedy spanner, but the experiments show that this is not correct.

2. Improving Space and Time Complexity

In this section, we propose an algorithm to construct the gap-greedy spanner in linear space and cubic time. Furthermore, using some modifications on this algorithm, we show how to construct the gap-greedy spanner in quadratic time using the quadratic space.

2.1. Improving the Space Complexity

Here, we show how to construct the gap-greedy spanner in linear space and cubic time. The construction is based on the well-separated pair decomposition and that is similar tothe algorithm which proposed by Alewijnse et al. [12] for constructing the greedy spanner in linear space. One of the useful data structures to efficiently solve the proximity problems is the well-separated pair decomposition (or WSPD) that introduced by Callahan and Kosaraju [13]. Let S be a set of n points in $\mathbb{R}^d$ and let $s > 0$ be a real number. A WSPD for S with separation ratio s is a set $\{(A_i, B_i)\}_{i=1}^{m}$ of pairs of nonempty subsets of S, for some integer m (m is called size of WSPD), such that (i) for each $1 \le i \le m$, A_i and B_i is s-well-separated i.e there are two disjoint d-dimensional balls C_A and C_B with the same radius, containing A_i and B_i, respectively such that the distance between C_A and C_B is greater than or equal to s times the radius of C_A and (ii) for every two distinct points $p, q \in S$ there is exactly one index i with $1 \le i \le m$ such that $p \in A_i$ and $q \in B_i$ or vice versa. Callahan and Kosaraju [13] proved that for every set $S \subseteq \mathbb{R}^d$ of n points and every separation ratio s, there exists a WSPD $(A_1, B_1), \dots, (A_m, B_m)$ of size $m = O(s^d n)$ that can be computed in $O(n\log n + s^d n)$ time and can be represented in $O(s^d n)$ space.

The well-separated pair decomposition has several applications. To see the applications of WSPD, we refer the reader to [6, 13, 14, 15].

Now, we briefly and intuitively describe our algorithm. The algorithm uses the property that the gap-greedy spanner includes at most a constant number of edges in each pair of WSPD (we show this later). So, the algorithm first constructs a WSPD with linear size that can be represented using linear space, then tries to find the edges of the gap-greedy spanner in each pair of WSPD. To this end, the algorithm increasingly sorts the well-separated pairs according to the minimum distance of the circles around of their bounding boxes. Then, for each well-separated pair, the algorithm finds a pair of points that is the closest pair with respect to a property that is defined later. We will see that if (u, v) be a pair with minimum Euclidean distance among all these closest pairs, then (u, v) and (v, u) are the edges of the gap-greedy spanner that the algorithm adds them to the edge set. After adding these edges into the edge set, the algorithm should update the closest pair of each well-separated pair that has been affected by adding theses edges. The main operation of the algorithm is updating some special well-separated pairs instead of updating all well-separated pairs. It is notable that since for sorting the WSPD a linear space in the memory is needed, the algorithm benefits this property to improve the space.

Before we describe our algorithm in details, we present some fundamental lemmas that are needed later. In the following, we assume that $t > 1$ is a constant real number.

Let S be a set of n points in the plane. Suppose that we sort the $2\binom{n}{2}$ ordered pairs of distinct points of S in non-decreasing order of their Euclidean distances and store them in a list L such that for each i with $0 \le i \le \binom{n}{2} - 1$, if $L[2i] = (p, q)$, then $L[2i + 1] = (q, p)$. We call $L[i]$ the ith entry of list L and for simplicity, we suppose that in each entry of L we store a pair of points.

Recall the gap-greedy algorithm (Algorithm 1.1). In the following, we suppose that in the first step of the gap-greedy algorithm, the list L is sorted with the above sorting. Now, let θ and w be two real numbers such that $0 < \theta < \pi/4$ and $0 \le w < (cos\theta - sin\theta)/2$. Let $G = (S, E)$ be the gap-greedy spanner generated by Algorithm 1.1 with respect to the parameters θ and w. Now, we have the following lemma.

Lemma 1. (q, p) is an edge of the gap-greedy spanner iff (p, q) is an edge of the gap-greedy spanner.

Proof. Let i with $0 \le i \le \binom{n}{2} - 1$ be an integer such $L[2i] = (p, q)$. To prove the lemma, it is sufficient to prove that $(q, p) \in L[2i + 1]$ if and only if $(p, q) \in L[2i]$. Here, we only prove that if $(p, q) \in L[2i]$, then $(q, p) \in L[2i + 1]$. Similarly, we can prove the case that $(q, p) \in L[2i + 1]$, then $(p, q) \in L[2i]$. The proof is by induction on i.

For the base step, suppose that $i = 0$. According to the gap-greedy algorithm, it is not hard to see that $L[0] \in E$ and $L[1] \in E$.

For the induction hypothesis, suppose that the lemma holds for any $k < i$.

For the induction step, let $i > 0$ be an integer, $L[2i] := (p, q), L[2i + 1] := (q, p)$. We suppose that $L[2i] = (p, q) \in E$. By induction hypothesis, for each $l < i$ if $L[2l] \in E$, then $L[2l + 1] \in E$.

Let $E_i = \{L[2l], L[2l + 1] | l < i \text{ and } L[2l] \in E\}$. To show that $(q, p) \in E$, it is sufficient to show that for each $(r, s) \in E_i$,

$$\begin{aligned} &1.\ \text{angle}(qp, rs) > \theta \text{ or} \\ &2.\ \text{angle}(qp, rs) \le \theta \text{ and} |qr| > w \times |rs| \text{ and} \\ &|ps| > w \times |rs|. \end{aligned} \tag{1}$$

Note that $\text{angle}(pq, qp) = \pi > \theta$. Now, let (r, s) be an arbitrary element of E_i. By the definition of E_i, since $(r, s) \in E_i$,$(s, r) \in E_i$. Now, since $(p, q) \in E$ and by the definition of E_i, for the edge (s, r) we have

$$\begin{aligned} &1.\ \text{angle}(pq, sr) > \theta \textit{ or} \\ &2.\ \text{angle}(pq, sr) \le \theta \text{ and} |ps| > w \times |rs| \textit{ and} \\ &|qr| > w \times |rs|. \end{aligned} \tag{2}$$

It is easy to see that $\text{angle}(pq, sr) = \text{angle}(qp, rs)$. Hence, by Equation (2), we can easily conclude Equation (1).

For two real numbers θ and w such that $0 < \theta < \pi/4$ and $0 \le w < (cos\theta - sin\theta)/2$, we define

$$s := \max\left(\frac{4}{\sin(\theta/2)}, \frac{2}{w}\right) \tag{3}$$

Clearly, $4/s \le \sin(\theta/2)$ and $2/s \le w$.

Lemma 2[6, Lemma 9.1.2 and Lemma 10.1.6]. Let $s > 0$ be a real number, A and B be two finite sets of points that are

s-well-separated, and p and p' be any two points in A, and q and q' be any two points in B. Then

1. $|pp'| \le (2/s)|pq|$.
2. $|p'q'| \le (1 + 4/s)|pq|$.
3. $\sin(\text{angle}(pq, p'q')) \le 4/s$.

Now, let θ and w be two arbitrary real numbers such that $0 < \theta < \pi/4$ and $0 \le w < (cos\theta - sin\theta)/2$, and let $s = \max\left(\frac{4}{\sin(\theta/2)}, \frac{2}{w}\right)$. Let $S \subseteq \mathbb{R}^2$ be a set of n points and let $\{(A_i, B_i)\}_{i=1}^{m}$ be a WSPD for S with respect to the separation ratio s. Furthermore, for a set C of points, let R(C) be the bounding box of C. The length of a well-separated pair (A_i, B_i), denoted by $\ell(A_i, B_i)$, defined to be the distance between the centers of the circles through $R(A_i)$ and $R(B_i)$. Now, we have the following lemma.

Lemma 3 ([12]). Let γ and ℓ be positive real numbers, and let (A_i, B_i) be a well-separated pair in the WSPD with respect to s, with $\ell(A_i, B_i) = \ell$. The number of well-separated pairs (A'_i, B'_i) such that the length of the pair is in the interval $[\ell/2, 2\ell]$ and at least one of $R(A'_i)$ and $R(B'_i)$ is within a distance $\gamma\ell$ of either $R(A_i)$ or $R(B_i)$ is less than or equal to $c_{s\gamma} = O(s^d(1 + \gamma s)^d)$.

Let $G = (S, E)$ be the gap-greedy spanner generated by Algorithm 1.1 with respect to the parameters θ and w. Now, we have the following lemma.

Lemma 4. For every i with $1 \le i \le m$, the gap-greedy spanner contains either exactly two edges (p, q) and (q, p) with $(p, q) \in (A_i \times B_i) \cup (B_i \times A_i)$ or no edges in $(A_i \times B_i) \cup (B_i \times A_i)$.

Proof. First, we claim that $|E \cap (A_i \times B_i)| \le 1$ and $|E \cap (B_i \times A_i)| \le 1$. We prove the case $|E \cap (A_i \times B_i)| \le 1$. Similarly, we can prove the case $|E \cap (B_i \times A_i)| \le 1$. Suppose, for contradiction, $|E \cap (A_i \times B_i)| \le 2$. Hence, the gap-greedy spanner includes two directed edges $e = (x, y)$ and $e' = (x', y')$ such that $x, x' \in A_i$ and $y, y' \in B_i$. Suppose without loss of generality that $|xy| \le |x'y'|$ and therefore e is considered by Algorithm 1.1 before e'. By Lemma 2, we have $\sin(\text{angle}(x'y', xy)) \le 4/s$. Also, since $s = \max\left(\frac{4}{\sin(\theta/2)}, \frac{2}{w}\right)$, we have $4/s \le \sin(\theta/2)$. Hence, $\sin(\text{angle}(x'y', xy)) \le \sin(\theta/2)$ and therefore, since $0 < \theta < \pi/4$, we have $\text{angle}(x'y', xy) \le \theta/2 \le \theta$.

On the other hand, by Lemma 2 we have $|xx'| \le (2/s)|xy|$. Also, since $s = \max\left(\frac{4}{\sin(\theta/2)}, \frac{2}{w}\right)$, we have $2/s \le w$. Combining these two inequalities gives $|xx'| \le w|xy|$. Hence, according to Algorithm 1.1, since the algorithm checks e' after e, it does not add e' to E, which is a contradiction. Hence, the claim holds. So, we conclude that $|E \cap ((A_i \times B_i) \cup (B_i \times A_i))| \le 2$. By Lemma 2, it is not hard to see that $|E \cap ((A_i \times B_i) \cup (B_i \times A_i))| \ne 1$. Thus, this proves the lemma.

A straightforward corollary of Lemma 4 is the following result.

Corollary 1. The gap-greedy spanner contains at most $O(s^d n)$ edges.

Let E be some edge set with endpoints in the point set S. For every i with $1 \le i \le m$, let C_i be the set of all pairs $(u, v) \in (A_i \times B_i) \cup (B_i \times A_i)$ such that for each $(x, y) \in E$,

1. $\text{angle}(uv, xy) > \theta$ or,
2. $\text{angle}(uv, xy) \le \theta$ and $|ux| > w \times min(|uv|, |xy|)$ and $|vy| > w \times min(|uv|, |xy|)$.

Now, let $(u_i, v_i) \in C_i$ be a pair with the minimum Euclidean distance. We call such a pair by the selected pair of (A_i, B_i) with respect to E. Note that if C_i is empty, then we denote by nil, the selected pair of (A_i, B_i).

Lemma 5. For each i with $1 \le i \le m$, the selected pair of (A_i, B_i) can be computed in $O(|A_i||B_i||E|)$ time using $O(|E|)$ space.

Proof. Let SELECTEDPAIR(i) be a procedure that gets an integer $1 \le i \le m$ as input, then by checking all pairs $(p, q) \in A_i \times B_i$ returns the selected pair (A_i, B_i) with respect to E. It is easy to see that the time complexity of SELECTEDPAIR algorithm is $O(|A_i||B_i||E|)$ and its space complexity is $O(|E|)$.

Now, we are ready to describe the linear space gap-greedy algorithm in details. Our algorithm is similar to the algorithm of constructing the greedy spanner, another geometric spanner in the greedy-class, in linear space (see [12]). Our algorithm has three procedures: SELECTEDPAIR, ADDTOQUEUE (see Algorithm 2.1) and LINEARSPACEGAP (see Algorithm 2.2). Algorithm 2.2 is the main procedure. In the following when we say "the algorithm" it means Algorithm 2.2. The algorithm starts with an empty edge set E and an empty queue Q. At first, the algorithm computes a WSPD $\{(A_i, B_i)\}_{i=1}^{m}$ for point set S with respect to $s = \max\left(\frac{4}{\sin(\theta/2)}, \frac{2}{w}\right)$ and sorts the pairs (A_i, B_i) increasingly according to $\min(A_i, B_i)$ (for two nonempty sets A_i and B_i, $\min(A_i, B_i)$ is defined the shortest distance between the two circles through the bounding boxes $R(A_i)$ and $R(B_i)$. Similarly $\max(A_i, B_i)$ is defined). Then the algorithm, at start, calls ADDTOQUEUE(Q, i) with i = 1. By this calling, the procedure ADDTOQUEUE adds the selected pair of the well-separated pairs into the queue Q in order of $\min(A_i, B_i)$. Note that adding to Q in each procedure call continues as long as $\min(A_i, B_i) \le \min(Q)$ or Q is empty.

Algorithm 2.1: ADDTOQUEUE(Q, i)

```
Input: Queue Q and integer i with 1 ⩽ i ⩽ m.
while i ⩽ m, and either Q is empty or min(A_i, B_i) ⩽ min(Q) do
    x := SELECTEDPAIR(i);
    if x is not nil, but a directed pair (u, v) then
        Add (u, v) to Q with key |uv|, and associate this entry with (A_i, B_i);
    end
    i = i + 1;
end
```

Algorithm 2.2: LINEARSPACEGAP(S, θ, w)

```
   /* This algorithm takes as input a set S of n points in the plane, and two real
      numbers θ and w such that 0 < θ < π/4, 0 ⩽ w < (cos θ − sin θ)/2. The algorithm
      returns a directed t-spanner G = (S, E), for t = 1/(cos θ − sin θ − 2w).   */
 1 s := max((4/(sin(θ/2)), 2/w));
 2 {(A_i, B_i)}_{i=1}^m :=WSPD of S with respect to s;
 3 Sort increasingly the pairs {(A_i, B_i)}_{i=1}^m according to min(A_i, B_i);
 4 E := ∅;
 5 Q := ∅;/* Q is a priority queue.                                              */
 6 i := 1;
 7 ADDTOQUEUE(Q, i);
 8 while Q is not empty do
 9     Extract the minimum from Q and denote it by (u, v). Also, suppose that (u, v) ∈ (A_k, B_k);
10     E := E ∪ {(u, v), (v, u)};
11     foreach pair (A_j, B_j) with an entry in Q for which either bounding box is at most
       w × max(A_j, B_j) away from either u or v do
12         x := SELECTEDPAIR(j);
13         if x is a pair (u', v') then
14             Update the entry associated with {A_j, B_j} in Q to include (u', v') and change its key to
               |u'v'|;
15         else
               /* x is nil                                                      */
16             Remove from Q the entry associated with (A_j, B_j);
17         end
18     end
19     ADDTOQUEUE(Q, i);
20 end
21 return the graph G = (S, E);
```

After returning from ADDTOQUEUE to the main procedure, the algorithm extracts the minimum value for Q (Line 9 of Algorithm 2.2). Let (u, v) be the pair corresponding to theminimum value of Q that was extracted from Q. It is easy to see that (u, v) is an edge of the gap-greedy spanner. Also, by Lemma 4, (v, u) is an other edge of the gap-greedy spanner. So, the algorithm adds these two edges to E. Note that after extracting, the algorithm removes the entry corresponding to the minimum value from Q.

Now, the algorithm must update Q. The updating should be done carefully (lines 11-14). In general, when the edges (u, v) and (v, u) is added to E, the algorithm does not update all entries in Q but it only updates some special entries in Q that the selected pair in the corresponding well-separated pair may be changed by adding (u, v) and (v, u) to E. Note that in Line 1 of Algorithm 2.1, it is possible that when while loopterminates, the value of i is less than m. Hence, the well-separated pairs with index greater than or equal to i are not considered by ADDTOQUEUE. So, in Algorithm 2.2, after adding the two edges to E and updating Q in lines 11-14, we still need to call the procedure ADDTOQUEUE with input parameters i and Q (we assumed that i is a global variable).

Theorem 1. The output of two algorithms LINEARSPACEGAP and GAPGREEDY are same on the same input.

Proof. We prove that if the algorithm adds (u, v) and (v, u) to E on line 10, then (u, v) is the closest pair of points that for each edge (x, y) in the edge set E computed so far,

1. angle(uv, xy) $> \theta$ or,
2. angle(uv, xy) $\leq \theta$ and $|ux| > w \times min(|uv|, |xy|)$ and $|vy| > w \times min(|uv|, |xy|)$.

The edge set of the gap-greedy spanner consists of exactly these edges. Hence, this proves the theorem. Clearly, if we call SELECTEDPAIR(i) on every well-separated pair and compute the closest pair, denoted by (u, v), of the non-nilresults, then that would be the closest pair of points that for each edge (x, y) in the edge set E computed so far, conditions 1 and 2 hold. Actually, algorithm LINEARSPACEGAP does this, except it does not recalculate SELECTEDPAIR(i) for every pair, but only for specific pairs.

Lemma 6. For any well-separated pair $(A_r, B_r)(1 \leq r \leq m)$, the number of times SELECTEDPAIR(r) is called is at most $1 + c_{sw'}$ where $w' = \frac{3}{2}w$ and $c_{sw'} = O(s^d(1 + w's)^d)$.

Proof. Proof is similar to the proof of Lemma 9 in [12]. For every integer r with $1 \leq r \leq m$, SELECTEDPAIR(r) is called once in the procedure ADDTOQUEUE. Moreover, SELECTEDPAIR(r) may be still called in Line 12 of Algorithm 2.2. Suppose that the well-separated pair (A_k, B_k) causes the SELECTEDPAIR(r) to be called. Similar to the proof of Lemma 9 in [12], we can prove that $\ell(A_k, B_k) \in [\ell(A_r, B_r)/2, 2\ell(A_r, B_r)]$. On the other hand, by condition of Line 11 of Algorithm 2.2 and since $\max(A_r, B_r) \leq \frac{3}{2}\ell(A_r, B_r)$ (see [12]), at least one of $R(A_k)$ and $R(B_k)$ is within distance $w'\ell(A_r, B_r)$ of either $R(A_r)$ or $R(B_r)$. Hence, by Lemma 3, we conclude that the number of the well-separated pairs (A_k, B_k) cause the SELECTEDPAIR(r) to be called is at most $c_{sw'}$. So, the lemma follows.

Suppose that in Line 2 of Algorithm 2.2, $\{(A_i, B_i)\}_{i=1}^m$ be a WSPD of size $m = O(s^d n)$ that can be computed in$O(n\log n + s^d n)$ time and can be represented in $O(s^d n)$ space. Now, we have the following theorem.

Theorem 2. Let θ and w be two real numbers such that $0 < \theta < \pi/4$ and $0 \le w < (cos\theta - sin\theta)/2$, and lets:= $\max\left(\frac{4}{\sin(\theta/2)}, \frac{2}{w}\right)$. Algorithm LINEARSPACEGAP computes the gap-greedy spanner in $O(s^{2d}(1+\frac{3}{2}ws)^d n^3)$ time using $O(s^d n)$ space.

Proof. Without affecting on the overall running time, we can easily preprocess all well-separated pairs in $O(m^2)$ time, to answer the following query: given a pair $(u,v) \in (A_k, B_k)$, for which well-separated pairs (A_j, B_j) either bounding box is at most $w \times \max(A_j, B_j)$ away from either u or v (see Line 11 of Algorithm 2.2). On the other hand, for each well-separated pair there is at most $c_{sw'}$ such well-separated pairs (see Lemma 3). Hence, the preprocessing uses linear space. Using Corollary 1, lemmas 5 and 6, all SELECTEDPAIR(i) calls take

$$O\left(\sum_{i=1}^{m}(1+c_{sw'})s^d|A_i||B_i|n\right),$$

time and the required space is $O(s^d n)$ (note that we can reuse the space for calls). Since $w' = \frac{3}{2}w$ and $c_{sw'} = O(s^d(1+w's)^d)$, we have

$$\begin{aligned} &O\left(\sum_{i=1}^{m}(1+c_{sw'})s^d|A_i||B_i|n\right) \\ &= O\left((1+c_{sw'})s^d n\sum_{i=1}^{m}|A_i||B_i|\right) \\ &= O\left((1+c_{sw'})s^d n\binom{n}{2}\right) \\ &= O(s^{2d}(1+\frac{3}{2}ws)^d n^3). \end{aligned}$$

Note that by the definition of WSPD we have $\sum_{i=1}^{m}|A_i||B_i| = \binom{n}{2}$. Also, we can ignore the time of all other steps compared to the time of SELECTEDPAIR (i) calls and these steps can be done using a linear space. This completes the proof.

2.2. Improving the Time Complexity

In this section, we show how to construct the gap-greedy spanner in quadratic time using the quadratic space. In the previous section, in procedures LINEARSPACEGAP and ADDTOQUEUE, we used the procedure SELECTEDPAIR, and we saw that the overall running time of the procedure LINEARSPACEGAP is much dependent on the running time of the procedure SELECTEDPAIR. To improve the overall running time of the procedure LINEARSPACEGAP, we make some modifications on SELECTEDPAIR. The idea is as follows. We saw that in LINEARSPACEGAP algorithm, after adding the two edges to the edge set, the algorithm updates the selected pair of the well-separated pairs by comparing all pairs of points in the well-separated pair with all edges computed so far, while here we associate a list to each well-separated pair including all possible edges of the gap-greedy spanner and update the selected pair of the well-separated pairs by comparing all elements of the corresponding list with the two edges that already added to the edge set. In particular, for each $1 \le i \le m$, we associate to the entry of queue Q corresponding to the pair (A_i, B_i), a list N_i including all pairs $(x,y) \in (A_i \times B_i) \cup (B_i \times A_i)$ which are the candidates for being the edges of the gap-greedy spanner. Then, after adding the edges (u,v) and (v,u) (see Line 10 of Algorithm 2.2), we update the entries of Q including list N_j and the key corresponding to pairs (A_j, B_j) which satisfy the requirements in Line 11 of the algorithm LINEARSPACEGAP. To this end, we replace the procedure SELECTEDPAIR by the following procedure, denoted by IMPROVEDSELECTEDPAIR (see Algorithm 2.3). Note that for each $1 \le i \le m$, we set $N_i := (A_i \times B_i) \cup (B_i \times A_i)$ and store them in Q at the start of the procedure LINEARSPACEGAP.

Algorithm 2.3: IMPROVEDSELECTEDPAIR(i,u,v)

```
1  minspair:=nil and let |minspair| := ∞;
2  F := {(u,v),(v,u)};
3  foreach pair (p,q) ∈ N_i do
4      flag:=CHECKPAIR(p,q,F);
5      if flag = true then
6          if |pq| < |minspair| then
7              minspair:=(p,q);
8          end
9      end
10     else
11         N_i := N_i − {(p,q)};
12     end
13 end
14 Update the entry associated with {A_i,B_i} in Q to include N_i;
15 return minspair;
```

Notably, in the updating step, we do not need to compare the elements of N_i with all edges in the edge set E computed so far because the checking is done in previous steps. It is easy to see that for each i with $1 \le i \le m$, $|N_i| \le |(A_i \times B_i) \cup (B_i \times A_i)$. Hence, the procedure IMPROVEDSELECTEDPAIR(i,u,v) takes $O(|A_i||B_i|)$ time. Also, in similar to the procedure SELECTEDPAIR(i), the procedure IMPROVEDSELECTEDPAIR(i,u,v) is called at most $1 + c_{sw'}$, then the time complexity of Algorithm 2.2 after the above modifications is

$$\begin{aligned} O\left(\sum_{i=1}^{m}(1+c_{sw'})|A_i||B_i|\right) &= O\left((1+c_{sw'})\binom{n}{2}\right) \\ &= O(s^d(1+\frac{3}{2}ws)^d n^2). \end{aligned}$$

Since for storing each list N_i we need $O(|A_i||B_i|)$ space, it is obvious that the overall space which is used for constructing the gap-greedy spanner by the above modifications is $O(n^2)$. By above arguments, we conclude the following result.

Theorem 3. The gap-greedy spanner can be computed in quadratic time using the quadratic space.

3. Improving the Dilation

Arya and Smid [8], proved that the gap-greedy spanner has the dilation $1/(\cos(\theta) - \sin(\theta) - 2w)$. Here, we improve the dilation of the gap-greedy spanner to $1/(1 - 2\sin(\theta/2) - 2w)$. To the best of our knowledge, this is the first improvement on the dilation of the gap-greedy spanner. We need the following lemmas.

Lemma 7 ([16]). Let a, b and c be three points such that $|ac| \le |ab|$ and $\angle bac \le \alpha < \pi$. Then

$$|bc| \le |ab| - \left(1 - 2\sin\left(\frac{\alpha}{2}\right)\right)|ac|.$$

Now, similar to the proof of lemmas 6.4.1 and 7.1.1 in [6], we prove the following lemmas.

Lemma 8. Let t, θ, and w be real numbers, such that $0 < \theta < \pi/3$, $0 \le w < (1 - 2sin(\theta/2))/2$, and $t \le 1/(1 - 2\sin(\theta/2) - 2w)$. Let p, q, r, and s be points in $\mathbb{R}^d$, such that

1. $p \ne q, r \ne s$,
2. $\text{angle}(pq, rs) \le \theta$,
3. $|rs| \le |pq|$, and
4. $|pr| \le w|rs|$.

Then $|pr| < |pq|, |sq| < |pq|$ and $t|pr| + |rs| + t|sq| \le t|pq|$.

Proof. First, we prove $|pr| < |pq|$. Since $|pq| < \sqrt{2}|pq|$ and by condition 3, we have $|rs| < \sqrt{2}|pq|$. Also, since $|pr| \le w|rs|$(condition 4) and $w < 1/2$, we have $|pr| < |rs|/2$. Now, using the equations $|rs| < \sqrt{2}|pq|$ and $|pr| < |rs|/2$, we conclude that $|pr| < \sqrt{2}|pq|/2 < |pq|$.

Now, we prove that $|sq| < |pq|$. Let (r, v) be a vector that has the same direction as the vector (p, q) with $|rv| = |pq|$. Let u be the orthogonal projection of s on the vector (r, v) (see figure 1). Note that since $|rs| \le |pq|$, there always exists the point u.

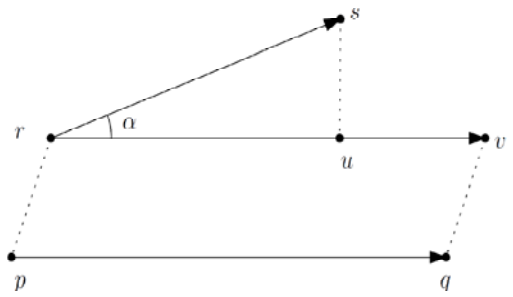

Figure 1. Illustrating of the proof of Lemma 8

It is clear that $|pr| = |vq|$. Let $\alpha = \text{angle}(pq, rs) \le \theta$. Now, by the triangle inequality and using Lemma 7, we have

$|sq| \le |sv| + |vq|$
$\le |rv| - (1 - 2\sin(\alpha/2))|rs| + |vq|$ (by Lemma 7)
$= |pq| - (1 - 2\sin(\alpha/2))|rs| + |pr|$
$\le |pq| - (1 - 2\sin(\alpha/2))|rs| + w|rs|$(by condition 4)
$\le |pq| - (1 - 2\sin(\theta/2))|rs| + w|rs|$(since $\alpha \le \theta$)
$= |pq| - (1 - 2\sin(\theta/2) - w)|rs|$
$< |pq|(since\ w < (1 - 2sin(\theta/2))/2)$.

Now, we prove $t|pr| + |rs| + t|sq| \le t|pq|$. By the above formulas, we have

$t|pr| + |rs| + t|sq|$
$\le t|pr| + |rs| + t(|pq| - (1 - 2\sin(\theta/2) - w)|rs|)$
$\le tw|rs| + |rs| + t(|pq| - (1 - 2\sin(\theta/2) - w)|rs|)$
$= t|pq| + (1 - t(1 - 2\sin(\theta/2) - 2w))|rs|$
$\le t|pq|$(since $t \le 1/(1 - 2\sin(\theta/2) - 2w)$).

This completes the proof.

Lemma 9. Let θ, w, and t be real numbers such that $0 < \theta < \pi/3$, $0 \le w < (1 - 2sin(\theta/2))/2$, and $t1/(1 - 2\sin(\theta/2) - 2w)$. Let S be a set of n points in the plane, and let $G = (S, E)$ be a directed graph, such that the following holds: For any two distinct points p and q of S, there is an edge $(r, s) \in E$, such that

1. $\text{angle}(pq, rs) \le \theta$,
2. $|rs| \le |pq|$, and
3. $|pr| \le w|rs|$ or $|qs| \le w|rs|$.

Then, the graph G is a t-spanner for S.

Proof. Proof is similar to the proof of Lemma 7.1.1 in [6] by applying Lemma 8.

The following theorem shows that the dilation of the gap-greedy spanner is at most $1/(1 - 2\sin(\theta/2) - 2w)$.

Theorem 4. Let θ, and w be real numbers, such that $0 < \theta < \pi/3$, $0 \le w < (1 - 2sin(\theta/2))/2$, and let S be a set of n points in the plane. The graph $G = (S, E)$ that is returned by the algorithm $\text{GAPGREEDY}(S, \theta, w)$ is a t-spanner for S, for $t = 1/(1 - 2\sin(\theta/2) - 2w)$.

Proof. Proof is similar to the proof of Lemma 7.2.1 in [6] by applying Lemma 9.

4. Some Other Cubic Time Algorithms

In Section 1, we presented the gap-greedy algorithm (Algorithm 1.1). In this section, we propose three other algorithms to construct the gap-greedy spanner. Although these algorithms have running time $O(n^3)$ and space $O(n^2)$, they have different behavior in practice. Note that all algorithms mention here generate exactly the same graph, so we do not compare the graphs generated by different algorithmin the following, we describe these three algorithms. In Section 5, we present the experimental results on their running time and some properties of the generated graph. We denote these algorithms by randomized gap-greedy algorithm, binary gap-greedy algorithm and forbidden gap-greedy algorithm.

4.1. Randomized Gap-Greedy Algorithm

Consider the lines 5-8 of Algorithm 1.1. In these lines, the algorithm compares the pair (p, q) with at most all edges in E already computed. Hence, the order of the pairs (r, s) which are compared with (p, q) is not important. So, we can randomly compare the pairs (r, s) with (p, q). Hence, in the rndomized gap-greedy algorithm, we first generate a random permutation on the members of E and then compare the pairs (r, s) with (p, q). Before implementing this algorithm, we conjectured that this improves the time, but after the implementing, we saw that the algorithm is somehow slower than Algorithm 1.1 (see Section 5).

4.2. Forbidden Gap-Greedy Algorithm

Let S be a set of points in the plane and $w > 0$ be a real number. For each pair (r, s), let D_{rs} be the set of all pointsp $\in S$ such that $|pr| \le w|rs|$. According to Algorithm 1.1, obviously if for a pair (p, q) we have $|pr| \le w|rs|$ or$|qs| \le w|rs|$ and $\text{angle}(pq, rs) \le \theta$, then the pair (p, q) is not added to E. In forbidden gap-greedy algorithm, we always add the shortest edge $(r, s) \in L$ (L is the sorted list of all ordered pairs of points) to E.

After adding (r, s) to E, we find the sets D_{rs} and D_{sr}. Then, for each point $p \in D_{rs}$ ($q \in D_{sr}$), if there is a point $q \in S$ ($p \in S$) such that $\text{angle}(pq, rs) \le \theta$, then we delete the pair (p, q) from L. For more details on the algorithm, see Algorithm 4.1. It is notable the pair (p, q) may be detected at least twice during the execution of the algorithm. We hoped that this will make the algorithm faster, at least in practice, but the experiments show that this is not correct (see Section 5).

Algorithm 4.1: ForbidGapGreedy(S, θ, w)

```
/* This algorithm takes as input a set S of n points in the plane, and two real
   numbers θ and w such that 0 < θ < π/4, 0 ⩽ w < (cos θ − sin θ)/2. The algorithm
   returns a directed t-spanner G = (S, E), for t = 1/(cos θ − sin θ − 2w).   */
1  Sort the 2(n choose 2) ordered pairs of distinct points in non-decreasing order of their distances (ties are
   broken arbitrarily), and store them in a list L;
2  E := ∅;
3  while L ≠ ∅ do
4  |  Choose the shortest edge (r, s) ∈ L;
5  |  E := E ∪ {(r, s)};
6  |  Find all points p ∈ S such that |pr| ⩽ w|rs|, and store them in D_rs;
7  |  Find all points q ∈ S such that |qs| ⩽ w|rs|, and store them in D_sr;
8  |  foreach point p ∈ D_rs do
9  |  |  foreach point q ∈ S do
10 |  |  |  if angle(pq, rs) ⩽ θ then
11 |  |  |  |  L := L − {(p, q)};
12 |  |  |  end
13 |  |  end
14 |  end
15 |  foreach point q ∈ D_sr do
16 |  |  foreach point p ∈ S do
17 |  |  |  if angle(pq, rs) ⩽ θ then
18 |  |  |  |  L := L − {(p, q)};
19 |  |  |  end
20 |  |  end
21 |  end
22 end
23 return G = (S, E);
```

Algorithm 4.2: BinaryGapGreedy(S, θ, w)

```
/* This algorithm takes as input a set S of n points in the plane, and two real
   numbers θ and w such that 0 < θ < π/4, 0 ⩽ w < (cos θ − sin θ)/2. The algorithm
   returns a directed t-spanner G = (S, E), for t = 1/(cos θ − sin θ − 2w).   */
1  Sort the 2(n choose 2) ordered pairs of distinct points in non-decreasing order of their distances (ties are
   broken arbitrarily), and store them in a list L;
2  E := ∅;
3  foreach ordered pair (p, q) ∈ L do
4  |  add := true;
5  |  Sort the pairs (r, s) in E according to the angle between the vector s − r and the x-axis;
6  |  Use the binary search algorithm to find the largest index i such that for each j < i we have
   |  angle(pq, r_j s_j) ⩽ θ, where E[j] := (r_j, s_j);
7  |  for (j = 0; j < i; j++) do
8  |  |  add := add ∧(|pr_j| > w|r_j s_j|) ∧ (|qs_j| > w|r_j s_j|);
9  |  end
10 |  if add = true then
11 |  |  E := E ∪ (p, q);
12 |  end
13 end
14 return G = (S, E);
```

4.3. Binary Gap-Greedy Algorithm

Algorithm 1.1, in Line 7, only considers all pairs (r, s) ∈ E such that angle(pq, rs) ≤ θ. To determine which pairs (r, s) satisfy the condition angle(pq, rs) ≤ θ, the algorithm checks all members in E. Obviously, checking all members in E is not a good method. Hence, we propose the binary gap-greedy algorithm. This algorithm is similar to Algorithm 1.1 just with some modifications as follows. The algorithm, during the execution, always keep the pairs (r, s) ∈ E increasingly sorted according to the angle between the vector $\overrightarrow{s-r}$ and the x-axis. Then, when the algorithm wants to process the pair (p, q), it uses the binary search algorithm to determine which pairs (r, s) ∈ E satisfy the condition angle(pq, rs) ≤ θ. For more details on the algorithm, see Algorithm 4.2. Although the running time of the binary gap-greedy algorithm is still $O(n^3)$, the experiments show a good improvement in time compared to Algorithm 1.1 In Section 5, we will see that the

binary gap-greedy algorithm is faster than Algorithm 1.1, randomized gap-greedy algorithm and forbidden gap-greedy algorithm.

5. Experimental Results

Here, at first, we present some tables and figures that show the behavior of the three algorithms presented in the previous section. Then, we present some tables and figures that illustrate the geometric properties of the gap-greedy spanner compared to the greedy spanner. The experiments were done on a machine for point sets up to 5,000 points in the Euclidean plane and the code was compiled using g++ version 4.9.2 and was ran on an Intel ® Core i3-4160 CPU @ 3.60GHz × 4 machine with 8 GB of RAM and Ubuntu 16.04 operating system. For larger point sets, we did not run the code because the algorithms take a long time.

The experiments are done on uniformly distributed point sets; for point sets up to 5,000 points. For each set of n points, the range of each coordinate of a point is between 0 and $\sqrt{n}$. For each case, we run the algorithms on 10 different point sets with different parameters and we take average between them and use maximum and minimum values between them to see how much the difference different point sets is. Note that in all figures we used the average values of the parameters.

Some results of the experiments are presented in tables 2 and 3 for different point sets and different parameters. The time in all tables and diagrams are in seconds. Note that for the sake of simplicity, we denote Algorithm 1.1, randomized gap-greedy, binary gap-greedy and Forbidden gap-greedy by Orig, Rand, Bina and Forb, respectively. Figure 2 illustrates the running time of the algorithms Orig, Rand, Bina and Forb for $n = 100, 500, 1000$ and different values θ and w. Since the running time of algorithm Rand and Forb is very much in compared to Orig and Bina, for $n = 2000, 5000$, we only ran the code for Orig and Bina.

Figure 3 shows the running time of the algorithms Orig and Bina for $n = 100, 500, 1000, 2000$ and 5000 points for different parameters. Obviously, the experiments show that the algorithm Bina is faster than the other algorithms.

5.1. Gap-Greedy Spanner VS Greedy Spanner

For a geometric graph G with vertex set S, size and degree of G is called the number of edges in G and the maximum degree of vertices in G, respectively. Also, we denote by weight^* the ratio between weight of G and $\text{wt}(\text{MST}(S))$ ($\text{weight}^* = \text{weight}(G)/\text{wt}(\text{MST}(S))$). Figure 4 shows size, degree, weight^* and the running time of the gap-greedy spanner compared to the greedy spanner. Note that for computing the greedy spanner, we used the FG-greedy algorithm proposed by Bose et al. [11]. The experiments show the greedy spanner is better than the gap-greedy spanner in terms of size, degree, weight^* and the running time, in practice (see table 2).

Table 2. The running time of computing the gap-greedy spanner and greedy spanner for different point sets and different parameters. Note that for computing the gap-greedy spanner and greedy spanner we used the algorithms Bina and FG-greedy, respectively, also T-Gap and T-Greedy mean the running time of the gap-greedy spanner and greedy spanner, respectively

n	t	θ	w	T-Gap			T-Greedy		
				Min	Avg	Max	Min	Avg	Max
2000	2	0.17453	0.1555	346.494	354.701	373.766	2.86321	2.97468	3.42027
2000	2	0.34907	0.0488	448.972	455.999	463.62	2.91376	2.94009	2.96666
2000	2	0.08727	0.2045	332.211	340.763	345.33	2.89091	2.92807	2.96824
2000	1.5	0.03491	0.1489	377.404	386.34	395.024	3.31111	3.46331	3.88642
2000	1.5	0.08727	0.1211	441.549	451.354	457.235	3.30634	3.44079	3.60082
2000	1.5	0.17453	0.072	420.453	425.17	434.652	3.32062	3.41541	3.51767
2000	1.2	0.03491	0.0655	528.159	537.698	541.108	4.56827	4.96712	6.21486
2000	1.2	0.06981	0.0472	557.811	567.492	579.142	4.54005	4.78704	4.92434
2000	1.2	0.13963	0.0088	526.323	530.403	535.756	4.56437	4.8384	5.18147
5000	2	0.17453	0.1555	8668.18	9008.08	9517.11	23.9422	24.8324	28.6767
5000	2	0.34907	0.0488	9965.3	10096.1	10351.3	24.0232	24.38	24.9487
5000	2	0.08727	0.2045	7436.58	7697.6	7813.71	23.7166	24.5205	27.7897
5000	1.5	0.03491	0.1489	7122.66	7267.73	7414.24	27.5712	28.4923	31.382
5000	1.5	0.08727	0.1211	9277.28	9544.13	9679.36	27.378	28.0962	30.0544
5000	1.5	0.17453	0.072	10703.9	10858.9	11007.5	27.3257	27.9032	28.9135
5000	1.2	0.03491	0.0655	9843.21	10101.8	10341.2	40.4196	41.3087	44.3336
5000	1.2	0.06981	0.0472	11567.7	11620.4	11664.3	39.7372	42.1711	47.2002

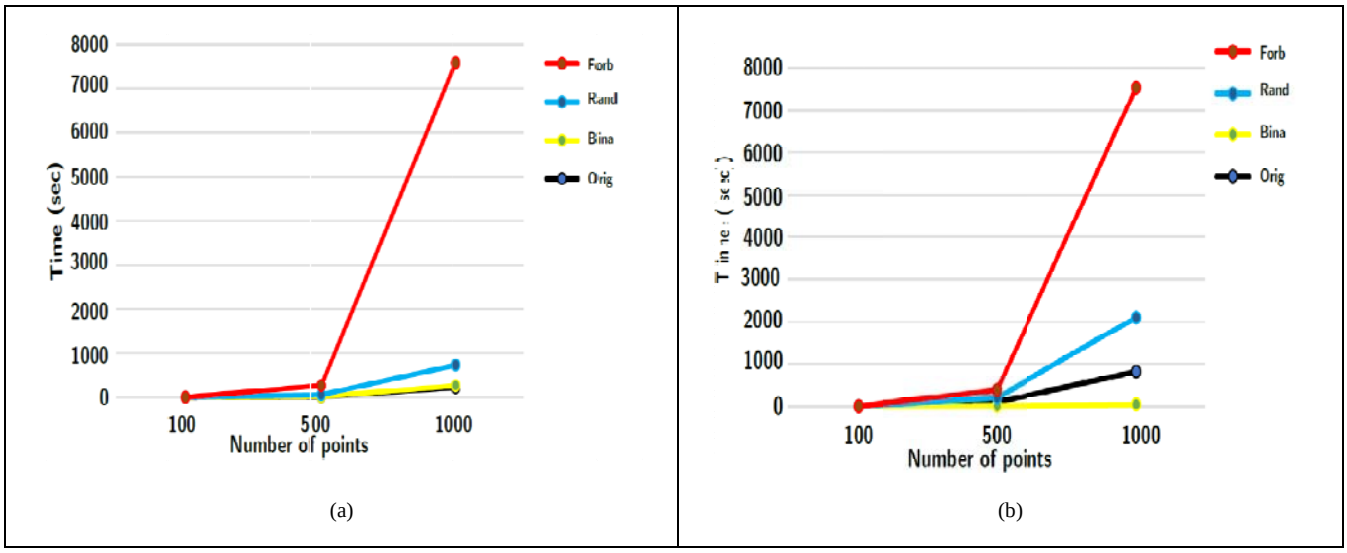

Figure 2. The running time of Orig, Bina, Rand and Forb for different values θ and w. (a): $\theta = 0.174533, w = 0.1555$. (b): $\theta = 0.139626, w = 0.0088$

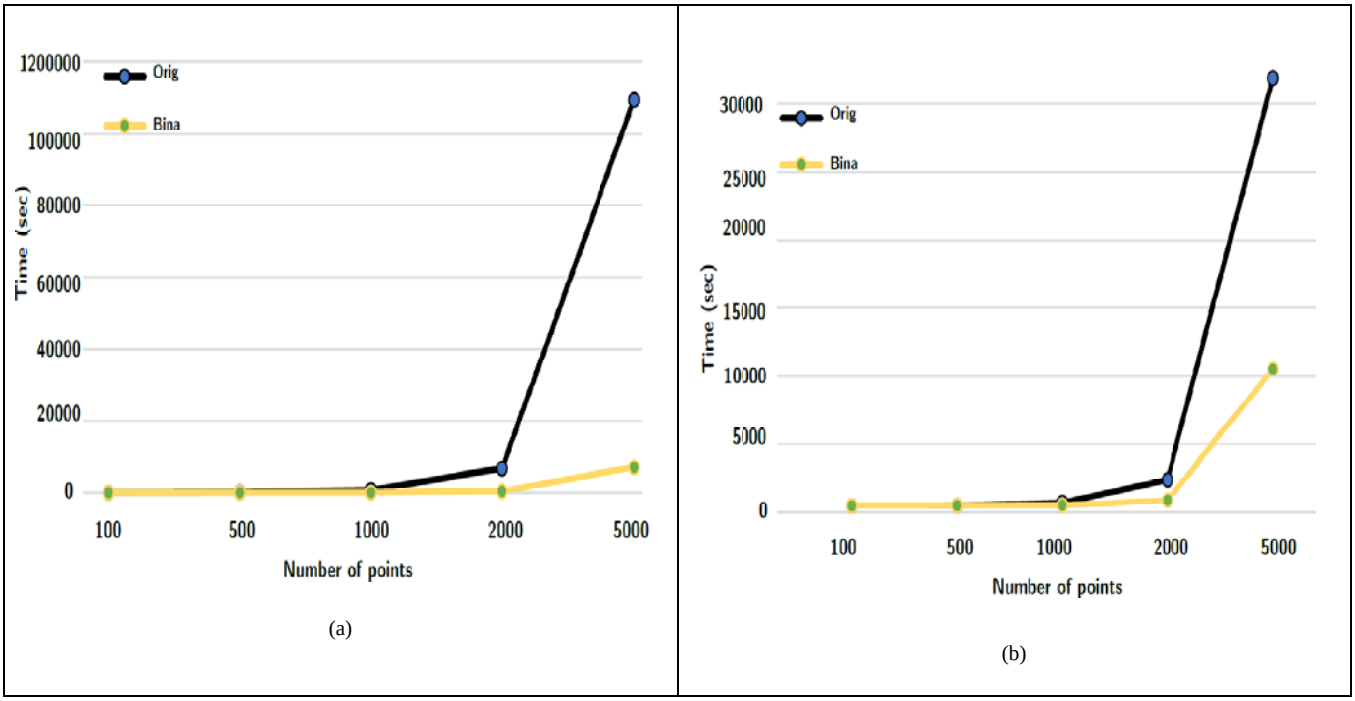

Figure 3. The running time of Orig and Bina for different values θ and w, and point sets with $100, 500, 1000, 2000$ and 5000 points. (a): $\theta = 0.034907, w = 0.1489$. (b): $\theta = 0.174533, w = 0.1555$

Table 3. The running time (in seconds) of Orig and Bina for different values of θ and w and point sets with 100, 500, 1000, 2000 and 5000 points

n	t	θ	w			Orig		Bina	
				Min	Avg	Max	Min	Avg	Max
100	2	0.174533	0.1555	0.189052	0.192168	0.200452	0.032138	0.033672	0.04154
100	2	0.349066	0.0488	0.104685	0.108556	0.111501	0.039204	0.039977	0.040599
100	2	0.087267	0.2045	0.292295	0.304976	0.325859	0.025279	0.027097	0.031798
100	1.5	0.034907	0.1489	0.5417	0.561884	0.579034	0.023938	0.025107	0.026552
100	1.5	0.087267	0.1211	0.314975	0.330566	0.342109	0.028222	0.029445	0.03107
100	1.5	0.174533	0.072	0.191486	0.197595	0.202048	0.032856	0.034317	0.035812
100	1.2	0.034907	0.0655	0.614096	0.6377	0.671346	0.028203	0.029879	0.031991
100	1.2	0.069813	0.0472	0.395185	0.412943	0.432324	0.028422	0.031437	0.034663
100	1.2	0.139626	0.0088	0.237887	0.241003	0.246899	0.032606	0.033359	0.03497
500	2	0.174533	0.1555	26.5781	16.028	26.9208	5.08668	3.09473	5.19633
500	2	0.349066	0.0488	14.4613	14.6394	14.8838	6.04733	6.08554	6.14374
500	2	0.087267	0.2045	44.2941	44.8861	45.5254	3.95437	4.06016	4.20062
500	1.5	0.034907	0.1489	96.0017	97.9644	100.697	3.41377	3.75095	3.88424
500	1.5	0.087267	0.1211	49.7005	50.3303	51.0666	4.60462	4.73807	4.88417
500	1.5	0.174533	0.072	28.0196	28.1248	28.2095	5.65276	5.78268	5.88584
500	1.2	0.034907	0.0655	114.142	115.154	115.854	4.48115	4.87984	5.06587
500	1.2	0.069813	0.0472	64.5877	65.3423	66.2237	4.90651	5.15237	5.26042
500	1.2	0.139626	0.0088	35.2026	35.4035	35.7052	5.85189	5.87883	5.91205
1000	2	0.174533	0.1555	217.829	218.409	219.309	41.9764	43.6197	45.036
1000	2	0.349066	0.0488	117.096	46.8453	117.13	53.9436	21.5988	54.0502
1000	2	0.087267	0.2045	366.308	373.244	381.436	44.2505	46.0837	47.7551
1000	1.5	0.034907	0.1489	803.76	814.099	824.686	36.6453	37.7144	38.6573
1000	1.5	0.087267	0.1211	411.853	414.84	418.608	39.3009	41.9302	45.6503
1000	1.5	0.174533	0.072	225.383	227.225	228.236	48.8309	49.7053	50.0877
1000	1.2	0.034907	0.0655	979.307	985.817	993.367	55.54	56.3426	57.1856
1000	1.2	0.069813	0.0472	540.035	542.59	544.645	47.8021	51.9081	54.6434
1000	1.2	0.139626	0.0088	287.979	289.424	290.258	49.4006	51.5847	53.3267
2000	2	0.174533	0.1555	1739.45	1749.4	1759.65	346.494	354.701	373.766
2000	2	0.349066	0.0488	973.64	985.434	997.16	448.972	455.999	463.62
2000	2	0.087267	0.2045	2985.98	3017.06	3051.53	332.211	340.763	345.33
2000	1.5	0.034907	0.1489	6738.6	6801.87	6836.32	377.404	386.34	395.024
2000	1.5	0.087267	0.1211	3393.83	3412.06	3439.93	441.549	451.354	457.235
2000	1.5	0.174533	0.072	1844.89	1854.7	1865.62	420.453	425.17	434.652
2000	1.2	0.034907	0.0655	8233.95	8253.75	8299.48	528.159	537.698	541.108
2000	1.2	0.069813	0.0472	4495	4515.3	4544.87	557.811	567.492	579.142
2000	1.2	0.139626	0.0088	2373.43	2381.67	2393.82	526.323	530.403	535.756
5000	2	0.174533	0.1555	28032.6	28119.9	28328.1	8668.18	9008.08	9517.11
5000	2	0.349066	0.0488	15093.1	15146.2	15193.3	9965.3	10096.1	10351.3
5000	2	0.087267	0.2045	48209.5	48523.4	48857.6	7436.58	7697.6	7813.71
5000	1.5	0.034907	0.1489	108824	109393	109737	7122.66	7267.73	7414.24
5000	1.5	0.087267	0.1211	54727.5	54817.6	54919	9277.28	9544.13	9679.36
5000	1.5	0.174533	0.072	29566.2	29763.2	29931.9	10703.9	10858.9	11007.5
5000	1.2	0.034907	0.0655	133987	134226	134362	9843.21	10101.8	10341.2
5000	1.2	0.069813	0.0472	72709.9	72828.8	72962.4	11567.7	11620.4	11664.3

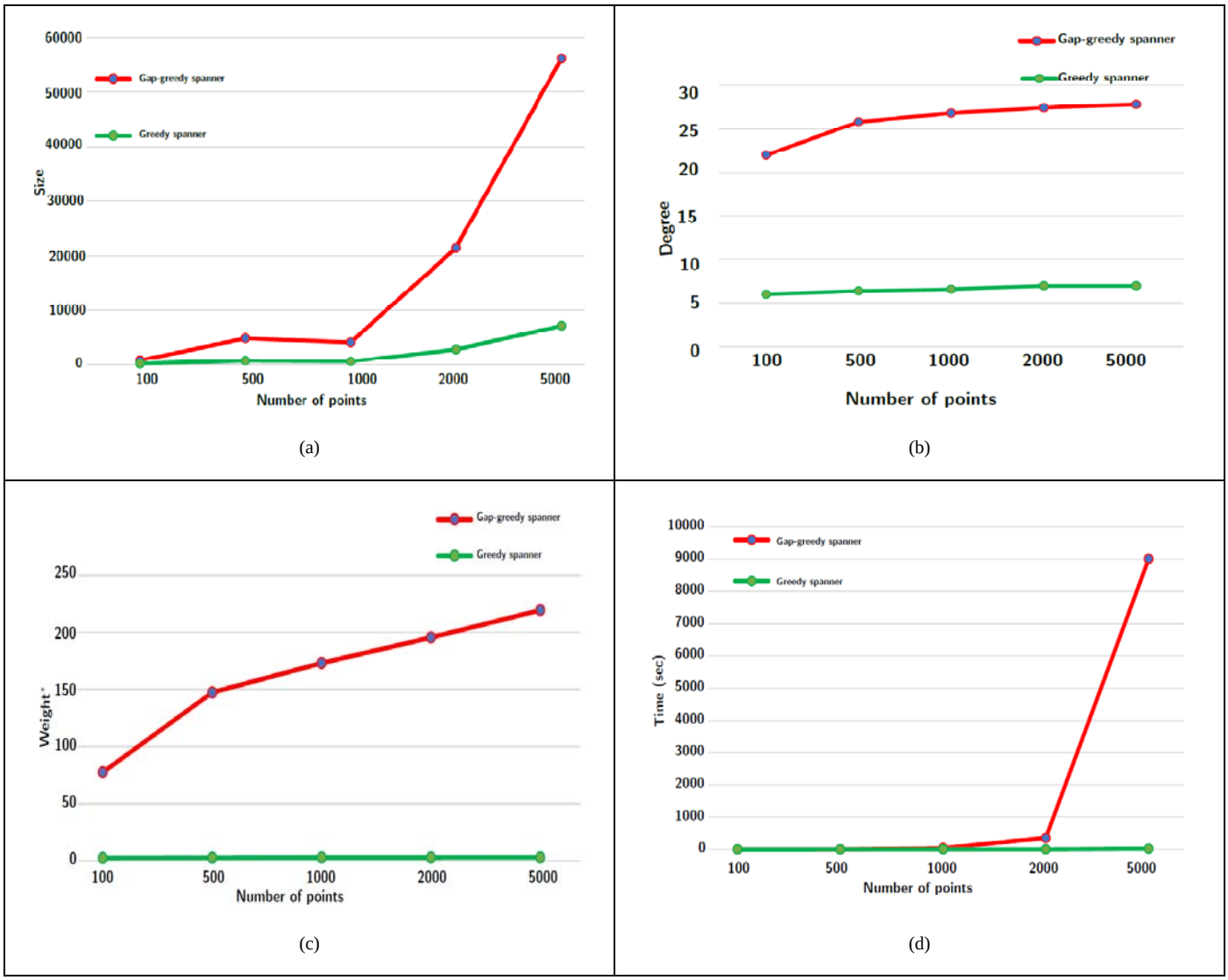

Figure 4. Size, degree, weight* and running time of the gap-greedy spanner vs greedy spanner for different parameters. (a): Size. $t = 2, \theta = 0.349066$ and $w = 0.0488$. (b): Degree. $t = 1.5, \theta = 0.174533$ and $w = 0.072$. (c): weight*. $t = 1.5, \theta = 0.034907$ and $w = 0.1489$. (d): Time. $t = 2, \theta = 0.174533$ and $w = 0.1555$

6. Conclusion

In this paper, we proposed an algorithm to construct the gap-greedy spanner using WSPD, in $O(n^3)$ time using $O(n)$ space. Although the greedy spanner is a high-quality spanner and the theoretical total weight of the greedy spanner is better than the theoretical total weight of the gap-greedy spanner, to the best our knowledge, there is still no algorithm taking $O(n^2)$ time to construct the greedy spanner. In this paper, we showed that there is an algorithm to construct the gap-greedy spanner in $O(n^2)$ time using $O(n^2)$ space. Also, we improved the theoretical dilation of the gap-greedy spanner. In this paper, we also proposed three other cubic time algorithms to construct the gap-greedy spanner and presented some experimental results on their running time and some properties of the generated graph. By the experiments we concluded that one of the proposed algorithm, i.e. Bina algorithm, is much faster than the original gap-greedy algorithm (Orig), in practice. Furthermore, we experimentally compared the geometric properties and the running time of gap-greedy spanner and the greedy spanner. The experiments show the greedy spanner is better than the gap-greedy spanner in terms of size, degree, weight* and the running time, in practice. We leave open the following problem.

Is the greedy spanner subgraph of gap-greedy spanner? We conjecture that the greedy spanner for any point set in the plane and with any dilation $t > 1$ is a subgraph of the gap-greedy spanner of the point set with some special parameters θ and w depend only on t. If the conjecture is correct, then to construct the greedy spanner, one can construct the gap-greedy spanner in $O(n^2)$ time using the our proposed technique, then using the pruning technique find the greedy spanner in $O(n\log n)$.

References

[1] D. Peleg, and A.A. Schäffer, "Graph Spanners, "*J. Graph Theory*, vol. 13, no.1, pp. 99–116, 1989.

[2] P. Chew, "There is a Planar Graph Almost as Good as the Complete Graph," Proc. A*nnual ACM Symp. Computational Geometry*, pp. 169–177, 1986.

[3] B. Chandra, G. Das, G. Narasimhan, and J. Soares, "New Sparseness Results on Graph Spanners," *Int. J. Computational Geometry & Application*, vol. 5, pp. 125–144, 1995.

[4] D. Eppstein, "Spanning Trees and Spanners, "*Handbook of Computational Geometry*, pp. 425–461, 1999.

[5] T. Lukovszki, "*New Results on Geometric Spanners and Their Applications*," Ph.D. Dissertation, Heinz Nixdorf Institute and Department of Mathematics and Computer Science, Paderborn University, Paderborn, Germany, 1999.

[6] G. Narasimhan, and M. Smid, *Geometric SpannerNetworks*, Cambridge University Press, 2007.

[7] M. Smid, "Closest Point Problems in Computational Geometry,"*Handbook on Computational Geometry*, 1997.

[8] S. Arya, and M. Smid, "Efficient Construction of aBounded-Degree Spanner with Low Weight," *Algorithmica*, vol. 17, no.1, pp. 33–54, 1997.

[9] I. Althöfer, G. Das, D. Dobkin, D. Joseph, and J. Soares,"On Sparse Spanners of Weighted Graphs," *Discrete Computational Geometry*, vol. 9, no. 1, pp. 81–100, 1993.

[10] G. Das, and G. Narasimhan, "A Fast Algorithm for Constructing Sparse Euclidean Spanners," *Int. J. Computational Geometry & Application*, vol. 7, no.4, pp. 297–315, 1997.

[11] P. Bose, P. Carmi, M. Farshi, A. Maheshwari, and M. Smid, "Computing the Greedy Spanner in Near-Quadratic Time," *Algorithmica*, vol. 58, no.3, pp. 711–729, 2009.

[12] P.S. Alewijnse, W.Q. Bouts, A.P. Ten Brink, and K. Buchin, "Computing the Greedy Spanner in Linear Space," *Algorithmica*, vol. 73, no.3, pp. 589–606, 2015.

[13] P.B. Callahan, and S.R. Kosaraju, "A Decomposition of Multidimensional Point Sets with Applications to k-Nearest-Neighbors and n-Body Potential Fields," *J. ACM*, vol. 42, no.1, pp. 67–90, 1995.

[14] S. Arya, D.M. Mount, and M. Smid, "Randomized and Deterministic Algorithms for Geometric Spanners of Small Diameter," *Proc.35th IEEE Symp. Foundations of Computer Science*, pp. 703–712, 1994.

[15] P.B. Callahan, and S.R. Kosaraju, "Faster Algorithms for Some Geometric Graph Problems in Higher Dimensions," *Proc. Fourth annual ACM-SIAM Symp. Discrete algorithms*, pp. 291–300, 1993.

[16] L. Barba, P. Bose, M. Damian, R. Fagerberg, W.L. Keng, J. O'Rourke, A. van Renssen, P. Taslakian, S. Verdonschot, and G. Xia, "New and Improved Spanning Ratios for Yao Graphs,"*J. Computational Geometry*, vol. 6, no.2, pp. 19–53, 2015.

Davood Bakhshesh received his B.S. degree in Computer Science from Vali-E-Asr University of Rafsanjan, Iran in 2006 and the M.S. degree in Computer Science from Sharif University of Technology, Iran in 2009. He is now a Ph.D. candidate at Department of Computer Science in Yazd University, Iran. His research interests include Computational Geometry, Computer Aided Geometric Design.
Email: dbakhshesh@gmail.com

Mohammad Farshi received his B.S. degree in Applied Mathematics from Yazd University in 1996, the M.S. in Pure Mathematics from Shiraz University in 1999 and Ph.D. degree in Computer Science from TU/e, The Netherlands in 2008. He is currently an assistant professor in Department of Computer Science at Yazd University, Iran. His research interests include Computational Geometry especially Geometric Spanners.
Email: mfarshi@yazd.ac.ir

Paper Handling Data:

Submitted: 23.11.2016
Received in revised form: 11.03.2017
Accepted: 18.03.2017
Corresponding author: Dr. Mohammad Farshi,
Department of Computer Science, Yazd University, Yazd, Iran.