Expected Case Analysis of β-Skeletons with Applications to the Construction of Minimum-Weight Triangulations (Extended Abstract) *

Siu Wing Cheng Mordecai J. Golin Jeffrey C.F. Tsang

Abstract: This note is divided into two parts. The first mathematically analyzes some properties of the β-skeleton of a set of n points independently chosen from the uniform distribution over the unit square and compares this analysis to empirically derived data. The second part describes a dynamic programming algorithm to construct the minimum weight triangulation for a planar point set.

1 Introduction

Let S be a finite set of points in the plane. A *triangulation* of S is a maximal collection of non-intersecting edges whose endpoints are all in S. The *weight* of a triangulation is the sum of the lengths of its edges. A *minimum weight triangulation* (MWT) of S is a triangulation that has minimum weight among all triangulations of S (Figure 1). The problem of finding a minimum-weight triangulation possesses an ambiguous status; having been open for many years it is still unknown as to whether an MWT is constructible in polynomial time.

Suppose, though, that besides being given the set S we are also given a set of edges, $E \subset S \times S$, that is known to be contained within some MWT of S and, further, that $G = (S, E)$ has k connected components. Then, it is possible to piece together various facts known about constrained MWTs and design an algorithm that finds an MWT in time $O\left(n^{k+2}\right)$ (details of such an algorithm are provided later in this note).

An obvious approach to efficiently constructing constructing an MWT would therefore be to find an edge set E such that $G = (S, E)$ has a small number of connected components or, even better, is connected.

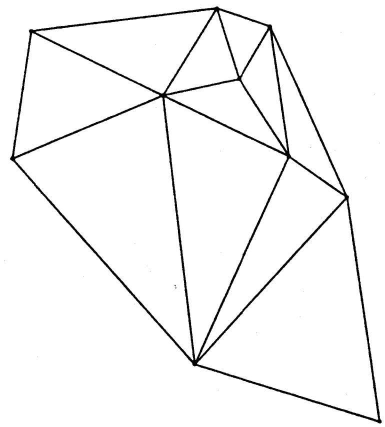

Figure 1: MWT example.

Unfortunately it is unknown how to find such a set, E. The only immediately obvious candidate E is the convex hull of S which is contained in every triangulation of S. The convex hull can be very small, though, and is therefore unhelpful because in the resulting graph can therefore have a large number of connected components.

Recently though, Keil [8] proved that the $\sqrt{2}$-skeleton of S is contained in every MWT.

At this point we digress slightly and describe what is meant by a β-skeleton. These were originally defined by Kirpatrick and Radke [9] (who defined both lune and disk based skeletons. The ones we use are, as in [8], the disk based ones). Let $p, q \in S$. The forbidden neighborhood $F(p, q)$ for p, q is defined to be the interior of the union of the two disks of radius $\beta \cdot d(p, q)/2$ that pass through both p and q (Figure 2). The edges in the

*Authors' address: Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong. Authors email: {scheng, golin, csjeff}@cs.ust.hk. The work of the first author was supported by HK RGC CRG grant HKUST 190/93E The work of the last two authors was supported by HK RGC CRG grant HKUST 181/93E.

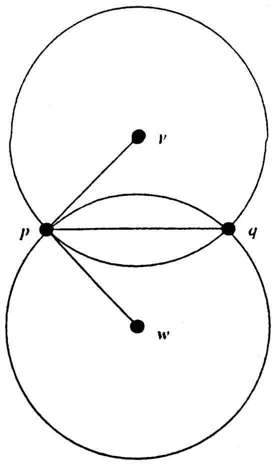

Figure 2: The forbidden neighborhood $F(p,q)$. $|pv| = |pw| = \beta|pq|/2$.

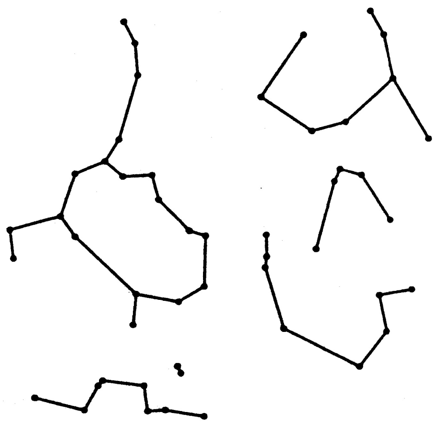

Figure 3: β-skeleton for 50 points; $\beta = 1.17682$.

β-skeleton of S are exactly the set of edges

$$B(S) = \{(p,q) : F(p,q) \cap \{S \setminus \{p,q\}\} = \emptyset\}$$

(Figure 3). The 1-skeleton (i.e., the β-skeleton with $\beta = 1$) is the well known Gabriel graph.

Cheng and Xu [3] improved upon Keil's result and proved that the β-skeleton is contained in a MWT for $\beta \geq 1.17682$.

In this note we examine some properties of the β-skeleton of random points and discuss the implications of our findings on the study of MWTs.

More specifically we assume that S_n is a set of n points chosen independently from the uniform distribution over the unit square in $\Re^2$. The *cost* of the β-skeleton is defined to be the sum of the lengths of its edges. In section 2 we mathematically prove that the expected number of edges in the β-skeleton of S_n is asymptotically equal to $b_\beta \cdot n$ while the expected cost is asymptotically $c_\beta \cdot \sqrt{n}$ where b_β and c_β are calculable constants dependent upon β. We also prove that the expected number of isolated points (points with no neighbors) in the β-skeleton is $\Omega(n)$. Finally, we present empirically observed data collected by constructing the β-skeleton of random points and compare this data to our mathematical predictions.

In section 3 we first describe an algorithm for constructing an MWT of n points in $O(n^{k+2})$ time given a graph G contained in the MWT that has at most k connected components. We are currently experimenting with this algorithm to construct minimum weight triangulations for moderately sized random point sets.

It is known [5] that the weight of the MWT of n random points grows asymptotically as $c\sqrt{n}$ where c is some unknown constant. The results of this section permit us to make a preliminary guess as to what c might be. We conclude this section with a discussion of what the results of Section 2 imply about the efficiency of our routines as n becomes large.

2 β-Skeletons

In this section we describe some properties of the β-skeleton of random points. More specifically we assume that $\beta \geq 1$ is fixed and S_n is a set of n points chosen independently from the uniform distribution over the unit square $[0,1]^2$ and then mathematically analyze how properties of the skeleton evolve as n grows to infinity.

Theorem 1 *Let β, S_n be as defined above. Set $B_n = |B(S_n)|$ to be the number of edges in the β-skeleton and $C_n = \sum_{(p,q)\in B(S_n)} d(p,q)$ to be the cost of the β-skeleton. Then*

$$\mathbf{E}(B_n) \sim \frac{\pi}{2a_\beta} n \quad \textit{and} \quad \mathbf{E}(C_n) \sim \frac{1}{4}\left(\frac{\pi}{a_\beta}\right)^{3/2} \sqrt{n}$$

where

$$a_\beta = \frac{\pi\beta^2}{2} + \frac{\sqrt{\beta^2 - 1}}{2} - \frac{\beta^2}{4}\cos^{-1}\left(\frac{\beta^2 - 2}{\beta^2}\right)$$

Proof. The value a_β is a quite natural one in this setting; it arises because $Area(F(p,q)) = a_\beta\,(d(p,q))^2$. (This can be proven by standard geometric arguments).

Now, for all $p, q \in S_n$, $p \neq q$ define

$$X_{p,q} = \begin{cases} 1 & \text{if } F(p,q) \cap \{S_n \setminus \{p,q\}\} = \emptyset \\ 0 & \text{otherwise} \end{cases},$$

$$Y_{p,q} = \begin{cases} d(p,q) & \text{if } F(p,q) \cap \{S_n \setminus \{p,q\}\} = \emptyset \\ 0 & \text{otherwise} \end{cases}$$

(if $p = q$ set $X_{p,q} = Y_{p,q} = 0$). Let p, q be random points in S_n. Summing over all p, q and using symmetry shows that

$$\mathbf{E}(B_n) = \binom{n}{2} \cdot \mathbf{E}(X_{p,q}) \text{ and } \mathbf{E}(C_n) = \binom{n}{2} \cdot \mathbf{E}(Y_{p,q}).$$

To prove the theorem it therefore suffices to evaluate $\mathbf{E}(X_{p,q})$ and $\mathbf{E}(Y_{p,q})$.

Define W be the region containing all points within distance $\frac{\beta \ln n}{\sqrt{n}}$ of the boundary of the unit square (Figure 2). Our approach is to calculate $\mathbf{E}(X_{p,q})$ by using the formula

$$\mathbf{E}(X_{p,q}) = \Pr(X_{p,q} = 1, p \in W) + \Pr(X_{p,q} = 1, p \notin W)$$

Now let p be fixed and set $\alpha = d(p,q)$. If $\alpha > \frac{\ln n}{\sqrt{n}}$ then

$$Area(F(p,q) \cap [0,1]^2) \geq \pi \frac{\ln^2 n}{4n}$$

so

$$\begin{aligned} \Pr\left(X_{p,q} = 1 \,|\, d(p,q) \geq \frac{\ln n}{\sqrt{n}}\right) &\leq \left(1 - \frac{\pi \ln^2 n}{4n}\right)^{n-2} \\ &= n^{-\Omega(\ln n)}. \end{aligned}$$

Thus

$$\begin{aligned} \Pr(X_{p,q} = 1) &\leq \Pr\left(d(p,q) \leq \frac{\ln n}{\sqrt{n}}\right) + n^{-\Omega(\ln n)} \\ &= O\left(\frac{\ln^2 n}{n}\right). \end{aligned}$$

This last equation is true for any fixed p. We can therefore combine it with $\Pr(p \in W) = O\left(\frac{\ln n}{\sqrt{n}}\right)$ to find

$$\Pr(X_{p,q} = 1,\, p \in W) = \Pr(X_{p,q} = 1 \,|\, p \in W) \cdot \Pr(p \in W)$$

is $O\left(\frac{\ln^3 n}{n^{3/2}}\right)$.

We must now calculate

$$\Pr(X_{p,q} = 1,\, p \notin W) = \Pr(X_{p,q} = 1 \,|\, p \notin W)\Pr(p \notin W)$$

where $\Pr(p \notin W) = 1 - O\left(\frac{\ln n}{\sqrt{n}}\right)$.

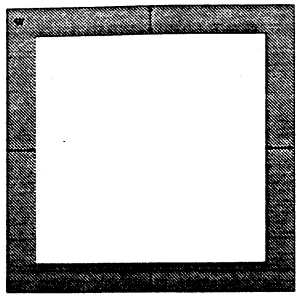

Figure 4: W is the gray region in the figure.

To continue, notice that if $p \notin W$ and $\alpha \leq \frac{\ln n}{\sqrt{n}}$ then $\Pr(d(p,q) \leq \alpha) = \pi\alpha^2$ so $\frac{d}{d\alpha}\Pr(d(p,q) \leq \alpha) = 2\pi\alpha$. Furthermore, for these α, $F(p,q)$ is totally contained inside of $[0,1]^2$ so

$$\Pr(X_{p,q} = 1 \,|\, p \notin W, d(p,q) = \alpha) = (1 - a_\beta \alpha^2)^{n-2}.$$

Combining everything we therefore find that

$$\begin{aligned} \mathbf{E}(X_{p,q}) &\sim \Pr(X_{p,q} = 1,\, p \notin W) \\ &\sim \int_0^{\frac{\ln n}{\sqrt{n}}} (1 - a_\beta \alpha^2)^{n-2}\, 2\pi\alpha \, d\alpha \\ &\sim \frac{\pi}{a_\beta n} \end{aligned}$$

and $\mathbf{E}(B_n) = \binom{n}{2} \cdot \mathbf{E}(X_{p,q}) \sim \frac{\pi}{2a_\beta} n$.

A similar calculation shows that,

$$\begin{aligned} \mathbf{E}(Y_{p,q}) &\sim \int_0^{\frac{\ln n}{\sqrt{n}}} (1 - a_\beta \alpha^2)^{n-2}\, 2\pi\alpha^2 \, d\alpha \\ &\sim \frac{1}{2}\left(\frac{\pi}{a_\beta n}\right)^{3/2} \end{aligned}$$

and $\mathbf{E}(C_n) = \binom{n}{2} \cdot \mathbf{E}(Y_{p,q}) \sim \frac{1}{4}\left(\frac{\pi}{a_\beta}\right)^{3/2} \sqrt{n}$. □

Table 1 presents observed data on the number of edges and costs of β-skeletons observed for different values of β.

Note that when $\beta = 1.18$ the β-skeleton contains roughly $0.87n$ edges on average. For moderately sized n this graph will have a relatively small number of connected components. We can therefore use a dynamic programming algorithm to find the MWT of the point set using the β-skeleton plus the convex hull as a starting set of edges. (See Section 3.) We should point out, though, that as n gets large this approach can no longer be used. This follows from the following observation:

Theorem 2 *Let $\beta \geq 1$ be fixed and S_n. For $i = 0, 1, 2, 3, \ldots$ define*

$$V_i(n) = \mathrm{E}\left(|\{p \in S_n : p \text{ has degree } i \text{ in } B(S_n)\}|\right).$$

Then there exists v_i such that $\lim_{n\to\infty} \frac{V_i(n)}{n} = v_i$. Furthermore $v_0 > 0$.

Proof. Omitted in this extended abstract except to mention that this can be proven using techniques developed in [5]. □

The theorem implies that the expected number of isolated points in $B(S_n)$ grows linearly in n so the expected number of connected components also grows linearly in n (since the expected size of the convex hull is $O(\ln n)$ adding the convex hull edges will therefore not dramatically change the number of connected components). Thus, as n gets large we are guaranteed that the number of components will grow large and that the dynamic programming algorithm will become very slow.

3 Minimum-Weight Triangulations

Overview. In this section, we describe an exhaustive search algorithm for finding a minimum weight triangulation of n points in the plane. It consists of two phases. The first phase identifies the convex hull and the 1.18-skeleton. The Graham-scan algorithm [6] is used to find the convex hull in $O(n \log n)$ time. β-skeletons can be constructed in $O(n \log n)$ time [9]. Taking advantage of the fact that the points being processed are randomly chosen from the unit square. We can construct the Voronoi diagram using the $O(n)$ expected time Voronoi diagram algorithm of [1]. Using the same techniques in [9], we are able to construct the β-skeleton in $O(n)$ time from the Voronoi diagram. Let H_0 be the ***plane graph*** obtained at the end of the first phase. A plane graph is a fixed plane embedding of a planar graph. Thus, H_0 induces a planar subdivision. Let k be the number of connected components in H_0. The second phase adds $k-1$ ***diagonals*** to H_0 to make it connected and then use dynamic programming to find the optimal triangulation of each polygon in the planar subdivision. A line segment is a diagonal if its endpoints are in H_0 and it does not cross any existing edge. If the $k-1$ diagonals are in a minimum weight triangulation, then the collection of the triangulated polygons is a minimum weight triangulation. Hence, backtracking is used to exhaust all possible choices of the $k-1$ diagonals. The running time is $O(n^{k+2})$. We provide below the details and timing analysis of the second phase.

Data structures. At any moment in time, one plane graph H_i, $0 \leq i \leq k-1$, is maintained which is represented by adjacency lists. Each vertex of H_i is a point p in S. $L_i(p)$ is the adjacency list for p organized as a doubly linked list. The vertices in $L_i(p)$ are ordered in clockwise order around p. The connected components $C_{i1}, C_{i2}, \ldots, C_{ik}$ of H_i are stored in a doubly linked list CL_i. C_{i1} contains the convex hull of S and it is always the leftmost entry in CL_i. Each C_{ij} in CL_i is stored as a list of vertices and edges. The vertices are labeled such that given any $p \in H_i$, the C_{ij} containing p can be reported in $O(1)$ time. Each C_{ij} is also associated with its convex hull $\mathsf{conv}\ C_{ij}$ which is represented as an ordered sequence of vertices and edges. For each vertex p on $\mathsf{conv}\ C_{ij}$, we maintain a list $\boldsymbol{diag_i}(p)$ of diagonals in H_i with p as one endpoint and each such diagonal does not intersect the interior of $\mathsf{conv}\ C_{ij}$.

Backtracking algorithm. Recall that H_0 is the plane graph containing the convex hull and the 1.18-skeleton of S. The recursion bottoms out at the kth recursive call. CL_{k-1} contains one connected component and dynamic programming is used to optimally triangulate each polygon in H_{k-1}. The solution is used to update the current best solution. In the ith recursive call $1 \leq i \leq k-1$, we try to add a diagonal to H_{i-1} as follows. Take an arbitrary vertex p in $\mathsf{conv}\ C_{i-1,2} \in CL_{i-1}$. Remove a diagonal $e \in \boldsymbol{diag_{i-1}}(p)$, add e to H_{i-1} to generate H_i, and remove e' from $\boldsymbol{diag_{i-1}}(q)$ for every e' and q such that e crosses e'. Suppose that e connect $C_{i-1,2}$ and $C_{i-1,j}$. Then remove $C_{i-1,j}$ from CL_{i-1}, merge $C_{i-1,2}$ with $C_{i-1,j}$, and merge $\mathsf{conv}\ C_{i-1,2}$ and $\mathsf{conv}\ C_{i-1,j}$. We are now ready to make the $(i+1)$th recursive call to process H_i. After returning, restore H_{i-1}, diagonals intersecting with e, and CL_{i-1}. Then remove another diagonal f from $\boldsymbol{diag_{i-1}}(p)$, generate a new H_i, and make another recursive call. This is repeated until $\boldsymbol{diag_{i-1}}(p)$ becomes empty. Then restore $\boldsymbol{diag_{i-1}}(p)$ and return. This completes the description of the backtracking algorithm. We provide below the details of identifying diagonals, merging connected components, and optimally triangulating a polygon.

Diagonals. For each vertex p in H_0, $\boldsymbol{diag_0}(p)$ is computed before the backtracking algorithm starts. Also, for each diagonal e, we maintain a doubly linked list of diagonals in H_0 that intersect e. Thus, in the ith recursive call, when e is added to H_{i-1}, it suffices to scan this list for e in order to remove all diagonals that intersect e. Note that this list may contain some redundant diagonals that do not exist in H_{i-1}. Thus, in each recursive call, we could spend $O(n^2)$ time for this which leads to $O(kn^2)$ time for each possible sequence of H_i, $1 \leq i \leq k-1$.

Merging. The merging of two connected components $C_{i-1,2}$ and $C_{i-1,j}$ is due to the insertion of a diagonal e to H_{i-1}. Thus, it suffices to put the two adjacency lists for the two connected components together. Insertion of e to the adjacency lists of its two endpoints takes linear time because we need to search for the correct position in the sorted order. The two convex hulls conv $C_{i-1,2}$ and conv $C_{i-1,j}$ can be merged in linear time [11]. Therefore, for each possible sequence of H_i, $1 \leq i \leq k-1$, $O(kn)$ time is spent for merging.

Triangulate polygons. A polygon P of size m is optimally triangulated in $O(m^3)$ time by a dynamic programming algorithm [4]. For completeness sake, we provide the recurrence relation below. Number the vertices of the polygon from p_1 to p_m in the clockwise order. For any p_i, p_j, p_k in clockwise order such that p_ip_k, p_kp_j and p_ip_j are diagonals or edges of P, $p_ip_kp_j$ is the triangle formed by them. For any p_i, p_j such that p_ip_j is a diagonal or an edge of P, Δ_{ij} is a minimum weight triangulation of the polygon enclosed by the p_ip_j and the edges of P traversed clockwisely from p_i to p_j. Then Δ_{ij} equals to $\Delta_{ik} \cup \Delta_{kj} \cup p_ip_kp_j - \{p_ip_k, p_kp_j\}$ for the choice of k such that p_ip_k and p_kp_j are diagonals or edges of P and the resulting weight is minimized. Thus, computing each Δ_{ij} takes $O(m)$ time which leads to a total of $O(m^3)$ time. Δ_{1m} is the solution desired.

Theorem 3 *The MWT for a set S of n points can be computed in $O(n^{k+2})$ time, where k is the number of connected components of the plane graph consisting of the convex hull and the 1.18-skeleton of S.*

Proof. The backtracking algorithm is essentially an exhaustive search, except that only diagonals incident to an arbitrary vertex on the boundary of conv C_{i2} may be added to H_{i-1} to obtain H_i. This is correct because every vertex on the boundary of conv C_{i2} must connect to some other connected component along a diagonal that does not intersect the interior of conv C_{ij}. Otherwise, in the final triangulation, some triangle will contain an internal angle greater than π which is impossible. For each possible sequence of H_i, $1 \leq i \leq k-1$, we need to spend $O(kn^2)$ time to eliminate diagonals, $O(kn)$ time to merge connected components and convex hulls, and $O(n^3)$ time to optimally triangulate all the polygons in H_{k-1}. There are n^{k-1} possible sequences and therefore, the total running time is $O(n^{k-1} \cdot n^3) = O(n^{k+2})$. □

References

[1] J.L. Bentley, B.W. Weide, and A.C. Yao, "Optimal Expected-Time Algorithms for Closest Point Problems," *ACM Trans. on Mathematical Software*, **6**(4) (Dec. 1980) 563-580.

[2] R. C. Chang and R.C.T. Lee, "On the Average Length of Delaunay Triangulations," *BIT*, **24** (1984) 269-273.

[3] Siu-Wing Cheng and Xu, Yin-Feng, "Approach the Largest β-skeleton within a Minimum Weight Triangulation," Manuscript, (1995).

[4] P. Gilbert, "New Results on Planar Triangulations," *Master's Thesis, University of Illinois*, 1979. Report No. UILUENG 78 2243.

[5] M. J. Golin, "Probabilistic Recurrence Relations, Euclidean Functionals and Minimum Weight Triangulations," Manuscript, (1995).

[6] R.L. Graham, "An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set," *Information Processing Letters*, **1** (1972) 132–133.

[7] L. Heath and S. Pemmaraju, "New Results for the Minimum Weight Triangulation Problem," *Algorithmica*, **12** (1994) 533–552.

[8] J. Mark Keil, "Computing a Subgraph of the Minimum Weight Triangulation," *Computational Geometry: Theory and Applications*, **4** (1994) 13-26.

[9] D. G. Kirpatrick and J. D. Radke, "A Framework for Computational Morphology," *Computational Geometry (G.T. Toussaint, ed.)*, (1985) 217-248.

[10] C. Levcopoulos and A. Lingas, "Greedy Triangulation Approximates the Minimum Weight Triangulation in the Average Case and Can be Computed in Linear Time in the Average Case," *Proceedings of the 3rd International Conference on Computing and Information (ICCI'91)*, **Lecture Notes in CS, 497** (1991) 139-148.

[11] F.P. Preparata and M.I. Shamos, "Computation Geometry," *Texts and Monographs in Computer Science (David Gries, ed.)*, (1985).

β	n	100	200	300	400	500	600	700	800	900	1000	predicted values
1	edges	195.49	395.42	594.82	794.54	995.97	1192.51	1397.51	1593.09	1794.43	1992.84	
	edges/n	1.95	1.98	1.98	1.99	1.99	1.99	2.00	1.99	1.99	1.99	2.00
	cost	19.10	27.43	33.67	38.98	43.82	47.82	52.00	55.46	58.96	62.13	
	cost/$\sqrt{n}$	1.91	1.94	1.94	1.95	1.96	1.95	1.97	1.96	1.97	1.96	2.00
1.1	edges	109.80	221.30	335.34	443.67	554.67	668.64	777.72	883.73	998.98	1107.28	
	edges/n	1.10	1.11	1.12	1.11	1.11	1.11	1.11	1.10	1.11	1.11	1.09
	cost	8.06	11.60	14.29	16.37	18.33	20.22	21.74	23.09	24.66	25.84	
	cost/$\sqrt{n}$	0.81	0.82	0.83	0.82	0.82	0.83	0.82	0.82	0.82	0.82	0.81
1.18	edges	90.41	179.35	269.76	358.29	447.90	538.58	625.90	714.62	803.90	892.03	
	edges/n	0.90	0.90	0.90	0.90	0.90	0.90	0.89	0.89	0.89	0.89	0.87
	cost	5.93	8.38	10.38	11.93	13.33	14.65	15.71	16.77	17.78	18.77	
	cost/$\sqrt{n}$	0.59	0.59	0.60	0.60	0.60	0.60	0.59	0.59	0.59	0.59	0.58
1.2	edges	84.97	171.09	259.42	342.02	426.94	512.44	596.67	681.49	766.19	849.96	
	edges/n	0.85	0.86	0.86	0.86	0.85	0.85	0.85	0.85	0.85	0.85	0.83
	cost	5.47	7.90	9.77	11.11	12.32	13.62	14.64	15.61	16.56	17.46	
	cost/$\sqrt{n}$	0.55	0.56	0.56	0.56	0.55	0.56	0.55	0.55	0.55	0.55	0.54
1.3	edges	68.73	141.29	209.36	279.10	347.70	414.72	484.19	550.40	622.35	690.53	
	edges/n	0.69	0.71	0.70	0.70	0.70	0.69	0.69	0.69	0.69	0.69	0.67
	cost	4.04	5.94	7.16	8.20	9.13	9.90	10.74	11.37	12.20	12.78	
	cost/$\sqrt{n}$	0.40	0.42	0.41	0.41	0.41	0.40	0.41	0.4	0.41	0.40	0.39
1.4	edges	59.35	120.01	174.76	232.55	290.54	349.03	403.20	463.82	520.92	576.01	
	edges/n	0.59	0.60	0.58	0.58	0.58	0.58	0.58	0.58	0.58	0.58	0.56
	cost	3.15	4.64	5.41	6.27	7.00	7.65	8.17	8.80	9.32	9.78	
	cost/$\sqrt{n}$	0.32	0.33	0.31	0.31	0.31	0.31	0.31	0.31	0.31	0.31	0.30
$\sqrt{2}$	edges	58.29	114.56	171.10	229.27	286.83	338.22	396.77	451.87	509.02	564.57	
	edges/n	0.58	0.57	0.57	0.57	0.57	0.56	0.57	0.56	0.57	0.56	0.55
	cost	3.14	4.34	5.27	6.08	6.82	7.32	7.93	8.49	8.99	9.46	
	cost/$\sqrt{n}$	0.31	0.31	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.29
1.5	edges	47.55	99.96	149.33	199.46	249.15	298.44	344.92	395.60	442.54	494.43	
	edges/n	0.48	0.50	0.50	0.50	0.50	0.50	0.49	0.49	0.49	0.49	0.48
	cost	2.43	3.48	4.26	4.99	5.53	6.05	6.47	6.94	7.31	7.75	
	cost/$\sqrt{n}$	0.24	0.25	0.25	0.25	0.25	0.25	0.24	0.25	0.24	0.25	0.24
1.6	edges	43.93	87.86	129.03	171.04	214.83	255.63	298.95	345.07	384.47	426.87	
	edges/n	0.44	0.44	0.43	0.43	0.43	0.43	0.43	0.43	0.43	0.43	0.42
	cost	2.08	2.91	3.46	3.97	4.46	4.85	5.24	5.68	5.92	6.21	
	cost/$\sqrt{n}$	0.21	0.21	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.19
1.7	edges	39.40	77.03	114.42	149.46	188.41	225.98	262.91	298.82	336.66	373.76	
	edges/n	0.39	0.39	0.38	0.37	0.38	0.38	0.38	0.37	0.37	0.37	0.36
	cost	1.84	2.39	2.89	3.24	3.66	3.98	4.30	4.60	4.88	5.10	
	cost/$\sqrt{n}$	0.18	0.17	0.17	0.16	0.16	0.16	0.16	0.16	0.16	0.16	0.16
1.8	edges	35.22	67.80	102.52	135.09	167.43	198.52	230.52	267.56	298.87	331.77	
	edges/n	0.35	0.34	0.34	0.34	0.33	0.33	0.33	0.33	0.33	0.33	0.32
	cost	1.44	1.94	2.43	2.76	3.09	3.33	3.58	3.88	4.06	4.26	
	cost/$\sqrt{n}$	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.13	0.13
1.9	edges	28.91	61.72	91.62	120.91	148.95	177.55	209.13	235.48	266.22	294.26	
	edges/n	0.29	0.31	0.31	0.30	0.30	0.30	0.30	0.29	0.30	0.29	0.29
	cost	1.10	1.74	2.06	2.34	2.59	2.79	3.05	3.21	3.41	3.58	
	cost/$\sqrt{n}$	0.11	0.12	0.12	0.12	0.12	0.11	0.12	0.11	0.11	0.11	0.11
2	edges	27.25	56.38	80.05	108.32	135.20	162.96	185.77	213.89	239.97	266.06	
	edges/n	0.27	0.28	0.27	0.27	0.27	0.27	0.27	0.27	0.27	0.27	0.26
	cost	0.96	1.47	1.69	1.99	2.22	2.44	2.55	2.76	2.92	3.06	
	cost/$\sqrt{n}$	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10

Table 1: Data collected during the construction of β-skeletons. For every β and every n, 100 random point sets of size n were generated. Each entry in an *edges* row contains the average number of edges in the 100 constructed skeletons while the entries in the *cost* rows contain the costs. Each entry is presented along with its appropriately scaled value. The last column contains the predicted values from our analysis.