The Path of a Triangulation

Oswin Aichholzer*

Institute for Theoretical Computer Science
Graz University of Technology
Klosterwiesgasse 32/1, A-8010 Graz, Austria
e-mail: oaich@igi.tu-graz.ac.at

Abstract

For a planar point set S let T be a triangulation of S and l a line properly intersecting T. We show that there always exists a unique path in T with certain properties with respect to l. This path is then generalized to (non triangulated) point sets restricted to the interior of simple polygons. This so-called triangulation path enables us to treat several triangulation problems on planar point sets in a divide & conquer-like manner. For example, we give the first algorithm for counting triangulations of a planar point set which is observed to run in time sublinear in the number of triangulations. Moreover, the triangulation path proves to be useful for the computation of optimal triangulations.

1 Introduction

In this paper we introduce a new concept called *triangulation path* or *t-path* for short. Consider a planar point set S, a triangulation T of S and a line l properly intersecting T. There are two edges of the convex hull of S intersected by l. A *path* is a monotone "zig-zag" chain of edges from T connecting these two convex hull edges. Thereby zig-zag means that all edges of the chain intersect l, and monotone means that these intersections are sorted on l. It is easy to show that for any triangulation T of S there exists such a path with respect to l. Moreover we may require some additional properties, i.e. the area bounded by the path and l has to be free of points from S. In this case we will show that any triangulation of S has a unique t-path for each line l.

We then generalize the path-concept to planar point sets. In the main theorem we show that triangulation paths of different triangulations of a given point set are either identical or properly intersect each other. This gives us a unique relation between triangulations and their paths, such that we can apply the triangulation path in a divide & conquer-like manner.

As a first application, we show how to count the number of triangulations of a planar point set by splitting it into independent sub-parts and treating them recursively. This gives us the first algorithm which runs in time sublinear in the number of triangulations counted. If we define the cost of the computation as the running time per triangulation we get an algorithm with decreasing cost for increasing problem size.

The best previously known methods to count triangulations are based on reverse search and have a running time of $\Theta(n \cdot t(S))$, where $n = |S|$ and $t(S)$ is the number of triangulations of S, see Avis and Fukuda [2]. Thus the cost of this approach increases linearly with the input size.

A related problem is the generation of random triangulations. We show how to utilize the results obtained for counting to produce random triangulations such that any triangulation of the given point set appears with the same probability.

As a further application we investigate optimal triangulations. In recent years, several subgraphs of optimal triangulations have been identified, like the β-skeleton [18] or the LMT-skeleton [4, 9]. In general, such a skeleton will consist of k disconnected parts. An approach proposed e.g. in Cheng et al. [6] tries all possibilities to add $k-1$ edges to obtain a connected graph and completes each of these graphs to a triangulation by optimally triangulating its faces. This approach, which basically is exhaustive search, can be implemented to run in $O(n^{k+2})$ time. Instead of a simple exhaustive investigation of all possibilities to connect the separated subgraphs we show how to use the triangulation path for this task. We expect this approach

*Research supported by APART [Austrian Programme for Advanced Research and Technology] of the Austrian Academy of Sciences and the Spezialforschungsbereich F 003 *Optimierung und Kontrolle*.

to speed up the computation for extremely 'bad' examples for the LMT-skeleton.

2 The Path

In this section we define the main data structure of this paper. To simplify the argumentation we first restrict the t-path to a fixed triangulation. In Section 2.2 we generalize the concept to untriangulated simple polygons with points in their interior, showing the usefulness for the mentioned applications.

2.1 The Path of a Triangulation

Let S be a planar point set and let E be the set of edges induced by S. A subset c of pairwise non-intersecting edges from E of size k is a *chain* with *length* k if there exists a labeling $e_1 \dots e_k$ of the edges in c such that any consecutive pair e_i and e_{i+1}, $1 \leq i \leq k-1$, shares exactly one common endpoint which is the endpoint of e_i and the starting point of e_{i+1}. The edges e_1 and e_k are called start and end edge, respectively.

By definition, a chain contains no cycles of length two, since two edges are not allowed to share both endpoints simultaneously. On the other hand several edges may share the same vertex, i.e. a chain may contain cycles.

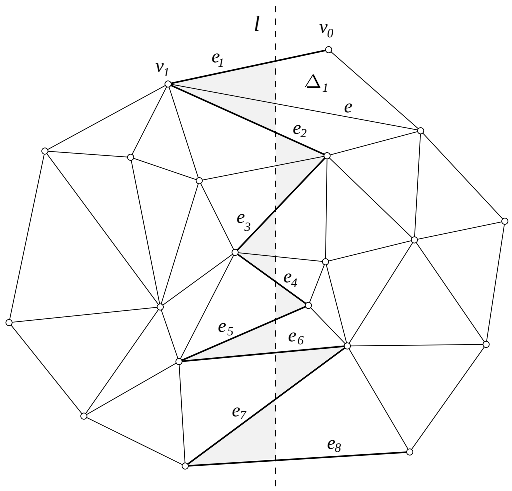

Figure 1: The path of a triangulation with respect to a line l. Shaded areas have to be free of points.

We are now ready to define a triangulation path (refer to Figure 1):

Definition 1 *Given a planar point set S, a triangulation T of S, and a line l avoiding S but intersecting the convex hull of S. The* triangulation-path *of T with respect to l, $p_l(T)$, is defined as follows:*

1. *$p_l(T)$ is a chain of T where every edge of $p_l(T)$ intersects l.*
2. *Start and end-edges of $p_l(T)$ are the two edges of the convex hull of S intersected by l.*
3. *From property 1 it follows that any two consecutive edges of $p_l(T)$ enclose a triangular area together with l. These regions have to be free of points from S (refer to the shaded area in Fig. 1).*

We first show that one can always construct a triangulation path (t-path) $p_l(T)$.

Theorem 1 *For a given planar point set S, a triangulation T of S and a line l avoiding S but intersecting T, there exists always a t-path $p_l(T)$.*

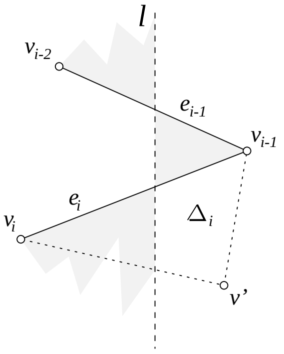

Figure 2: v_i and v' have to lie on opposite sides of l.

Proof. W.l.o.g. assume that l is a vertical line (for the whole proof ref. to Fig. 1). To construct $p_l(T)$, start with the upper convex hull edge $e_1 = (v_0, v_1)$ of T intersected by l. Let Δ_1 be the triangle in T based on e_1 and let e be the second edge of Δ_1 intersected by l. The common vertex v_1 of e_1 and e is the start point for the remaining t-path. Note that choosing v_0 instead would violate Definition 1.

We now show that starting from v_1 we can always choose consecutive edges $e_i = (v_{i-1}, v_i)$ of $p_l(T)$, $2 \leq i \leq k$, with the following properties: 1) e_i intersects l and this intersection point lies below the intersection of e_{i-1} and l. 2) The triangular area bounded by l, e_i and e_{i-1} is free of points from S. 3) e_i maximizes the angle between e_{i-1} and e_i among all edges emanating from v_{i-1} and fulfilling conditions 1) and 2).

We have to show that it is always possible to choose such edges so that after k steps we finally reach the second edge of the convex hull of S intersected by l, attaining the t-path $p_l(T)$.

Suppose that the last edge of $p_l(T)$ so far is $e_i = (v_{i-1}, v_i)$, $1 \leq i < k$. Let Δ_i be the triangle in T based on e_i and not including the vertex v_{i-2} (the triangle "below" e_i, ref. to Fig. 2). Then the third vertex of Δ_i, v', and v_i have to lie on opposite sides of l. Else by condition 3) the edge (v_{i-1}, v') had to be chosen instead of e_i. Thus the edge (v_i, v') clearly fulfills conditions 1) and 2) and we have at least one candidate to choose (according to condition 3) the next edge e_{i+1} of $p_l(T)$. $\square$

Note that the above proof gives a deterministic way to construct a t-path $p_l(T)$. The next theorem shows that this is in fact the only way to do so.

Theorem 2 *For a given planar point set S, a triangulation T of S and a line l avoiding S but intersecting T, there exists exactly one t-path $p_l(T)$.*

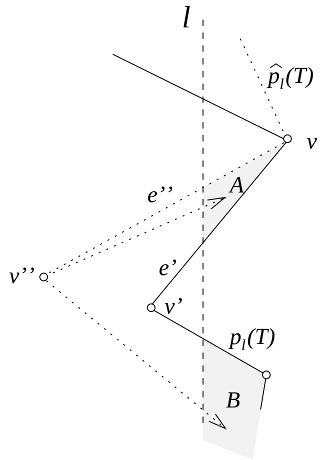

Figure 3: Not both of $p_l(T)$ and $\hat{p}_l(T)$ can be t-paths.

Proof. From Theorem 1 we know that there always exists a t-path. It remains to show that this t-path is unique. Suppose to the contrary there exist at least two different t-paths $p_l(T)$ and $\hat{p}_l(T)$. Since T is a triangulation, the edges of $p_l(T)$ and $\hat{p}_l(T)$ do not properly intersect, where properly means that they have edges which intersect in their interior. Thus if $p_l(T)$ and $\hat{p}_l(T)$ have no common vertex of S then w.l.o.g. $\hat{p}_l(T)$ has to lie entirely to the left of $p_l(T)$. But since $p_l(T)$ is a t-path, non of the edges of $\hat{p}_l(T)$ can intersect l (cf. condition 3 of Definition 1) and thus $\hat{p}_l(T)$ cannot be a t-path.

Suppose now $p_l(T)$ and $\hat{p}_l(T)$ have a vertex $v \in S$ in common but the successor of v for $p_l(T)$ and $\hat{p}_l(T)$ is different, namely v' and v'', respectively (ref. to Fig. 3). W.l.o.g. let the edge $e'' = (v, v'')$ be "above" the edge $e' = (v, v')$. Then the next but one vertex of $\hat{p}_l(T)$ can not lie above e' (shaded area A in Fig. 3) without violating condition 3 of Definition 1 for $p_l(T)$. But on the other hand it also can not lie below e' (shaded area B in Fig. 3) since then for the same reason v' would cause $\hat{p}_l(T)$ to be no valid t-path.

The argumentation remains still valid if the construction is done bottom-up (i.e. Fig. 3 is turned upside down) and thus we conclude that $p_l(T)$ and $\hat{p}_l(T)$ have all vertices in common and the theorem follows. $\square$

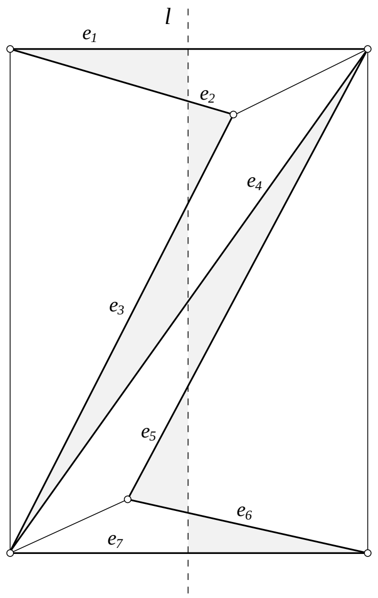

Figure 4: The path $p_l(T)$ may contain cycles and is in general not the shortest connection between the intersected convex hull edges.

From the construction given in the proof of Theorem 1 and the uniqueness given in Theorem 2 we immediately get the following.

Observation 1 *Let $p_l(T) = e_1, ..., e_k$. Then $e_1 \cap l, ..., e_k \cap l$ are sorted on l.*

Note also that from Theorem 2 it follows that the construction of the t-path given in the proof of Theorem 1 does not depend on which side of the convex hull we start, i.e. the realization of a t-path is direction independent.

That a t-path may contain cycles is shown in Figure 4. Note that all other chains which connect the two intersected edges of the convex hull would violate Definition 1 and that even the convex hull edges may be involved in a cycle.

Lemma 1 *The length of a t-path $p_l(T)$ is bounded from above by $2n - 4$ and this bound is tight.*

Proof. From Definition 1 we know that a t-path is a planar, bipartite graph and thus it has at most $2n - 4$ edges. Figure 5 shows that this bound is tight. To simplify the explanation let $n \equiv 0 \mod 4$. There are $n/2$ points lying on a horizontal line on the right side of l and two groups of $n/4$ points on two lines on the left of l. The left lines are arranged such that they 'split' the set on the right in two equal sized parts. From the given path we see that all but four points have degree 4

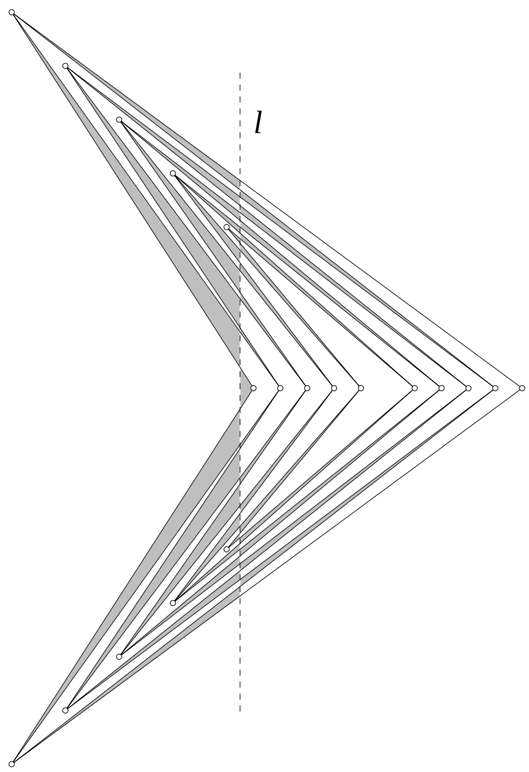

Figure 5: A path with the maximum number of $2n - 4$ edges.

and thus the path has $2n-4$ edges. A simple case analysis shows that we can achieve similar bounds for $n \not\equiv 0 \mod 4$. $\square$

2.2 The Path of a Simple Polygon

In Section 2.1 we defined the t-path for a fixed triangulation T of S. To see the usefulness of the concept we now generalize it to point sets in the plane restricted to the interior of a simple polygon.

Definition 2 *Given a planar point set S, a simple polygon P spanned by S, all enclosing points from S not lying on P, and a line l, $l \cap S = \emptyset$, intersecting P. A* triangulation path of P and S *with respect to l, $p_l(P, S)$, is defined as follows:*

1. *$p_l(P, S)$ is a chain of edges induced by S and lying on or inside P, where every edge of $p_l(P, S)$ intersects l.*
2. *Start and end-edges of $p_l(P, S)$ are two edges of P intersected by l such that the segment of l between the two intersection points lies in the interior of P.*
3. *The area bounded by consecutive edges of $p_l(P, S)$ and l must be free of points from S.*

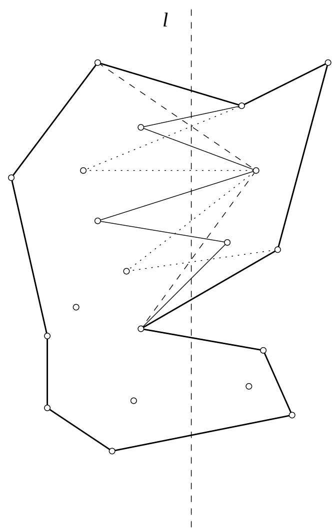

Figure 6: A simple polygon containing points with different t-paths (solid, dotted and dashed).

For the special case that P is the convex hull of S we will write $p_l(S)$ instead of $p_l(P, S)$.

Figure 6 gives an example for Definition 2, showing three different t-paths for the upper half of the simple polygon. Figure 9 gives all t-paths of a set of 7 points.

A simple but important observation is that if we triangulate the interior of P, all properties of a t-path proven in Section 2.1 still hold. In particular, Definition 1 is compatible with Definition 2 in the sense that the (unique) path of a fixed triangulation is well defined in the polygonal case, too.

Let us now state our main theorem.

Theorem 3 *Given are S, P and l according to Def. 2 and two triangulations T_1 and T_2 of P. Then the two t-paths $p_l(T_1)$ and $p_l(T_2)$ are either identical or properly intersect each other, where properly means that edges of $p_l(T_1)$ and $p_l(T_2)$ have to intersect in their interior.*

Proof. Suppose $p_l(T_1)$ and $p_l(T_2)$ do not properly intersect. But then we can apply the same argumentation as used in the proof of Theorem 2 and it follows that $p_l(T_1)$ and $p_l(T_2)$ are identical. $\square$

The merit of this theorem is that for given S, P and l a t-path $p_l(P, S)$ can not lie entirely on one side of another t-path $\hat{p}_l(P, S)$. Thus if we split P with $p_l(P, S)$ into distinct parts (again simple polygons, possibly with points in their interiors) Theorem 3 guarantees that none of these parts can be a proper subset of a part of P obtained from the splitting with $\hat{p}_l(P, S)$. In other words, we can divide P (and thus S) into independent subparts, investigate these subparts (recursively), and then combine the results. If we repeat this process for any t-path of P, S and fixed l we thus investigate all triangulations of P and S (by Theorem 1) exactly once (by Theorem 3). This can be seen as an application of the divide & conquer approach to triangulation problems; see the next sections for more details and examples. Note that if only a planar point set S is given we have the special case that P is the convex hull of S to start with.

To guarantee that we can always apply a t-path to divide P into smaller subparts, we have to show that we can always find a *non-trivial* t-path. Here trivial means a t-path consisting only of start and end edge, i.e. the two edges of P intersected by l. Obviously this is the only kind of path which does not divide P into smaller parts. In the divide & conquer approach we do not need to consider trivial paths, but to succeed we have to find at least one non-trivial path.

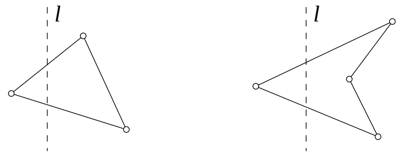

Figure 7: The two special cases with no non-trivial t-path.

Lemma 2 *Let S and P be as in Def. 2. If either P contains inner points from S, or P is no triangle or non-convex quadrilateral, then we can always find a line l with given direction such that there exists a non-trivial t-path $p_l(P, S)$.*

The lemma shows that the two trivial examples given in Figure 7 are the only cases where we cannot apply the divide & conquer approach. Moreover Lemma 2 states that we can fix the orientation of the line l, e.g. we only need to consider vertical lines (requiring w.l.o.g. that all points in S have different x-coordinates). Thus there are at most $n-1$ distinct lines l, i.e. we can find a non-trivial t-path efficiently, see also the discussion in the next section.

The proof of the lemma requires a surprisingly extensive but boring case analysis which can be found in [11].

Let us finally consider the number $|p_l(P, S)|$ of possible t-paths for given S, P and l. With $t(P)$ we denote the number of different triangulations of P. From Theorem 3 we immediately get

Lemma 3 $|p_l(P, S)| \leq t(P)$

We conjecture that the maximum for $|p_l(P, S)|$ is obtained if S is the vertex set of a convex polygon P. Therefore let l divide S into two sets of size $\lceil \frac{n}{2} \rceil$ and $\lfloor \frac{n}{2} \rfloor$. By induction over the size of the two subsets we derive that the number of t-paths for this configuration is $\binom{n-2}{\lceil \frac{n}{2} \rceil - 1}$. Compared to the $\frac{1}{n-1}\binom{2n-4}{n-2}$ triangulations of a convex n-gon we get $\Theta(2^{n-\log n})$ triangulations per t-path. This shows that in general we might have to consider an exponential number of t-paths but may also count an exponential number of triangulations per t-path.

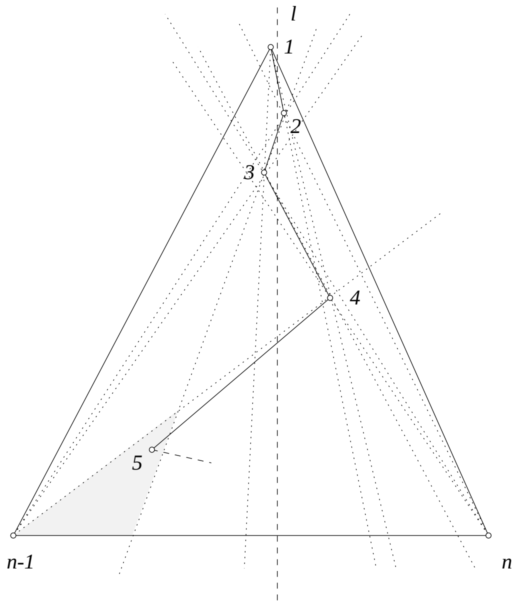

Figure 8: Point set with unique path. Points 2 to $n-2$ are put according to the line arrangement of the previous points, such that the t-path follows the fixed order.

On the other hand it is possible to construct point sets of arbitrary size which, for a given line l, have only one t-path. The idea is to insert inner points in a way that the next edge of the t-path is always forced in a unique way, see Fig. 8 for an example.

3 Divide & Conquer

Let us now take a look at the algorithmic issue of t-paths. As mentioned in the last section we may investigate problems which require all triangulations of a point set to be considered in a recursive way. First we separate the point set S into two parts by a line l. Then we compute a possible t-path $p_l(S)$ for S and l and get thus two or more independent subproblems. We can now (recursively) investigate these subsets (e.g. compute the number of their triangulations) and combine the results (e.g. multiply the numbers obtained). This gives us (e.g. the number of) all triangulations of S containing the t-path $p_l(S)$. We repeat this process for all possible t-paths of S and l, putting together the obtained information (e.g. we sum up the numbers obtained for every t-path to get the number of different triangulations of S).

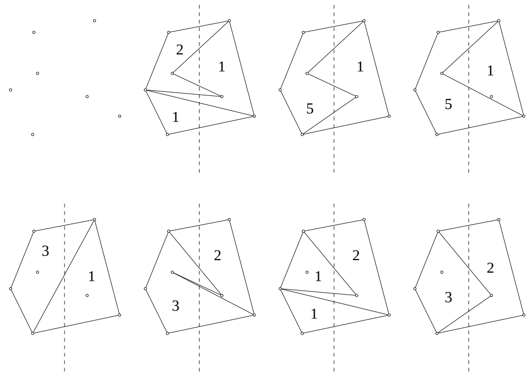

Figure 9: The 7 t-paths of 7 points with 29 triangulations. Every face contains its number of triangulations.

From Theorem 3 it follows that two different t-paths of S and l can never be part of the same triangulation of S and thus we cannot consider (e.g. count) triangulations twice in this way. Since by Theorem 1 any triangulation of S contains exactly one t-path for l we also can not miss a triangulation. Together this shows that we indeed investigate all triangulations of S exactly once (e.g. count the exact number of triangulations of S). Refer to Figure 9 for an example with 7 different t-paths for fixed l.

Clearly the choice of the splitting line l does not influence the result of the computation, but affects the efficiency of the algorithm. Experiments showed that during the recursive steps it is most efficient to split the points into two equal sized sets alternately by horizontal and vertical lines, respectively. If in this way we obtain a line with no non-trivial t-path we choose an alternative strategy according to Lemma 2. For a subset of m points we simply check all $m-1$ positions between the points (for fixed direction) and take an arbitrary (e.g. the most central) line with a non-trivial t-path. It turned out that this is necessary for less than 1 % of all t-path computations.

We also have to show that the recursive steps terminate. These can be seen from the fact that any non-trivial t-path reduces the size of the subpolygons by at least one triangle induced by S. Considering all nested recursive steps we obtain at most a triangulation of S and thus we can have only a linear number of nested recursive calls.

The presented divide & conquer approach seems to be applicable whenever parts of a triangulation separated by a t-path can be treated independently. This is e.g. the case for counting the number of triangulations or computing the minimum weight triangulation. We expect further applications to come up in the future.

3.1 Computing t-paths

Let us now briefly explain how to compute all t-paths of a given point set S, polygon P and line l. For each point $v \in S$ we first sort all remaining points in clockwise order around v. Moreover for every point $v' \neq v$ we store the index of v in the sorted order around v' in v, such that for an edge (v, v') we can find the neighboring edges emanating from v' in constant time. The preprocessing can be done in $O(n^2)$ time and space, see [10].

To simplify the presentation we assume a vertical l line. We start with the top most edge of P intersected by l, e.g. an edge of the convex hull of S. Separately starting from both endpoints we then construct all t-paths recursively. Assume now we already have a part of a valid t-path and let e and v be its last edge and point so far. If e is the second edge of P intersected by l we have found a complete t-path. Else we consider all edges $e' = (v, v')$ to continue in order of increasing angle w.r.t. the 'incoming' edge e. As long as v and v' lie on opposite sides of l we can extend the t-path with e' and v' recursively. If v and v' are on the same side of l then neither e' nor any other edge with greater angle is a valid continuation of the actual t-path and we therefore return from the recursion, removing edge e. The conceptual sketch of the algorithm is summarized below.

```
NEXT-EDGE(e,v)
IF e ∈ P THEN
  t-path is complete, split P
ELSE
  IF v is left of l THEN
    REPEAT
      Let e' = (v, v') be the next edge w.r.t. e
      emanating from v in cw-order.
      IF v' is right of l THEN
        NEXT-EDGE(e',v')
    UNTIL v' is left of l
  ELSE
    REPEAT
      Let e' = (v, v') be the next edge w.r.t. e
      emanating from v in ccw-order.
      IF v' is left of l THEN
        NEXT-EDGE(e',v')
    UNTIL v' is right of l
```

The correctness of the approach follows from Definition 2 together with Observation 1. Any step within the recursion can be done in constant time. Since any t-path has at most linear length and most considered edges are multiple used we expect the number of steps to be low polynomial in the number of t-paths. In fact experiments show that for all investigated point sets, either randomly chosen or 'badly' constructed, on average only a constant number of steps has to be carried out per t-path. On the other hand we might also get dead ends, i.e. points where it is impossible to continue a t-path. Thus we might have to backtrack several recursive steps without getting a t-path. This makes it hard to give a reasonable worst-case bound.

4 Applications

4.1 Counting and Enumerating Triangulations

The number $t(S)$ of different triangulations of a set S of n points in the plane is exponentially growing. The best known lower and upper bounds are (roughly) $\Omega(2^{3n})$ [13] and $O(2^{8n})$ [8], respectively. For the special case that S is in convex position we get the well known Catalan numbers. Results for special point sets, e.g. for near-convex polygons, have recently been shown, see [12, 15–17].

Figure 10: 25 points with 10 095 432 673 807 triangulations. The points were randomly generated in the unit square.

Numerical investigations of arbitrary point sets are helpful to investigate the structure of 'extreme' examples. Unfortunately, previous approaches like reverse search are capable of handling at most 12–15 points [2]. From the last section it follows that we can apply the t-path divide & conquer to this problem. With $P_l(S)$ we denote the set of all t-paths of S according to l and for any t-path $p_l(S) \in P_l(S)$ let $R(p_l(S))$ be the set of subregions into which S is split by $p_l(S)$. We get $t(S) = \sum_{p_l(S)\in P_l(S)} \prod_{R_i \in R(p_l(S))} t(R_i)$, representing the different parts of the algorithm.

In Section 2.2 we have shown that for a given planar point set S we might have to consider an exponential number of different t-paths. Nevertheless the application of the developed divide & conquer approach to the problem of counting triangulations of S turned out to be much more efficient than the best previously known methods. For example we are able to count the number of triangulations of up to 25 points, see Figure 10.

Figure 11 gives the cost for counting, defined as the running time (in ms) per triangulation. The time was observed on a Pentium© PC with 266 MHz running under Linux. It can be seen that

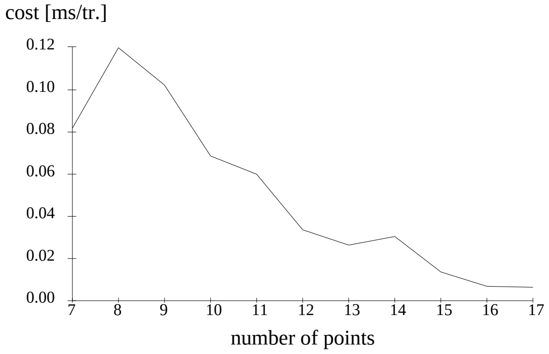

Figure 11: The cost (time in ms per triangulation) of counting triangulations is decreasing.

the algorithm runs in time sublinear in the number of triangulations counted, i.e. with decreasing cost for increasing problem size.

The best previously known methods to count triangulations are based on reverse search and have a running time of $\Theta(n \cdot t(S))$ [2], i.e. the cost of this approach increases linear with the input size. Thus our method constitutes a clear improvement. The main reason is that instead of enumerating triangulations we can multiply the numbers of triangulations for independent subsets and continue this approach recursively.

The interested reader may run the program for sets of up to 20 points simply via e-mail. For more information see http://www.cis.tu-graz.ac.at/igi/oaich/infotricount.html .

Note that instead of counting triangulations we might also enumerate them. To this end it is sufficient for any fixed t-path to combine all different triangulations of all obtained subpolygons. This can simply be done in the recursion since the parts are independent. For enumeration the running time will be proportional to the number of triangulations, which is still by a linear factor faster than previous methods.

4.2 Random Triangulations

Let us now consider the problem of generating random triangulations. To be 'random' here we require that any triangulation of the given point set S appears with the same probability $1/t(S)$. Random triangulations are on one hand used to test programs based on triangulations and on the other hand are helpful to investigate the structure of 'typical' triangulations.

The basic idea of our approach is to utilize the results obtained from counting to quickly produce random triangulations afterwards. During the counting process we therefore store for every generated t-path of S the number of triangulations considered so far, including the actual t-path.

To obtain a random triangulation we choose

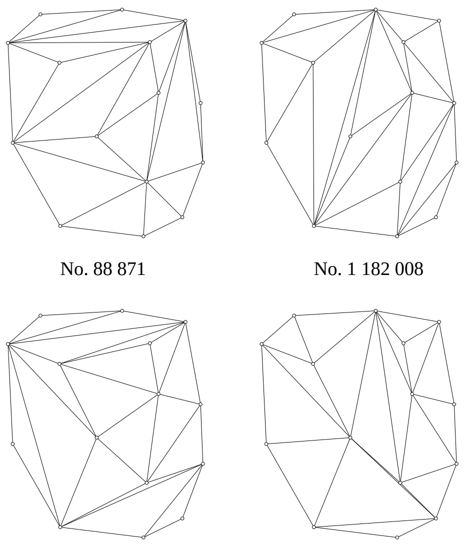

Figure 12: Four (out of 1 558 796) randomly chosen triangulations of 15 points.

a random number $r \in [1, \ldots, t(S)]$ and by binary search we find out the t-path $\tilde{p}_l(S)$ which generates the r^{th} triangulation. We then fix $\tilde{p}_l(S)$ and repeat the counting process, starting with the number of triangulations given by the previous t-path. Recursively we get for every polygonal part induced by $\tilde{p}_l(S)$ the number of triangulations. Step by step this determines how to triangulate every subpart, until we find triangulation No. r.

Figure 12 shows four random triangulations of a set of 15 points, together with their sequence number.

The running time of this approach is clearly dominated by the required preprocessing. As mentioned in the last section this limits the number of points to about 25, i.e. the number of paths and thus the amount of information to be stored is reasonable (e.g. 1047 t-path for the example of Fig. 12). Computing the sub-triangulations for fixed $\tilde{p}_l(S)$ is expected to be fast because of the reduction of the problem size (generating the four examples of Fig. 12 was done in less than 1 sec.).

Note that with this method we can also compute constrained random triangulations. This means we can either force edges induced by S to be in any triangulation considered, or exclude them. As mentioned in the next section, the t-path and thus our divide & conquer approach is still well defined as long as at least one valid triangulation of S exists.

4.3 Optimizing Triangulations

In recent years, several subgraphs of optimal triangulations have been identified. One of them arises from a class of empty neighborhood graphs defined by Kirkpatrick and Radke [19], and is called the *β-skeleton*. Keil [18] has shown that for specific values of β, the β-skeleton is a subgraph of the minimum weight triangulation (MWT), see [1] for an overview on recent improvements of this result.

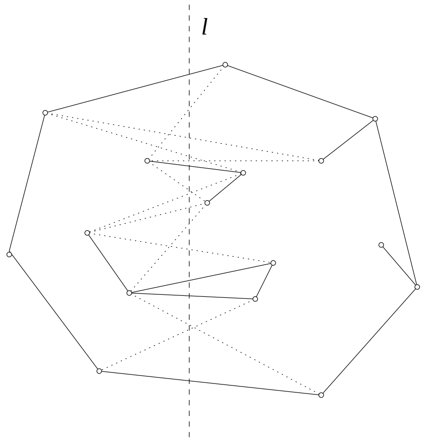

Figure 13: Skeleton subgraphs of a point set (solid lines) and possible t-paths connecting them (dotted lines).

Another large subgraph of local optimal triangulations is the so-called *LMT-skeleton* of S, which can be computed by a simple and cute method, proposed in Belleville et al. [4] and in Dickerson, et al. [9], see also Beirouti and Snoeyink [3]. In fact the LMT-skeleton of S tends to be a connected graph even for large point sets and thus by dynamic programming [14, 20] it can be completed to an optimal triangulation in polynomial time. But in general the resulting subgraph will consist of k disconnected parts. In [5] it has been shown that in the average case k may be linear in $|S|$.

The approach proposed e.g. in Cheng et al. [6] tries all possibilities to add $k-1$ edges to obtain a connected graph. Completing each of these graphs to a triangulation by optimally triangulating its faces with the above mentioned polynomial time methods, and selecting a triangulation with minimum weight, gives the optimal triangulation. This approach can be implemented to run in $O(n^{k+2})$ time.

Instead of a simple exhaustive investigation of all possibilities to connect the separated subgraphs we show how to use the t-path for this task. For the MWT we can obviously optimize the triangulation containing a given t-path by optimizing the subparts independently, i.e. if a t-path fails to connect all subgraphs we can continue recursively.

It is important to observe that a t-path is still well defined if we consider a restricted set E of edges, $E \subset S \times S$, instead of all edges induced by S, provided E includes a triangulation of S. This follows directly from Theorem 1. Thus we can exclude edges from being in a triangulation (cf. the exclusion region for any MWT by Das and Joseph [7]), or force an edge e by removing all edges from E intersected by e. Therefore, given a subgraph G obtained by any of the above mentioned methods, we try to find a line l which intersects as many of the disconnected parts of G as possible. By condition 3) of Definition 1 any t-path $p_l(S)$ connects all intersected subgraphs and considering all t-paths reflects all possibilities to do that. Again we can continue on remaining disconnected subgraphs recursively. Together with the fact that we expect the number of resulting t-paths to be significantly smaller than $\Theta(2^{k-1})$, which is the number of steps for exhaustive search, this gives rise to hope that the t-path will speed up the computation of MWTs for highly disconnected skeletons.

Let us finally mention that by excluding edges we might apply the t-path also to point sets with collinearities. We simply have to exclude all edges which contain collinear points in their interior. We thus e.g. plan to investigate the number of triangulations of small regular grids of points, which is a well known (hard) problem.

5 Conclusion and Open Problems

In this paper we have introduced a new geometric data structure, the triangulation path. Despite its simple definition it reveals some nice properties like existence and unambiguity for any triangulation. We then showed how to use such paths for a divide & conquer like approach for some triangulation problems. Applications include counting and enumerating triangulations, generating random triangulations, and computing optimal triangulations from their skeletons.

Open questions mainly arise from the algorithmic point of view. What is the expected number of paths to be considered in the divide & conquer method? What are the expected probabilistic performance of the mentioned applications? Is it possible to show reasonable worst case bounds?

Acknowledgments

We would like to thank Franz Aurenhammer and Günter Rote for several discussions and helpful comments on the presented topic and Bernhard Erkinger for implementing and testing the mentioned algorithms [11]. We also thank an anonymous referee for suggestions to improve the presentation of the paper. Thanks also goes to Harry Fuchs for recovering the files of this paper after a hacker attack.

References

[1] F. Aurenhammer, Y.-F. Xu, *Optimal Triangulations*, Encyclopedia of Optimization, Kluwer Acad. Publ., to appear.

[2] D. Avis, K. Fukuda, *Reverse Search for Enumeration*, Discrete Appl. Math. 65, 1996, 21-46.

[3] R. Beirouti, J. Snoeyink, *Implementations of the LMT heuristic for minimum weight triangulation*, Proc. 14th Ann. ACM Symp. on Computational Geometry, 1998, 96-105.

[4] P. Belleville, M. Keil, M. McAllister, J. Snoeyink, *On computing edges that are in all minimum-weight triangulations*, Proc. 12th Ann. ACM Symp. on Computational Geometry, 1996, V7-V8.

[5] P. Bose, L. Devroye, W. Evans, *Diamonds are not a minimum weight triangulation's best friend*, Proc. 8th Canadian Conf. on Computational Geometry, 1996

[6] S.-W. Cheng, M.J. Golin, J.C.F. Tsang, *Expected-case analysis of β-skeletons with applications to the construction of minimum-weight triangulations*, Proc. 7th Canadian Conf. on Computational Geometry, 1995, 279-283

[7] G. Das, G. Joseph, *Which triangulations approximate the complete graph?*, Optimal Algorithms, Lecture Notes in Computer Science 401, Springer, 1989, 168-192.

[8] M. Denny, C. Sohler, *Encoding a triangulation as a permutation of its point set*, Proc. 9th Canadian Conf. on Computational Geometry, 1997, 39-43

[9] M.T. Dickerson, J.M. Keil, M.H. Montague, *A large subgraph of the minimum weight triangulation*, Discrete & Computational Geometry 18, 1997, 289-304.

[10] H. Edelsbrunner, *Algorithms in Combinatorial Geometry*, Springer-Verlag, 1987.

[11] B. Erkinger, *Struktureigenschaften von Triangulierungen*, Master Thesis, TU-Graz, 1998.

[12] P. Flajolet, M. Noy, *Analytic Combinatorics of Non-crossing Configurations*, INRIA Report 3196, 1997, to appear in Discrete Mathematic.

[13] A. Garcia, M. Noy, J. Tejel, *Lower Bounds for the number of crossing-free subgraphs of K_n*, Proc. 7th Canadian Conf. on Computational Geometry, 1995, 97-102.

[14] P.D. Gilbert, *New results in planar triangulation*, M.S. thesis, Coordinated Science Laboratory, University of Illinois, Urbana, 1979.

[15] F. Hurtado, M. Noy, *Counting triangulations of almost-convex polygons*, Ars Combinatoria 45, 1997, 169-179.

[16] F. Hurtado, M. Noy, *Ears of Triangulations and Catalan Numbers*, Discrete Mathematics 149, 1996, 319-324.

[17] F. Hurtado, M. Noy, *Triangulations, visibility graph and reflex vertices of a simple polygon*, Computational Geometry: Theory and Applications 6, 1996, 355-369.

[18] M. Keil, *Computing a subgraph of the minimum weight triangulation*, Computational Geometry: Theory and Applications 4, 1994, 13-26.

[19] D.G. Kirkpatrick, J.D. Radke, *A framework for computational morphology*, G.T.Toussaint (ed.), Computational Geometry, Elsevier, Amsterdam, 1985, 217-248.

[20] G.T. Klincsek, *Minimal triangulations of polygonal domains*, Annals of Discrete Mathematics 9, 1980, 127-128.