Combinatories and Triangulations*

Tomas Hlavaty** and Václav Skala***

University of West Bohemia,
Department of Computer Science and Engineering,
Univerzitni 8, 306 14 Plzen,
Czech Republic
{thlavaty,skala}@kiv.zcu.cz

Abstract. The problem searching for an optimal triangulation with required properties (in a plane) is solved in this paper. Existing approaches are shortly introduced here and, specially, this paper is dedicated to the brute force methods. Several new brute force methods that solve the problem from different points of view are described here. Although they have NP time complexity, we accelerate the time needed for computation maximally to get results of as large sets of points as possible. Note that our goal is to design the method that can be used for arbitrary criterion without another prerequisite. Therefore, it can serve as a generator of optimal triangulations. For example, those results can be used in verification of developed heuristic methods or in other problems where accurate results are needed and no methods for required criterion have been developed yet.

1 Introduction

Assume that N points (in a plane) are given. Construct a triangulation on this set of points that is optimal from the point of view of required properties.

The mentioned problem above try to solve many applications and criterions that describe the properties of triangulations can have many forms (e.g., a triangulation that minimizes sum of edge weights or that maximizes minimal angle in triangles, etc.). This paper is just dedicated to this issue and several algorithms that solve this problem are described here.

Next two chapters are a short introduction about triangulations and approaches of triangulation generating. The first chapter is dedicated to the definition of triangulation and to the general properties of triangulations. The second one contains an overview of existing approaches that can solve this issue. The remainder chapters are dedicated to methods based on the brute-force approach and they describe several algorithms. The paper is finished by a comparison of the individual methods mutually and by a conclusion. Note that the comparison is based on implementation of methods for a given problem, exactly, they search for the MWT (i.e., Minimum Weight Triangulation) [5], [7], [10].

* This work is supported by the Ministry of Education of the Czech Republic projects:
** FRVS 1342/2004/G1, *** MSM 235200005.

A. Laganà et al. (Eds.): ICCSA 2004, LNCS 3045, pp. 81–89, 2004.

2 Triangulation

First of all, we should define the term *triangulation*. However, no exact definition exists. The triangulation can be seen from several views as it is shown in following two definitions (we only will think about triangulation of points in a plane here):

Definition 1. Let us assume that we have a set of different points in a plane $S = \{p_i\}$, $p_i \in E^2$, $i = 1, \ldots, N$. Then a set of so called edges represents a triangulation $T(S) = \{e_i\}$ if the following conditions are valid:

1. Each edge e_i in the triangulation includes just two points from the set S and these points are end points of the edge (the edge is an abscissa that connects two given end points).
2. Two arbitrary edges from the triangulation do not cross mutually.
3. It is impossible to insert another edge into the triangulation and to keep the previous conditions valid simultaneously.

Definition 2. Let us assume that we have a set of different points in a plane $S = \{p_i\}$, $p_i \in E^2$, $i = 1, \ldots, N$. Then a set of so called empty triangles represents a triangulation $T(S) = \{t_i\}$ if the following conditions are valid:

1. Each triangle in the triangulation includes just three points from the set S and these points are vertices of the triangle (another point inside the triangle cannot be included - this triangle is called the *empty triangle*).
2. Intersection of two arbitrary empty triangles from the triangulation can be a vertex or an edge of the triangle maximally.
3. It is impossible to insert another empty triangle into the triangulation and to keep the previous conditions valid simultaneously.

In the first moment, the definitions seem to be similar. It is valid because they only look on the triangulation from two different views. In the first definition the triangulation is represented as a set of edges and in the second one the triangulation is represented as a set of triangles. An example of a triangulation is shown on the Fig. 1. Note that many other definitions can be made up.

The boundary of the triangulation is the convex hull of a set of points S (see the Fig. 1). Note that this is always valid for all triangulations constructed according to the mentioned definitions and we can use this fact to determine those edges automatically.

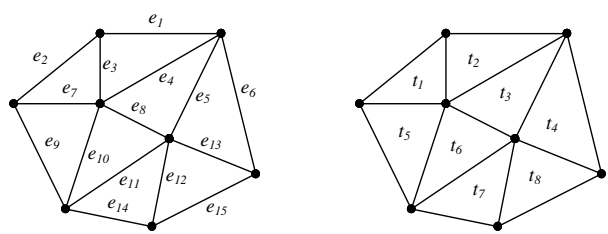

Fig. 1. Triangulation – a set of edges, a set of triangles

We mentioned that edges of the convex hull are always in the triangulation. Let us denote this kind of the edges as the common edges. However, the edges of convex hull are not alone in this group of edges. It can be expanded by extra edges according to the Definition 1. Then a general definition of the common edges can be following:

Definition 3. Let us assume that we have a set of different points in a plane $S = \{p_i\}$, $p_i \in E^2$, $i = 1, \ldots, N$ and complete undirected graph on this set of point $G = \{e_k : e_k = \{p_i, p_j\}, i \neq j, i, j = 1, \ldots, N\}$. The edges e_k from the graph G which cross no other edge are always in the arbitrary triangulation that can be constructed in the input set of points S and these edges are denoted as common edges.

Some examples of common edges for several sets of points are shown on the Fig. 2 (note that edges of convex hull also fulfill mentioned definition).

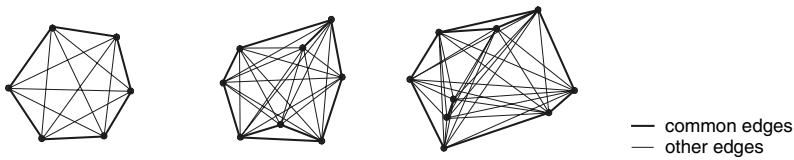

Fig. 2. Examples of common edges

The next important property is the theorem about a number of edges and triangles in an arbitrary triangulation that is possible to construct on a given set of points S.

Theorem 1. Let us assume that we have a set of N points $S = \{p_i\}$, $i = 1, \ldots, N$. If the number of points in the convex hull is N_{CH} then:

$$\begin{aligned} N_E &= 3 \cdot (N-1) - N_{CH} \\ N_T &= 2 \cdot (N-1) - N_{CH} \end{aligned}, \tag{1}$$

where N_E is the number of edges and N_T is the number of empty triangles in the triangulation.

Last property, which we can use, follows from the definition of the triangulation. No edges in the triangulation can cross mutually. Possibly, we can say about triangles that no two triangles in the triangulation can overlap more than in an edge.
This fact minimizes the number of edges that can be inserted into triangulation from the set of all possible edges. If an edge is inserted into a triangulation, we can be sure that all edges that cross this edge cannot be in the triangulation (this is also valid for triangles).

These three properties are valid for arbitrary sets of points. If we knew more about desired triangulations, we could find any extra properties (see [2], [5]). However, our goal is to design an algorithm which can be use for all kinds of the triangulations and which can find the result for all arbitrary criterions of the triangulation. Therefore, we will not think about this alternative.

3 Introduction about Triangulation Generating Methods

Generally, several approaches that solve the issue of searching for triangulations with a given properties exist. The ideal approach is based on usage algorithms with polynomial time complexity. However, those algorithms are only known for some properties of triangulation (e.g., Delaunay triangulation [1], [7]). In remaining cases a brute force algorithm has to be used. The brute force term means that all possible triangulations are generated, evaluated, and then the best one is selected. This approach is general and triangulations with arbitrary properties can be found. However, it also has a disadvantage. The algorithms generating all triangulations generally do not have polynomial time complexity (the NP problem [4], [6]) and, therefore, they only can find solutions on small sets of points. For all that, this paper is just dedicated to this approach and several algorithms are proposed in the following chapters. We will use knowledge from combinatorics [3], [8], [9] (combination generating and triangulation generating are similar problems) and knowledge about triangulations (see previous chapter) to design a fast, accurate and robust algorithm.
Note that one more approach exists. It is based on heuristic methods and can find some solutions for large sets of points. However, the triangulation found by this approach has not to be optimal. We only can be sure that it is an approximation of the exact solution with an error. This approach can be considered as a compromise between the polynomial time complexity and the exact solution.

3.1 Generator of Combination

From the equation (1) we know that all triangulations that can be constructed on a set of points still have the same number of edges NE. This fact and a generator of combinations together can be used to design an algorithm generating all triangulations as it is described in the following text.

If we made a unification of the edges from all the triangulations, which can be constructed, we would obtain a complete undirected graph of the set of points. Note that the maximal number of the edges in this graph is equal:

$$n = \binom{N}{2} = \frac{N \cdot (N-1)}{2}, \tag{2}$$

where N is the number of points.

Let us assign an index (from value *1* to n) to each edge in that complete undirected graph. Suppose also that a generator of combinations generates all possible sequences of N_E numbers where individual numbers are different mutually and they are from the range *1* to n. Then each combination can represent a triangulation and the number of those combinations is equal to the binomial coefficient of n and k that is defined as:

$$\binom{n}{k} = \frac{n!}{(n-k)!\cdot k!}, \; k = N_E - N_{CE}, \tag{3}$$

where n is the number of edges of the complete undirected graph (see the equation 2), N_E is the number of edges in triangulation (see the equation 1) and N_{CE} is the number of common edges.

This combinatorial number proofs that we can expect non polynomial time complexity. On the other hand, this is the worst case. Many combinations do not represent a triangulation because the condition of the crossing edges is not guaranteed. There is a question how to select the combinations representing the triangulations effectively. Two methods are possible:

A. All combinations are generated by very fast algorithm [3], [8], [9], and then the individual combinations are tested if they represent triangulations.
B. The algorithm is designed that it only generates the combinations of edges representing triangulations.

Theoretically, it is very hard to decide which of the methods is better. The first method uses a fast generator of the combinations. However, all combinations have to be generated and tested if they represent a triangulation. The second method only generates the combinations representing triangulations. However, the generator is slower because a test that excludes the unsuitable combinations is included in the generator. A threshold of the decision if it is better to use the A or B method affects many factors (the speed of generating combinations, the speed testing if a combination represents a triangulation, how many percents of combinations represent triangulations, etc.). Practically, it is more simple and infallible to implement the given algorithms and to compare them mutually as in our case. Note that a comparison of both methods is shown later in the chapter containing results.

3.2 Edge Removing Method

Complete undirected graph is remarked in the previous method. If we looked at the complete undirected graph again, we could find out that the unification of all edges, which are in the individual triangulations, also represents this graph. This fact is used in this method.

The start point of the algorithm is the complete undirected graph. When we will select and mark an edge in the graph as the edge that has to be in triangulation, we can remove all the edges that the given edge crosses. So we will obtain a new graph without any edges from the complete graph and with an edge that is marked as the edge of the triangulation. This procedure can be repeated until we obtain a graph that only includes edges representing a triangulation. Of course, we need to find all triangulations. The generating of the other triangulations is hidden in the mechanism of edge selecting that decides if individual edges have to be in the triangulation. This mechanism has to provide that no triangulation will be omitted and that any triangulations also will not be generated twice or more times.

The result structure that fulfils the requirements is a binary tree. The root of the tree represents the complete undirected graph and the leaves of the tree can be divided into two groups. In the first group, there are the leaves representing the triangulations according to the definition and, in the second one, there are the leaves that include non crossing edges, but their number is not adequate (see the equation 1).

Like in the previous algorithm, we have to assign the unique index to each edge. Then we can try to remove or to keep on the individual edges in the graph according to the index of the edges step by step. Each decision represents one level of the tree,

therefore, the maximal number of the levels is equal to the number of the edge in the complete graph (see the equation 2). However, this value is less in practice because the general properties of triangulations can be used in the implementation (see the chapter about the triangulation). An example of that tree with a binary vector representation is shown on the Fig. 3 (each bit represents one edge with a given index, the value '1' means that the edge is in the graph).

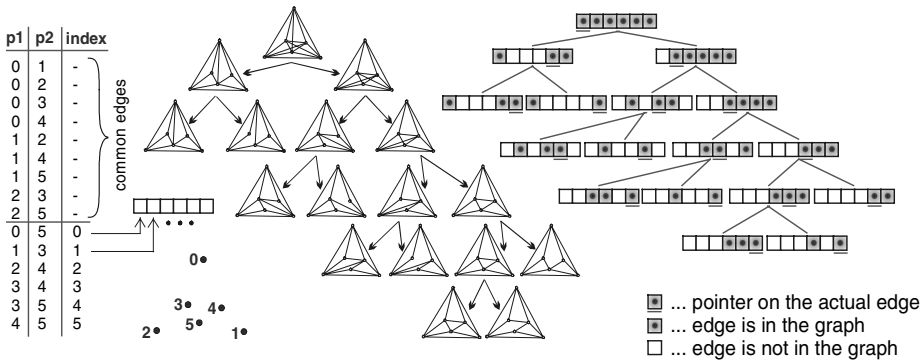

Fig. 3. An example of the edge removing method

3.3 Edge Inserting Method

This method is very similar to previous one. The only main difference is that the root node of the tree represents the complete graph but the so called empty graph (it means the graph with no edges). Otherwise, the algorithm is the same. A good question is if this method towards the previous method is faster. Theoretically, it is very hard to decide. It is affected many factors and, therefore, the implementation on the given kind of the problem is the infallible way. An example of the tree with the representation by a binary vector (like in the previous method) is shown on Fig. 4.

It perhaps seems that a representation by the binary vector is not possible. It is not true. When we select an arbitrary node in the graph, we can separate the binary vector into two parts (the left and right part) by the pointer on an actual edge. The bits of the left part represent the edges which have been in process and their status only indicates that the edges are or are not in the triangulation. The bits in the right part of the binary vector (inclusive of the actual edge) represent the edges which have not been in process yet and their status can say if the given edge still can be inserted into the triangulation or if it is not possible. Now, it is sure that the binary representation is sufficient and suitable in this case.

3.4 Triangle Inserting Method

In this last method that is introduced here we look on a triangulation like on a set of the triangles. Of course, we could look at the triangulation from the same view in previous methods and we could work with the empty triangles instead of the edges. However, this approach would be worse and the final algorithm would be slower.

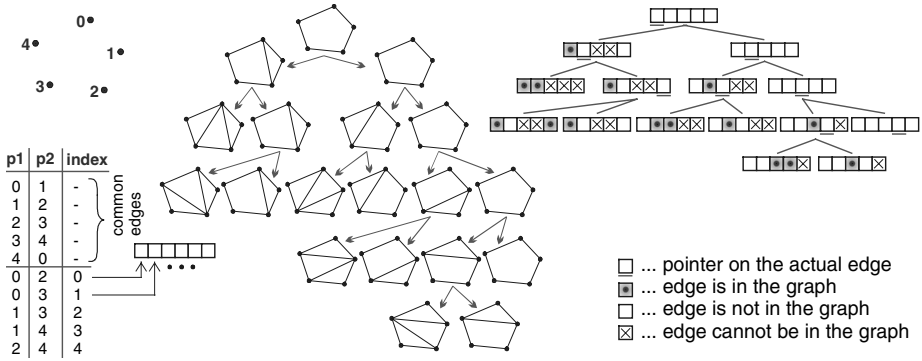

Fig. 4. An example of the edge inserting method

Let us return to our algorithm. At the beginning of this paper we said that the convex hull was in all the triangulations. We used this fact here and the convex hull is the start point of this algorithm. Exactly, the convex hull represents a polygon (the so called *boundary polygon*) surrounding a region into which triangles have to be inserted for creating a correct triangulation. The procedure of the algorithm is very simple. An edge is chosen from the boundary polygon, and then the so called empty triangle is inserted if it contains the selected edge and if it is inside the boundary polygon. The empty triangle means a triangle whose vertices are any points from the input set and which contains no other points from this set (see definition 2). By inserting the triangle, the boundary polygon will be changed and will demark the original region without the region of the inserted triangle. From this new polygon an edge is selected and another empty triangle, which contains this selected edge and which is included inside the new region, is inserted again. That procedure is repeated until a correct triangulation is created (the boundary polygon just represents an empty triangle).

Now we obtain one triangulation, however, we need to generate all triangulations. It is possible to generate them when we ensure inserting all combinations of the empty triangles for the given selected edge. We will get a tree data structure where the root is the node including the edge of the convex hull and where the leaves of the tree represent the triangulations. Each intermediate node has as many branches as many empty triangles can be inserted for the selected edge of the given boundary polygon. An example of this tree is shown on the Fig. 5.

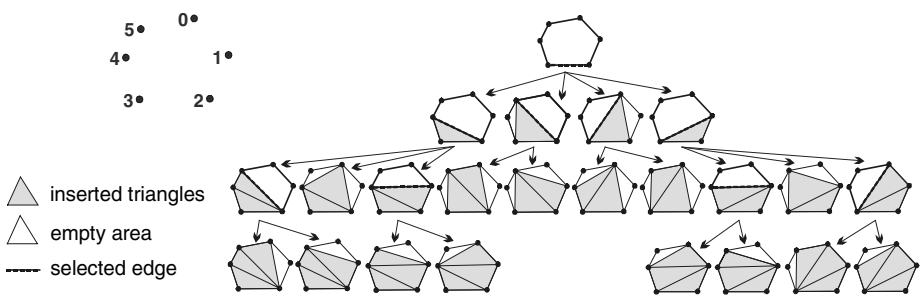

Fig. 5. An example of the triangle inserting method

4 Results

We described a few algorithms that generate all triangulations. In this chapter we compared them mutually. The described algorithms were implemented for the MWT (*Minimum Weight Triangulation*) [5], [7], [10] where weights of edges are represented by Euclidian distances between the end points of edges. For this criterion, an algorithm with a polynomial time complexity still has not been found, therefore, it is an ideal situation for testing mentioned algorithms.

Our goal is to find the triangulation that has a minimal sum of weights of edges. A structure of programs with the individual algorithms is similar and simple. When a triangulation is found, it is evaluated and tested (the main task of the test is to remember the triangulation with the best evaluation). When all possible triangulations are found, we can be sure that we have obtained the best one. Note that an advantage of this approach is in a small memory requirement and we always find the global optimal solution. We do not need to remember all triangulations but only the best one. We tested all algorithms for randomized generated sets of points on the same computer (DELL, 450 MHz, 1 GB RAM) with OS Windows 2000. The resultant graph that characterizes the time dependence of the calculation on the number of points is shown on the Fig. 6.

The values in the graph were calculated as an average of times that had been measured for the sets with the same number of points. Consequently, the values in the graph are only expected times that were measured for the given kind of data (the uniform distribution of points in a plane) on the given computer. For all that, we can obtain some basic information about the individual algorithms and we can determine which method is faster or slower. We can obtain an estimation of time for evaluation of a bigger set of points, etc.

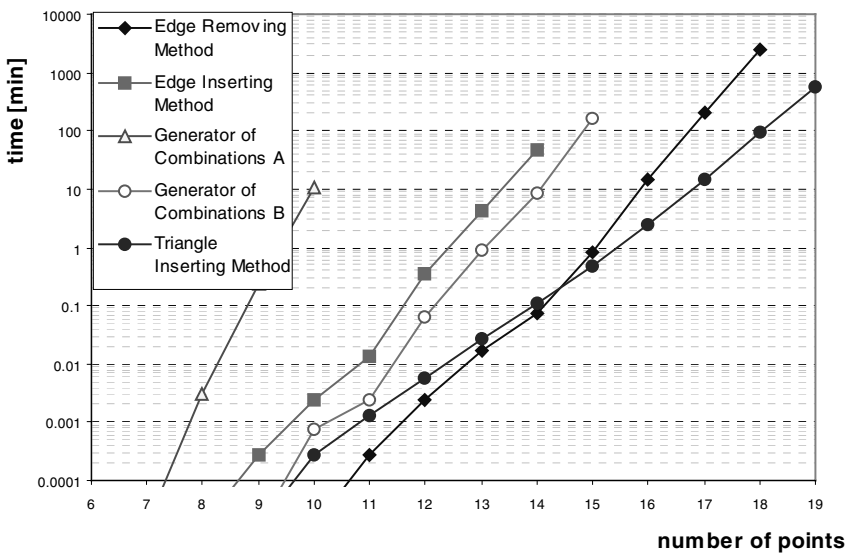

Fig. 6. The graph that shows the expected time needed to finding for the MWT by the designed methods (a dependence on the number of points)

We can also estimate time complexity of the algorithms for another criterion on the triangulation. The test searching for the MWT has *O(N)* time complexity in the algorithm (the sum of edge weights has to be calculated for the found triangulation). When we select the criterion that has the same time complexity for criterion evaluation in the algorithm, we can use these results to estimate of needed time for calculation.

5 Conclusion

The main goal of this work was to generate optimal triangulations for a required criterion. It is expected that such generated triangulations will be used for verification of new algorithms and for effective triangulation generating.

This paper presents an overview of new approaches. Several methods searching for global optimal triangulations with required properties were developed, implemented and tested. The comparison of developed algorithms generating all possible triangulations was also made. By comparing the individual curves in the graph (see the Fig. 6), we can see properties of developed algorithms. Generally, the complexity of the triangular mesh generator is not polynomial and, therefore, a selection of an unsuitable data structure or algorithm influences extensively the time that is needed for the computation.

Finally, note that although the algorithms are designed for a triangulation generator, the introduced algorithms can also be used to solve similar problems (e.g., combination generating, etc.).

References

1. Aurenhammer, F.: Voronoi Diagrams - A Survey of a Fundamental Geometric Data Structure. ACM Computing Surveys 23(3): 345–405, 1991.
2. Drysdale, R., L., S., McElfresh, S., Snoeyink, J., S.: An improved diamond property for minimum weight triangulation. 1998.
3. Ehrlich, G.: Loopless algorithms for generating permutations, combinations, and other combinatorial configurations. Journal of the ACM, vol. 20, Issue 3, pp. 500–513, 1973.
4. Garey, M., R., Johnson, D., S.: Computers and Intractability: A Guide to the theory of NP-completeness. W. H. Freeman, San Francisco, 1979.
5. Jansson, J.: Planar Minimum Weight Triangulations, Master's Thesis, Department of Computer Science, Lund University, Sweden, 1995.
6. Kucera, L.: Combinatorial Algorithms, ISBN 0-85274-298-3, SNTL, Publisher of Technical Literature, 1989.
7. Preparate, F. P., Shamos, M. I.: Computational Geometry - an Introduction, Springer-Verlag, New York, 1985.
8. Takaoka, T.: O(1) time algorithms for combinatorial generation by tree traversal. Computer Jurnal, vol. 42, no. 5, pp. 400–408, 1999.
9. Xiang, L., Ushijima, K.: On O(1) Time Algorithms for Combinatorial Generation. The Computer Journal, vol. 44, no. 4, pp. 292–302, 2001.
10. Yang, B., T., Xu, Y., F., You, Z., Y.: A chain decomposition algorithm for the proof of a property on minimum weight triangulations. 1994.