VOL.3 NO.2September 2014 - ISSN 2233 – 1859

Operations Research Society in Bosnia and Herzegovina

Southeast Europe Journal of Soft Computing

Available online: www.scjournal.com.ba

IUS Soft Computing Research Group

FINDING OPTIMAL TRIANGULATION BASED ON BLOCK METHOD

Sead H. Mašović[1], Muzafer H. Saračević[2]

[1]Faculty of Science and Mathematics, University of Nis, Višegradska 33, Niš, Serbia, sead.masovic@pmf.edu.rs
[2]Department of Computer science, University of Novi Pazar, Dimitrija Tucovića, Serbia, muzafers@uninp.edu.rs

Article Info

Article history:
Received June 2014
Received in revised form Nov 2014

Keywords:
Computational geometry, Computer graphic, Optimal triangulation, Block method

Abstract

In this paper we give one new proposal in finding optimal triangulation which is based on our authorial method for generating triangulation (Block method). We present two cases in calculation the triangulation weights (classical case and case based on block method). We also provide their equality and established relationship in calculation the weights for both models, with an emphasis on simplicity of calculations which occurs in the second case. The main goal of this paper is on the speed of obtaining optimal triangulation.

1. INTRODUCTION AND PRELIMINARES

Polygon triangulation is an important problem applicable in computer graphics. One of the topics in the field of triangulation is optimal triangulation. Many authors deals with the problem how to find the optimum triangulation of a convex polygon based on some criterion, eg a triangulation which minimizes the perimeters of the component triangles (value of: perimeter, sum of heights, length of the longest median and etc. [1]).

For example, the triangulation that has minimum total length of its edges is known in the literature as Minimum Weight Triangulation (MWT). Because of the difficulty of finding the exact solutions of MWT, many authors have studied heuristics that may in some cases find the solution although they cannot be proven to work in all cases. In particular, much of this research has focused on the problem of finding sets of edges that are guaranteed to belong to the minimum-weight triangulation. This is by far the most studied problem in the area of optimal triangulations [2,3,4,5,6,7].

Some other approaches in finding optimal triangulation are: *MinMax* and *MaxMin* [8], Edge-insertion paradigm [9], Subgraph Scheme [10] and etc.

In this paper we give proposal how to find optimal triangulation based on our authorial method for generating triangulation [11].

Since our method works with database the main goal of our proposal is to use recorded data (in this case internal diagonals) to obtain optimal triangulation. Discussion in finding optimal triangulation using internal diagonals is given in section 3. In section 4 we provide experimental results and comparative analysis.

Before all here we give some basic information about generating triangulation. Polygon triangulation implies decomposition of the interior of the polygon to triangles, with non-transversing internal diagonals.

The total number of all triangulations T_n of n-gon is given in the following equation:

$$T_n = \frac{1}{n-1}\binom{2n-4}{n-2} = \frac{(2n-4)!}{(n-1)!(n-2)!}; n \geq 3 \qquad (1)$$

According to our Block method [11] the number of triangulations T_n is expressed with following equation:

$$T_n = 2T_{n-1} + rest(R_n) \quad (2)$$

where T_{n-1} is number of triangulation for polygon of $n-1$ vertices, and $rest(R_n)$ are new triangulations that are not appear in the block.

2. ALGORITHM OF BLOCK METHOD

Block method is based on the recognition of polygon triangulation with smaller number of vertices (*blocks*) in a set of vertices corresponding to the polygon with larger number of vertices. More specifically, this method is based on the usage of already established triangulation stored in the form of internal diagonals.

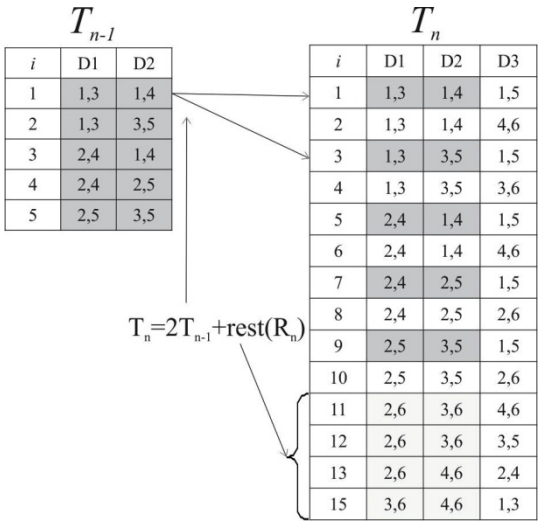
The advantage of the Block method is in usage of already obtained triangulation for the (*n*-1) polygon in obtaining triangulation for larger polygons, while for the rest is to find remaining internal diagonals. The term block represent all triangulation for a given (*n*-1) polygon which appears twice in its entirety in a set of triangulation for *n*-polygon. In this way, it avoids to re-generate the same triangulation (a technique of recursion with memoization).

The algorithm which realizes Block method can be divided into four phases:

1. In the first phase of this method, all records of the triangulation of smaller polygon are transposed
   (step 1. in Algorithm 4. in paper [11]).
2. In second phase, transposed records of (n-1) polygon are recorded in temporary database where is planned to store records of *n*-polygon triangulation (step 2).
3. In the next phase on the all records are added one additional record for another internal diagonal
   (step 3). In this step we call algorithm that is responsible for finding the remaining allowed vertices which also call additional algorithm to eliminate all prohibited (closed) vertices (definition for prohibited vertex is given in the paper [11]).
4. In the last phase of the Block method we find all remaining triangulations which have at least two internal diagonals that contain last vertex *n* (step 4.1). In this segment we call once again additional algorithm in order to find the remaining allowed vertices (step 4.2).

Since the Block method works with databases here we recall some details of storage: (1) All results for blocks (base) are recorded in the .db format; (2) Each block is recorded in the form [Tn_base].db; (3) Database contains a set of internal diagonals for each line individually, which defines a triangulation of the polygon.

Original schema of Block method is given on the following picture.



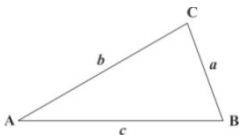
i	D1	D2
1	1,3	1,4
2	1,3	3,5
3	2,4	1,4
4	2,4	2,5
5	2,5	3,5

i	D1	D2	D3
1	1,3	1,4	1,5
2	1,3	1,4	4,6
3	1,3	3,5	1,5
4	1,3	3,5	3,6
5	2,4	1,4	1,5
6	2,4	1,4	4,6
7	2,4	2,5	1,5
8	2,4	2,5	2,6
9	2,5	3,5	1,5
10	2,5	3,5	2,6
11	2,6	3,6	4,6
12	2,6	3,6	3,5
13	2,6	4,6	2,4
15	3,6	4,6	1,3

Figure 1. Generating triangulation based on Block method

3. CALCULATION OF OPTIMAL TRIANGULATION AND APPLICATION OF BLOCK METHOD

The base of all triangulation is triangle, so here we discuss how to find perimeter of triangle (P), as one of the measure in finding optimal triangulation.



$$P_{\Delta ABC} = a + b + c$$

Before calculating perimeter for all triangles first we have to obtain distance between two vertex (length of the sides) using following equation:

$$d = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2}$$

Note: Every point in the plane is defined by the coordinates x, y.

In the procedure of finding the optimal triangulation is used total sum of perimeter of triangles, expressed in weights (W) which define one triangulation.

$$W_n = \sum P(T_n)$$

3.1 CASE WITH CLASSICAL CALCULATION OF TRIANGULATION WEIGHT

For the case of classical calculation of optimal triangulation we will take example of convex pentagon which has 5 different triangulations (based on equation (1)).

Example 1. Finding optimal triangulation via classical calculation of perimeter:

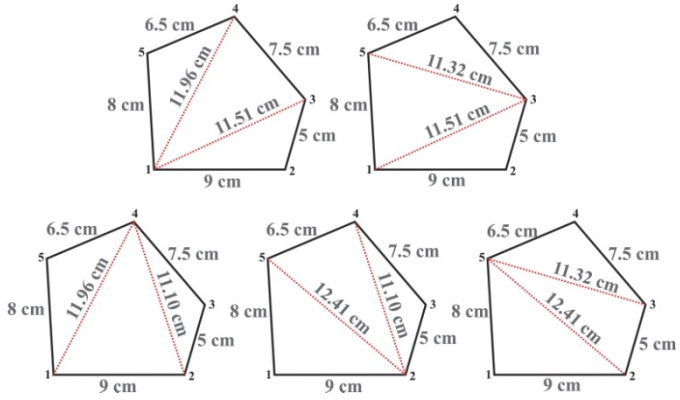


Figure 2. All combination of the pentagon triangulations

$W_1(T_5) = P_{\Delta 123}+P_{\Delta 134}+P_{\Delta 145}=$

$=(9+5+11.51)+(11.51+7.5+11.96)+(11.96+6.5+8)$

$=25.51+30.97+26.46=\mathbf{82.94}$

$W_2(T_5) = P_{\Delta 123}+P_{\Delta 135}+P_{\Delta 345}=$

$=(9+5+11.51)+(11.51+11.32+8)+(11.32+7.5+6.5)$

$=25.51+30.83+25.32=\mathbf{81.66}$

$W_3(T_5) = P_{\Delta 145}+P_{\Delta 124}+P_{\Delta 234}=$

$=(11.96+6.5+8)+(9+11.10+11.96)+(5+7.5+11.10)$

$=26.46+32.06+23.6=\mathbf{82.12}$

$W_4(T_5) = P_{\Delta 125}+P_{\Delta 245}+P_{\Delta 234}=$

$=(8+9+12.41)+(12.41+11.10+6.5)+(11.10+5+7.5)$

$=29.41+30.01+23.6=\mathbf{83.02}$

$W_5(T_5) = P_{\Delta 125}+P_{\Delta 235}+P_{\Delta 345}=$

$=(8+9+12.41)+(5+11.32+12.41)+(11.32+6.5+7.5)$

$=29.41+28.73+25.32=\mathbf{83.46}$

After calculation weights of all triangulation we find minimal (optimal) weight, as follows:

$min\ W(T_5) = min(82.94; 81.66; 82.12; 83.02; 83.46)$

$min\ W(T_5) = W_2(T_5) = \mathbf{81.66}$

In this case optimal triangulation is defined with following triangles: **Δ(1,2,3); Δ(1,3,5); Δ(3,4,5).**

3.2 CASE WITH CALCULATION OF TRIANGULATION WEIGHT BASED ON BLOCK METHOD

Before calculation here we give some properties of our proposal:

- Because the outer sides are the same for all combinations of a given polygon triangulation we exclude their calculation because it does not affect the final result;
- Considering that only the internal diagonals are different and that they determine the uniqueness of the final result, we give proposal to use only them in finding optimal triangulation.

Example 2. Proposed scenario based on block method for the given polygon T_5.

1. Reading blocks of the pentagon (T_5) from the database

i	D1	D2
1	1,3	1,4
2	1,3	2,5
3	2,4	1,4
4	2,4	2,5
5	2,5	3,5

2. Updating table with new column for storing sum of length of the internal diagonals (L_{int}) for each line (denoted with S).

$$S = \sum L_{int}$$

i	D1	D2	S
1	1,3	1,4	
2	1,3	2,5	
3	2,4	1,4	

4	2,4	2,5	
5	2,5	3,5	

3. Calculating distance between vertices and their length for each line individually, which defines a triangulation of the polygon
   - Calculating distance
   - Calculating length of the internal diagonals and their sum

4. Updating table with all data

i	D1	D2	S
1	1,3	1,4	23.47
2	1,3	2,5	**22.83**
3	2,4	1,4	23.06
4	2,4	2,5	23.51
5	2,5	3,5	23.73

5. Determining optimal triangulation based on all data from the table
   - Finding minimal value of the last column *min*(S)

After obtaining weights for all triangulation, we find minimal one (optimal), as follows:

min S(T_5)= *min*(23,47; 22,83; 23,06; 23,51; 23,73)
min (S)= **22.83**

In this case optimal triangulation is determined with following internal diagonals: **D1(1,3); D2(2,5)**.

More precisely, if we express the same triangulation via triangles **Δ(1,2,3); Δ(1,3,5); Δ(3,4,5),** we get the same optimal triangulation as in Example 1.

Conclusion is that we get the same result in both cases, only that in second case we calculate with less data which is reflected in the simplicity of the proposed method.

3.3 EQUALITY OF CASE 1 AND CASE 2

If we observe the method of calculating in the Example 1, is noted that in the calculating set of perimeters of triangles (weights) in one triangulation, every internal diagonal is always used twice.

If we eliminate length of outer diagonals (L_{out}) of given polygon we do not disturb the unity of calculating triangulation weight. Further elimination in the sum of the lengths of internal diagonals which appears twice, we obtain the same value as in Example 2 (table in step 4).

Procedure of elimination is given in following equation:

$$(W (T_n) - \sum L_{out})/2 = S \qquad (3)$$

Applying equation (3) on concrete example for the Case 1 and Case 2 is given below:

1: (82,94-36)/2=23,47

2: (81,66-36)/2=**22,83**

3: (82,12-36)/2=23,06

4: (83,02-36)/2=23,51

5: (83,46-36)/2=23,73

4. EXPERIMENTAL RESULTS AND COMPARATIVE ANALYSIS

In this section are specified testing results in finding optimal triangulation through two algorithms. Considering that our method relies on finding optimal triangulation from the set of all generated triangulation, for the purposes of comparative analysis we selected Hurtado-Noy algorithm (more about this algorithm you can see in the paper [12], and details of the implementation in [13]).

Details and experimental results for Block method (which are partially taken in table 1) can be found in paper [13].

Table 1: Execution times for Block method and time for finding optimal triangulation (in seconds)

n	***Number of triangulation***	***Block method***	***Optimal triangulation***
5	5	0.16	0
6	14	0.26	0.01
7	42	0.34	0.02
8	132	0.41	0.03
9	429	0.46	0.05
10	1,430	0.54	0.08
11	4,862	0.85	0.14
12	16,796	1.32	0.21
13	58,786	4.4	0.34
14	208,012	24.13	0.59
15	742,900	85.3	1.07

Table 2: Execution times for Hurtado-Noy algorithm and time for finding the optimal triangulation (in seconds)

n	*Number of triangulation*	*Hurtado-Noy*	*Optimal triangulation*
5	5	0.19	0
6	14	0.34	0.01
7	42	0.42	0.03
8	132	0.49	0.05
9	429	0.67	0.07
10	1,430	1.18	0.11
11	4,862	3.81	0.21
12	16,796	12.46	0.34
13	58,786	43.51	0.58
14	208,012	119.05	0.79
15	742,900	318.63	2.75

Table 3: The total execution time for two algorithms and speedup

n	*Optimal triangulation based on Block method*	*Optimal triangulation based on Hurtado-Noy*	*Speedup*
5	**0.16**	**0.19**	**1.19**
6	**0.27**	**0.35**	**1.30**
7	**0.36**	**0.45**	**1.25**
8	**0.44**	**0.54**	**1.23**
9	**0.51**	**0.74**	**1.45**
10	**0.62**	**1.29**	**2.08**
11	**0.99**	**4.02**	**4.06**
12	**1.53**	**12.8**	**8.37**
13	**4.74**	**44.09**	**9.30**
14	**24.72**	**119.84**	**4.85**
15	**86.37**	**321.38**	**3.72**

The testing is performed in NetBeans testing module "Profile Main Project / CPU Analyze Performanse" in configuration*: CPU - Intel(R) Core(TM)2Duo CPU, T7700, 2.40 GHz, L2 Cache 4 MB (On-Die,ATC,Full-Speed), RAM Memory - 2 Gb, Graphic card - NVIDIA GeForce 8600M GS.

5. CONCLUSIONS

The significance of the presented method is reflected in the fact that using the recorded values can be a very effective way to find the optimal triangulation. That means in finding allowed triangulation (combination of internal diagonals which do not intersect) directly is recorded values for vertices and their weight. This is suitable for fast drawing the optimal triangulation of irregular convex polygon.

REFERENCES

[1] Keil M.J., Vassilev T.S. (2006) "Algorithms for optimal area triangulations of a convex polygon", *Computational Geometry*, Vol. 35:173-187.

[2] Anagnostou E. and Corneil D. (1993) "Polynomial-time instances of the minimum weight triangulation problem", *Computational Geometry: Theory and Applications*, 3:247–259.

[3] Beirouti R. and Snoeyink J. (1998) "Implementations of the LMT heuristic for minimum weight triangulation" *In Proceedings of the 14th Annual ACM Symposium on Computational Geometry*, pp 96–105.

[4] Belleville P., Keil M., McAllister M. and Snoeyink J. (1996) "On computing edges that are in all minimum–weight triangulations", *In Proceedings of the 12th Annual ACM Symposium on Computational Geometry*, pp. V7–V8.

[5] Dickerson M. T., Keil J. M. and Montague M. H. (1997) "A large subgraph of the minimum weight triangulation", *Discrete and Computational Geometry*, 18:289–304.

[6] Drysdale R. L., McElfresh S. and Snoeyink J. S. (2001) "On exclusion regions for optimal triangulations", *Discrete Applied Mathematics*, 109:49–65.

[7] Heath L. S. and Pemmaraju S. V. (1994) "New results for the minimum weight triangulation problem", *Agorithmica*, 12:533–552,

[8] Keil M.J., Vassilev T.S. (2003) "An algorithm for the MaxMin area triangulation of a convex polygon", *In Proceeding of the 15th Canadian Conference on Computational Geometry*, pp. 145-149.

[9] Bern M., Edelsbrunner H., Eppstein D., Mitchell S., Tan T.S. (1993) "Edge Insertion for Optimal Triangulations", *Discrete and Computational Geometry* 10 (1): 47–65, doi:10.1007/BF02573962, MR 1215322

[10] Keil M. J. (1994) "Computing a subgraph of the minimum weight triangulation", *Computational Geometry*. pp. 413–26

[11] Stanimirovic P, Krtolica P, Saracevic M, Masovic S (2012) "Block Method for Convex Polygon Triangulation", *Romanian Journal of Information Science and Technology - ROMJIST*, Vol.15, No.4:344-354

[12] Hurtado F. and Noy M.: *Graph of Triangulations of a Convex Polygon and tree of triangulations*, Comput. Geom. 13 (1999), 179–188.

[13] Saracevic M., Stanimirovic P.S., Masovic S. and Bisevac E., Implementation of the convex polygon triangulation algorithm, Facta Universitatis, series: Mathematics and Informatics, Vol. 27, pp. 213–228, 2012.