Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2019) 27: 346 – 361

doi:10.3906/elk-1805-143

Convex polygon triangulation based on planted trivalent binary tree and ballot problem

Muzafer SARAČEVIĆ[1], **Aybeyan SELIMI**[2,*]

[1]Department of Computer Sciences, University of Novi Pazar, Novi Pazar, Serbia

[2]Department of Computer Sciences, Faculty of Informatics, International Vision University, Gostivar, Macedonia

Received: 21.05.2018 • **Accepted/Published Online:** 01.11.2018 • **Final Version:** 22.01.2019

Abstract: This paper presents a new technique of generation of convex polygon triangulation based on planted trivalent binary tree and ballot notation. The properties of the Catalan numbers were examined and their decomposition and application in developing the hierarchy and triangulation trees were analyzed. The method of storage and processing of triangulation was constructed on the basis of movements through the polygon. This method was derived from vertices and leaves of the planted trivalent binary tree. The research subject of the paper is analysis and comparison of a constructed method for solving of convex polygon triangulation problem with other methods and generating graphical representation. The application code of the algorithms was done in the Java programming language.

Key words: Computational geometry, triangulation, Catalan number, planted trivalent binary tree, ballot notation

1. Introduction

Computational geometry is the branch of computer science which deals with finding the algorithms for solving geometric problems. The polygon triangulation is a significant problem of the polygon partition that is applied in computational geometry.

In this paper, the polygon triangulation algorithms were developed based on planted trivalent binary trees (PTBT) and ballot record. These records were obtained with the selection of the particular edge of the polygon as a base and through which is entered in the tree from the starting position. Implementation of our method was realized through the following three phases:

1) Generation of a complete triangulation tree from the initial basic triangle to the given n-gon. The resulting triangulation hierarchy is based on the decomposition of the Catalan numbers (the method is described in Section 3).

2) Storage of all obtained triangulation in tree based on ballot notation (the procedure is described in Section 4).

3) Generation of individual triangles within each level of the triangulation tree based on the planted trivalent binary tree (this procedure is described in Section 5).

Two new algorithms are presented. The ballot notation for PTBT to triangulation algorithm generates convex polygon triangulation based on ballot records obtained from movements through vertices and leaves of

*Correspondence: aybeyan@vizyon.edu.mk

the planted trivalent binary tree. The inverse algorithm with movements through vertices and leaves of the planted trivalent binary tree generates the ballot record for convex polygon triangulation.

Other sections of the paper are organized in the following order. Sections 2 and 3 consist of some preliminary exposures related to polygonal triangulation, binary trees, and the Catalan numbers decomposition. Section 4 presents the problem of ballot records, lattice paths, and the correspondence of the Catalan numbers with the well-formed sequences of parentheses. Section 5 contains the method for construction of convex polygon triangulations based on ballot records and a planted trivalent binary. The comparative analysis of experimental results for ballot-lattice, ballot-trivalent and Hurtado-Noy trees are given in Section 6. Also in this section, the complexity of algorithms from the aspect of data storage amount for triangulations and number of operations for generating triangulations is presented. The final section provides concluding observations and possible directions for further research in this domain.

2. Preliminaries about trees and hierarchy of triangulations

The Catalan numbers (C_n) represent a sequence of numbers that are used as a solution to a large number of combinatorial problems. They are uncovered by seeking a general solution of the problem for the different polygon triangulation. The Catalan numbers are defined under the following formula [1]:

$$C_n = \frac{(2n)!}{(n+1)!n!} = \frac{1}{n+1}\binom{n}{k}, n \geq 0. \tag{1}$$

The tree, as an important class of graph theory [2], also has an important use in the triangulation of polygon and they are defined as an acyclic connected graph [3–5]. The trees with root are rooted trees. In rooted trees, the roots are drawn at the top and they grow downward. A rooted tree in which the vertices at each level are ordered as the first, second, third, and so on is an ordered tree. An ordered rooted tree is a binary tree if each vertex has a degree less than two (each vertex has two children at most; the left and the right child) [6]. The number of nonisomorphic binary trees with n vertices that can be drawn has a direct connection with the Catalan numbers and their relationship is given with the following theorem. The number of binary trees with n vertices is C_n.

A binary tree is planted trivalent if the degree of its root is one and that of each internal vertex is three. By deleting the root of a planted trivalent binary tree, we get an ordinary binary tree, and by attaching a new root to the existing root of a binary tree, we get a planted trivalent binary tree. Thus, there is a bijection between set of planted trivalent binary trees with n vertices and set of binary trees with $n-1$ vertices. The number of triangulation of convex n-gon is equal to the number of planted trivalent binary tree with $n-1$ leaves. Let P_n stand for convex polygon with n vertices, and $\mathcal{T}_n$ for set of all triangulations of P_n and τ_n denotes a particular triangulation from $\mathcal{T}_n$. Also, $\deg(i)$ denotes the degree of a vertex i in a triangulation. The vertex i satisfying $\deg(i) = 2$ is called an *ear*. It is well known that the number of triangulations T_n of polygon P_n is equal to $(n-2)$th Catalan number, denoted by C_{n-2}:

$$T_n = C_{n-2} = \frac{1}{n-1}\binom{2n-4}{n-2} = \frac{(2n-4)!}{(n-1)!(n-2)!}, \quad n \geq 3. \tag{2}$$

In [7], authors Hurtado and Noy suggested an algorithm for graph of triangulations of a convex polygon and tree of triangulations, where triangulations of P_n are derived from triangulations of P_{n-1}. Their procedure consists of "spli0tting" polygon diagonals (both internal and external which are polygon edges) incident to the

highest mark vertex ($n-1$ for P_{n-1}). If we perform splitting of these diagonals $\delta_{i,n-1}, i \in \{1,2,\ldots,n-2\}$ in increasing order of i, we get an ordering of triangulations of P_n.

Moreover, Hurtado and Noy defined the infinite tree of triangulations for all convex polygons where at tree level n, we have all triangulations of P_n. Every triangulation in $\mathcal{T}_n$ has a parent in $\mathcal{T}_{n-1}$ and a specific number of exactly defined descendants in $\mathcal{T}_{n+1}$.

Hurtardo and Noy proposed an algorithm to generate the triangulations of P_n based on the triangulations of P_{n-1}. Moreover, they defined the tree of triangulation where all triangulations of P_n, i.e. the triangulations from $\mathcal{T}_n$, are arranged at the level n of this tree. Each triangulation at the level n has a "father" in $\mathcal{T}_{n-1}$ and two or more "sons" in $\mathcal{T}_{n+1}$. The sons of the same father are "brothers". There is an ordering among the children of a triangulation, and consequently among all triangulations. Implementation of this algorithm in three programming languages (Java, Python, and C++) was analyzed in [8].

3. Tree of triangulations and expression of the Catalan numbers

The first step in our method is generating a complete triangulation tree from the initial basic triangle to the given n-gon. The resulting triangulation hierarchy is based on the decomposition of the Catalan numbers (this method is described in details in [9]). Similarly, in [10], we presented the construction of hierarchy of triangulations based on lattice path and ballot problem. Below, we will list some basics of interpretation of the Catalan numbers in the form of a set of expressions $(2+i)$.

Let us denote a triangulation $\tau_{n-1} \in \mathcal{T}_{n-1}$ satisfying $\deg(n-1) = l$ by τ_{n-1}^l. Assume that diagonals incident to $n-1$ are sorted from the left by $\delta_{i_k,n-1}$, $k = 1,\ldots,l$. Then the number of diagonals incident to $n-1$ which are located left from $\delta_{i_k,n-1}$ is equal to $k-1$ [11]. The sons of τ_{n-1}^l are derived by "splitting" the diagonals incident to vertex $n-1$

$$\delta_{i_k,n-1},\ k = 1,\ldots,l,\ i_k \in \{1,\ldots,n-2\},\ 1 = i_1 < i_2 < \cdots < i_l = n-2$$

in increasing order with respect to i_k. If we split the diagonal $\delta_{i_k,n-1}$, we get the son $S^{i_k}(\tau_{n-1}^l)$. Then the sons of triangulation τ_{n-1}^l are

$$S^{i_1}(\tau_{n-1}^l), S^{i_2}(\tau_{n-1}^l), \ldots, S^{i_l}(\tau_{n-1}^l). \tag{3}$$

The splitting of the diagonal $\delta_{i_k,n-1}$ produces the son $S^{i_k}(\tau_{n-1}^l)$ with $\deg(n) = 2+k-1$. We are assigning the weights of the form $(2+i)$ to the edges in the tree of triangulations. When numbering the edges in the tree of triangulations, we will be guided by the basic principle that each weight of an edge represents the number of descendants for a triangulation at end of this edge [12]. The number of τ_{n-1}^l descendants is between 2 and $n-2$. So, from one particular triangulation of P_{n-1}, we can derive $2+i$ triangulations of P_n where $i \in \{0,1,\ldots,n-4\}$ in the general case. As the number of descendants is greater than or equal to 2, the usage of the weights $(2+i)$, $i \in \{0,1,\ldots,n-4\}$ is obvious.

By $\left(\tau_{n-1}^l, S^{i_k}(\tau_{n-1}^l)\right)$, we denote the edge in the tree of triangulations connecting the nodes $\tau_{n-1}^l \in \mathcal{T}_{n-1}$ and $S^{i_k}(\tau_{n-1}^l) \in \mathcal{T}_n$ between the levels $n-1$ and n. The triangulation $S^{i_k}(\tau_{n-1}^l)$ is derived by splitting the diagonal $\delta_{i_k,n-1}$. Since the diagonal $\delta_{i_k,n-1}$ has $k-1$ diagonals left to it and incident to the vertex $n-1$, by splitting this diagonal, we get $2+k-1$ descendants of the triangulation $S^{i_k}(\tau_{n-1}^l)$ (i.e. $2+k-1$ diagonals

incident to n). Then, we assign the expressions of the form $(2+i)$ as the weights to outgoing edges of τ_{n-1}^l, namely

$$\left(\tau_{n-1}^l, S^{i_1}(\tau_{n-1}^l)\right), \ldots, \left(\tau_{n-1}^l, S^{i_l}(\tau_{n-1}^l)\right). \tag{4}$$

The weight $(2+k-1)$ of the edge $\left(\tau_{n-1}^l, S^{i_k}(\tau_{n-1}^l)\right)$ means that the triangulation τ_{n-1}^l has $k-1$ diagonals incident to vertex $n-1$ left to $\delta_{i_k,n-1}$, and that $S^{i_k}(\tau_{n-1}^l) \in \mathcal{T}_n$ has $2+k-1$ descendants. According to Eq. (3), the sons of τ_{n-1}^l are derived by splitting the diagonals $\delta_{i_1,n-1}, \ldots, \delta_{i_l,n-1}$. The edges in Eq. (4) have the following weights $(2+0), (2+1), \ldots, (2+l-1)$, respectively.

The triangulation τ_{n-1}^l has l descendants. According to the adopted principle of assigning weights to the tree of triangulations, incoming edge to node τ_{n-1}^l has the weight $2+l-2$. In the origin (before the tree root), there is the expression $(2+0)$, which indicates that the triangle in the tree root has an ear in vertex 3, i.e. has 2 descendants (these are two possible triangulations of a quadrilateral).

The sum of all weights of the edges connecting tree levels $n-1$ and n is equal to C_{n-1}, and the number of summands, i.e. the number of these edges is equal to C_{n-2}. By $\Delta(C_{n-1})$, we denote the decomposition of C_{n-1} defined by summing all weights of edges connecting tree levels $n-1$ and n. The cardinal number of basic terms of the form $(2+i)$ in $\Delta(C_{n-1})$ is equal to C_{n-2}. The sum of terms included in $\Delta(C_{n-1})$ is equal to C_{n-1} [9].

The incoming edge to node τ_{n+1}^l with the weight $2+i$ produces the edges outgoing to the node τ_{n+1}^l with weights $(2+0), (2+1), \ldots, (2+i+1)$. Therefore, our decomposition of Catalan number C_n can be derived from the already made decomposition of C_{n-1} and the transformation f defined by

$$(2+i) \to (2+0) + (2+1) + \cdots + (2+i+1) = f(2+i),\ i \geq 0. \tag{5}$$

Further, assume the operator f is distributive with respect to $+$:

$$f\left(\sum(2+i)\right) = \sum f(2+i) = \sum \sum_{l=0}^{i+1}(2+l).$$

The first property of our decomposition $\Delta(C_n)$ is discovered in Lemma 3.1.

Lemma 3.1 *For each $n \geq 4$, the recurrent relation $\Delta(C_n) = f(\Delta(C_{n-1}))$ is valid.*

There is C_{n-2} edges between tree levels $n-1$ and n. For each l, the edge weights

$$\left(\tau_n^l, S^{i_1}(\tau_n^l)\right), \ldots, \left(\tau_n^l, S^{i_l}(\tau_n^l)\right) \tag{6}$$

are defined applying the function f on weight of some of the edges $\left(\tau_{n-1}^k, S^{i_p}(\tau_{n-1}^k)\right)$, $k, p \leq l$.

As the sum of weights, belonging to the edges connecting tree levels $n-2$ and $n-1$, is equal to C_{n-2} and the sum of weights assigned to edges connecting tree levels $n-1$ and n, is equal to C_{n-1}, it is clear that the following holds: $\Delta(C_{n-1}) = f(\Delta(C_{n-2}))$, or $\Delta(C_n) = f(\Delta(C_{n-1}))$ in general case.

In [9], we presented one way of decomposing the Catalan numbers that can be used to generate tree of triangulation (or hierarchy of triangulation). For the sake of simplicity, we used the notation

$$\sigma_i = (2+0) + (2+1) + \cdots + (2+i). \tag{7}$$

Then Eq. (5) should be simplified to $f(\sigma_0) = \sigma_1,\ f(2+i) = \sigma_{i+1},\ i \geq 1$.

Furthermore, using $f(\sigma_i) = \sigma_1 + \cdots + \sigma_{i+1} = \sum_{k=1}^{i+1} \sigma_k$, $i \geq 0$ expression of the Catalan numbers should be further simplified:

$$\begin{aligned}
&\Delta(C_2) = (2+0) = \sigma_0,\\
&\Delta(C_3) = \sigma_1,\\
&\Delta(C_4) = \sum_{l=1}^{2} \sigma_l,\\
&\Delta(C_5) = \sum_{l=1}^{2} \sigma_l + \sum_{l=1}^{3} \sigma_l,\\
&\Delta(C_6) = \left(\sum_{l=1}^{2} \sigma_l + \sum_{l=1}^{3} \sigma_l\right) + \left(\sum_{l=1}^{2} \sigma_l + \sum_{l=1}^{3} \sigma_l + \sum_{l=1}^{4} \sigma_l\right).
\end{aligned} \tag{8}$$

Now, we replace

$$f\left(\sum_{l=1}^{i} \sigma_l\right) = \sum_{l=1}^{i} f(\sigma_l) = \sum_{l=1}^{i} \sum_{l_1=1}^{l+1} \sigma_{l_1},\ i \geq 1.$$

By this, we simplify Eq. (8) and get:

$$\begin{aligned}
&\Delta(C_2) = \sigma_0,\\
&\Delta(C_3) = f(\sigma_0) = \sigma_1,\\
&\Delta(C_4) = f(\sigma_1) = \sum_{l=1}^{2} \sigma_l,\\
&\Delta(C_5) = \sum_{l=1}^{2} f(\sigma_l) = \sum_{l=1}^{2} \sum_{l_1=1}^{l+1} \sigma_{l_1},\\
&\Delta(C_6) = \sum_{l=1}^{2} f^2(\sigma_l) = \sum_{l=1}^{2} \sum_{l_1=1}^{l+1} f(\sigma_{l_1}) = \sum_{l=1}^{2} \sum_{l_1=1}^{l+1} \sum_{l_2=1}^{l_1+1} \sigma_{l_2}.
\end{aligned}$$

Theorem 3.1 *Decomposition of the Catalan number C_n defined by summation of all weights of edges connecting tree levels n and $n+1$ is defined by*

$$\Delta(C_n) = f^{n-2}(\sigma_0) = f^{n-3}(\sigma_1) = \sum_{k=1}^{2} f^{n-4}(\sigma_k) = \sum_{l=1}^{2} \sum_{l_1=1}^{l+1} \cdots \sum_{l_{n-4}=1}^{l_{n-5}+1} \sigma_{l_{n-4}},\ n \geq 3, \tag{9}$$

where σ_i is defined in Eq. (7).

From the inductive hypothesis and Lemma 3.1, we get

$$\Delta(C_n) = f(\Delta(C_{n-1})) = f\left(\sum_{l=1}^{2}\sum_{l_1=1}^{l+1}\cdots\sum_{l_{n-5}=1}^{l_{n-6}+1}\sigma_{l_{n-5}}\right)$$

$$= \sum_{l=1}^{2}\sum_{l_1=1}^{l+1}\cdots\sum_{l_{n-5}=1}^{l_{n-6}+1} f(\sigma_{l_{n-5}}) = \sum_{l=1}^{2}\sum_{l_1=1}^{l+1}\cdots\sum_{l_{n-5}=1}^{l_{n-6}+1}\sum_{l_{n-4}=1}^{l_{n-5}+1}\sigma_{l_{n-4}},$$

which completes the proof.

Besides $\Delta(C_n) = f(\Delta(C_{n-1}))$, the following recurrent relation is also satisfied. For $n \geq 4$, the general recurrent relation holds $\Delta(C_n) = f^k(\Delta(C_{n-k}))$, $k \geq 1$. Catalan number decomposition into the sum of terms of the form $(2+i), i \in \{0, \ldots, n-4\}$, guides us in generation of $\mathcal{T}_n$ using the already known $\mathcal{T}_{n-1}$ (for more details see [9]). Decomposition is unique and suitable for generation of tree of triangulations (Figure 1).

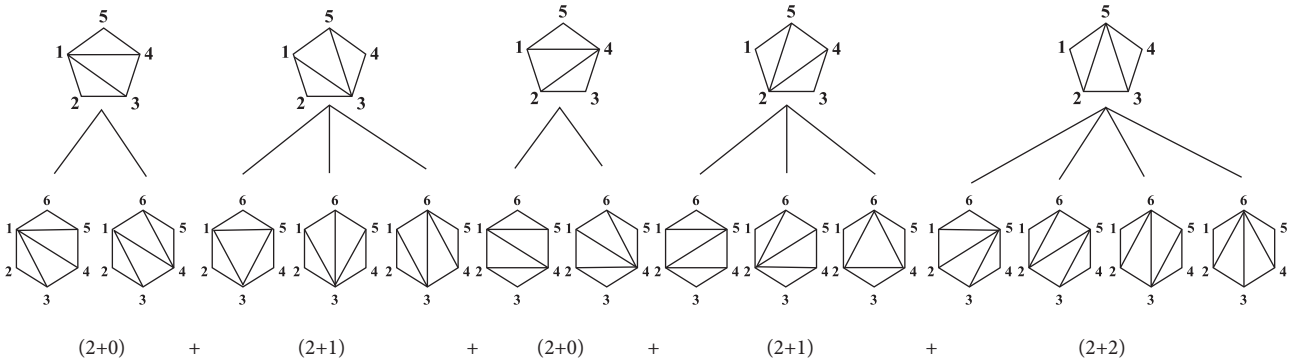

Figure 1. Levels five and six of the tree of triangulations based on decompostion of the Catalan numbers.

This idea with the Catalan numbers decomposition was used in the construction of an algorithm for the trivalent trees method. In this way, we performed triangulation of the convex polygon based on graph of triangulations of a convex polygon and tree of triangulations (similar principles as Hurtado's in [7]).

4. Relationship of Catalan numbers and ballot problem

The second step in our method is storing of all obtained triangulation in tree based on ballot notation. The ballot problem can be illustrated graphically by a lattice paths under certain constraints in the Cartesian coordinate system. In the two-dimensional space, by paths from the lattice point $(0,0)$ to the lattice point (n,n), we mean directed paths beginning in $(0,0)$ and ending in (n,n).

The paths that are always underneath diagonal and passes through lattice only with movements right (R) and up (U) are called legal paths. Let us define two movements starting at the point $(0,0)$ in the Cartesian coordinate system:

$$R : (x,y) \longrightarrow (x+1,y), U : (x,y) \longrightarrow (x,y+1).$$

We see that the path one unit on the right is defined with R, and with U one unit up in the lattice. With the application of these movements, we obtain that the problem of different paths from the point $(0,0)$ to the point (n,n) in the lattice correspond to the ballot problem. Every legal path on lattice must begin with an A and end in a B.

The number of valid movement through the $n \times n$ grid is the Catalan number C_n (for details on the combination of ballot notation and lattice path or problem of movement in discrete grid see [10]). Any movement to right corresponds with the vote for candidate A, while any movement to the up corresponds with the vote for candidate B. There is a bijection between the set of well-formed sequences of parentheses with n pairs and the a set of legal lattice paths. This procedure is presented in Example 1.

Example 1 *The following table can be formed for the voting sequence* ***ABAABB****. This voting sequence corresponds to the movement through the lattice path. Note that integers within a row denote numbers of the specific vote occurrence.*

Table 1. The tally of the votes in the ballot record ***ABAABB***

Tally for candidate	i_1	i_2	i_3	i_4	i_5	i_6
	A	***B***	***A***	***A***	***B***	***B***
A	1	1	2	3	3	3
B	0	1	1	1	2	3

On the basis of bijection, we can present a set of well-formed sequences in parentheses with n pairs with the ballot record. For this purpose, let us replace each left parenthesis in the sequence of well-formed parentheses with an A and the right with a B. In this way, we get the unique presentation of the corresponding set of well-formed sequence of parentheses with n pairs by the ballot record.

The problem of well-formed sequences of parenthesis corresponds to the problem number of different ways of multiplying. To prove the mutual bijection, we establish the bounce in the following way: we remove all except right parentheses and signs of multiplying. We replace the multiplying signs with the left parenthesis. The obtained sequence of parenthesis is a sequence of well-formed parenthesis. This procedure is presented in Example 2.

Example 2 *Well-formed parenthesis:* $(((ab)c)d), (a(b(cd))), ((a(bc))d), ((ab)(cd)), (a((bc)d)))$ *are five possible multiplication expressions (they also correspond to planted trivalent tree of pentagon).*

$$\begin{gathered}(((ab)c)d) \longleftrightarrow ()()(); \ (a(b(cd))) \longleftrightarrow ((())); \\ ((a(bc))d) \longleftrightarrow (())(); \ ((ab)(cd)) \longleftrightarrow ()((())); \\ (a((bc)d)) \longleftrightarrow (()()).\end{gathered}$$

The third step in our method is generation of individual triangles within a certain level of the triangulation hierarchy based on the planted trivalent binary tree (from ballot notation in PTBT or vice versa).

5. Algorithms for polygon triangulations based on trivalent binary tree and ballot notation

Let the PTBT correspond to the convex polygon with $v = n + 2$ vertices. The path between vertices of the tree defines the appearance of the sign A, while from the vertex of the tree to the corresponding leaves defines the appearance of the sign B.

Figure 2 shows the general form of moving through the polygon where the movement is based on the appearance of signs A and B in the ballot record. In this way, it is always possible to determine the path of movement.

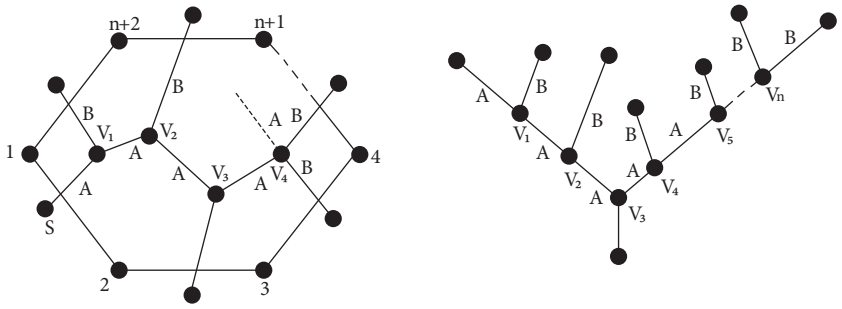

Figure 2. General schema of the triangulated polygon and their corresponding planted trivalent binary tree.

Corresponding planted trivalent binary tree with $n-1$ leaves used in Algorithm 1 is constructed by choosing one particular edge called base (in our case $(2,3)$). For this algorithm, we choose a point inside every triangle in the triangulated polygon, and one point outside each side of the triangle except the side which is in the interior of the polygon. The resulting graph is a planted trivalent binary tree with $n-1$ leaves.

Algorithm 1 From ballot notation to triangulation based on PTBT.

Require: A ballot record for PTBT, denoted by $b = \{b_1, \ldots, b_{2n}\}$.

1: **Initialization**: Create a polygon with $v = n+2$ vertices. Make the corresponding PTBT. The vertices of the tree (except the root) are marked w,th V_j, $j = 1, \ldots, n$ in the counterclockwise order. The leaves of tree corresponding to the outside points are labelled by B except the starting point. Every vertex has two indicators $visited$ and $finished$, all initially on $false$. Set $k = 0$. Make an empty list $aux = \{\}$. Set an array $visited_j$, $j = 1, \ldots, n$ on $false$. Begin the movement from (1,2) side in the counterclockwise order. Set $k = 1$, $j = 1$, the starting point of the movement is $P_k = S$ and always begin with $b_1 = A$.

2: for $i = 1$ to $2n$

2.1: If the current character in the ballot record is $b_i = A$, then find the smallest $j_2 \geq j$, where $visited_{j_2} = false$. Set $j = j_2$, $k = k+1$, $P_k = S$.

2.2: If the current character in the ballot record is $b_i = B$, then find the smallest $j_2 \geq j$ set $visited_{j_2} = false$ and $visited_{j_2} = true$.

2.2.1: If there are two successive characters B in ballot record, then return to the parent of $B(V_j)$ and set $j = j-1$.

3: Draw the internal diagonals based on Algorithm 2.

Output: Generated triangulation-based PTBT corresponding to the ballot record from the input.

Algorithm 2 Finding intersections in the polygon.

Require: Path P_k, $k = 1, \ldots, n$ inside a polygon.

1: For $i = 1$ to m; where m is the number of all polygon diagonals, $m = v(v-3)/2$

1.1: Choosing the diagonal from the set of all internal diagonals ($\{\delta_{1,3}; \ldots \delta_{1,v}\}$, $\{\delta_{2,4}; \ldots \delta_{2,v}\}$ etc.)

1.1.1: If intersection in a polygon with only one part of path is detected *AND* if it does not intersect with the previous contained diagonal, *THEN* draw the diagonal *ELSE* choose next diagonal (go to Step 1.1)

Output: $v-3$ internal diagonals (without intersection).

Example 3 *Let us illustrate how Algorithm 1 produces a pentagon triangulation corresponding to the ballot record ABAABB (which corresponds to well-formed sequence of parentheses with 3 pairs ()(()) or planted trivalent binary tree given in the first step). The process can be presented with the following steps:*

(1) The first step is to create the corresponding planted trivalent binary tree for the pentagon. This tree depends on n, and can be easily generated, with corresponding vertex labeling and initializing attributes visited and finished on false and integer k on 0. Also, $aux = \{\}$, see Figure 3.

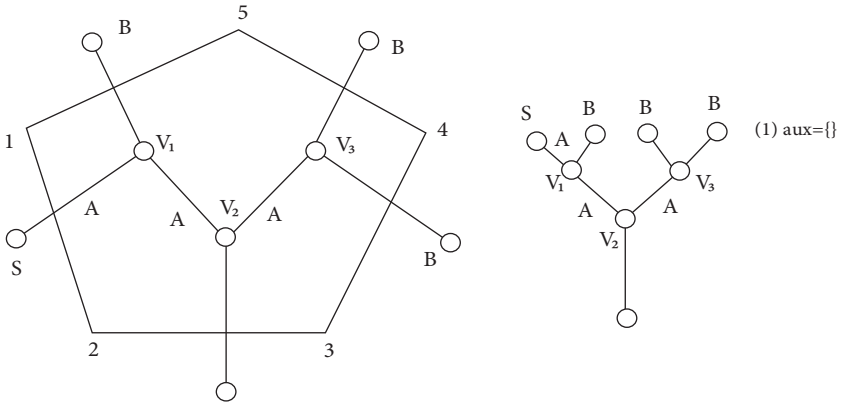

Figure 3. Create the corresponding planted trivalent binary tree for the pentagon.

(2) Since the first character in ballot record is A, move from the starting point S to the next V labeled descendent (this is V_1); $k = 1$, $aux = aux \cup \{k\}$, $V_1.visited = true$.

(3) The next character is B; set B.visited and B.finished on true, and return to the parent of leaf $B(V_2)$. The next character is A, go to the V_2, $k = k + 1$, $aux = aux \cup \{k\}$, $V_2.visited = true$, see Figure 4.

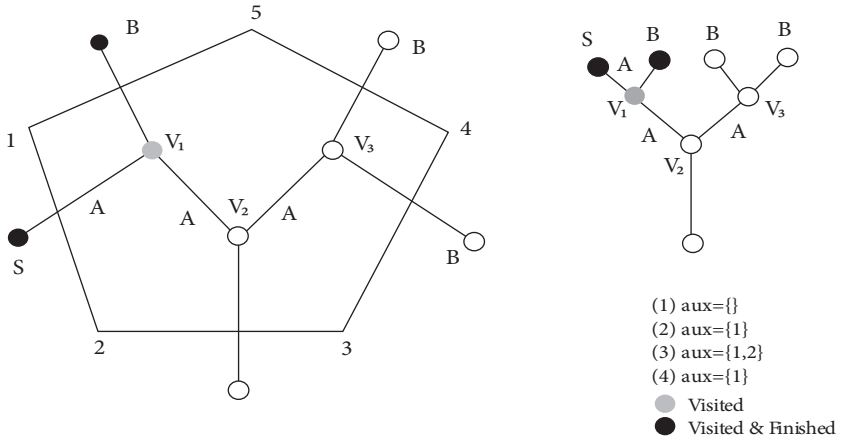

Figure 4. Steps 2 and 3.

(4) The next character is A; go to the V_3, $k = k + 1$, $aux = aux \cup \{k\}$, $V_3.visited = true$ The next character is B, set B.visited and B.finished on true, and return to the parent of leaf $B(V_3)$.

(5) The next character is B; set B.visited and B.finished on true, set $V_3.visited = finished$, return to vertex V_2 set $V_2.finished = true$, return to vertex V_1 and set $V_1.finished = true$. Thus, there are no more characters in the ballot record, the movements are finished, see Figure 5.

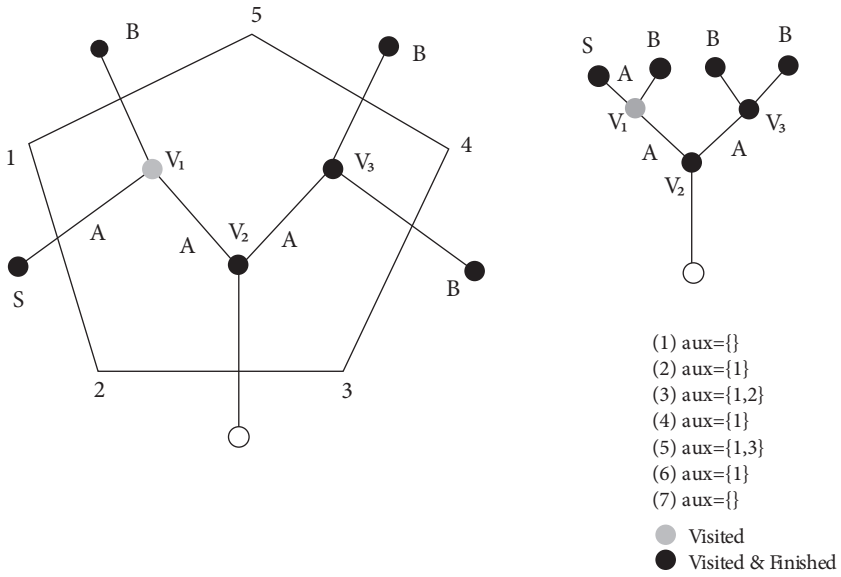

Figure 5. Steps 4 and 5.

(6) Upon the path P traversed inside the polygon, the corresponding triangulation can be constructed. Draw all possible internal diagonals that cut a path inside the polygon marked with A (based on Algorithm 2). In this case, the possible noncrossing internal diagonals that intersect the given path marked with A's (i.e. with V_1, V_2, V_3) are $\delta_{2,5}$ and $\delta_{3,5}$.

Now we present a reverse algorithm for this process; namely, how to get an appropriate ballot record from the convex polygon triangulation based on PTBT (reverse of the Algorithm 1). Algorithm 3 shows the procedure of generating corresponding notation based on PTBT triangulation.

Algorithm 3 From triangulation based on PTBT to ballot notation.

Require: A convex polygon triangulation based on PTBT.

1: Make the corresponding planted trivalent binary tree for given polygon triangulation, where the point out of polygon next to the side $(2;3)$ is omitted.

2: Mark the path between vertices in the tree with A and tree leaves by B. The exception is point next to $(1;2)$ which is a starting point and needs to be marked with A.

3: Use in-order traversal taking vertex marks along the traversal. In this way, we get the ballot record.

Output: The corresponding ballot record.

Example 4 *Let us illustrate the application of Algorithm 3 in the process of moving through the pentagon triangulation defined by internal diagonals $\delta_{2,5}$ and $\delta_{3,5}$. Create the corresponding planted trivalent binary tree for the given triangulation and use in-order traversal in taking vertex marks. Movement between two vertices marks with A, between vertex and leaf with B.*

(1) The triangulation tour starts from position S (see the figure below), more precisely from the edge $(1;2)$, and the first character in the ballot notation is marked with A. Set $k=1$, $aux = aux \cup \{k\}$, and $V_1.visited = true$.

(2) Go to the leaf of PTBT (outer edge (1; 5)) and mark character in the ballot notation with B.

(3) Go to the vertex V_2 of PTBT, set $k = k + 1$, $aux = aux \cup \{k\}$, and $V_2.visited = true$. Mark the third character in ballot notation with A.

(4) Go to the vertex V_2 of PTBT, set $k = k + 1$, $aux = aux \cup \{k\}$, and $V_3.visited = true$. Mark the fourth character in ballot notation with A.

(5) Go to the leaf of PTBT (outer edge (4; 5)) and mark character in the ballot notation with B.

(6) Go to the leaf of PTBT (outer edge (3; 4)) and mark character in the ballot notation with B. Set $V_3.visited = finished$, return to vertex V_2 set $V_2.finished = true$, return to vertex V_1 and set $V_1.finished = true$. We will get the corresponding ballot record ABAABB*, see Figure 6.*

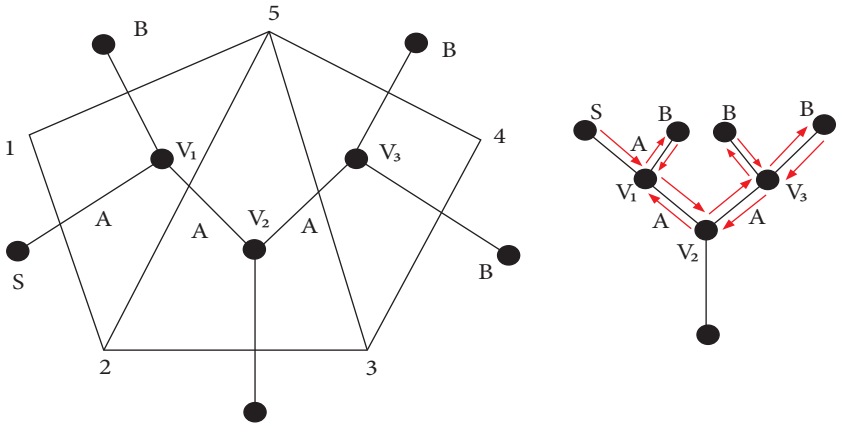

Figure 6. Triangulation based on PTBT and corresponding ballot notation $ABAABB$.

6. Comparative analysis of experimental results

In order to get an evaluation of the presented methods (ballot lattice path [10] and ballot trivalent trees), we offer a comparative analysis with the existing method for triangulation of a convex polygon given by Hurtado and Noy [7] (hereinafter the Hurtado method). The reason why we chose this method is that the generation of triangulation of a convex polygon is based on hierarchy or trees of triangulations. The number of all combinations in each level is determined by expressions of Catalan number.

Generating triangulation in our methods (ballot lattice [10] and ballot trivalent) is also based on expressions of the Catalan number for n-gon, respectively on n-th level like as in the existing Hurtado method. The only difference is in generating the new triangulation, by Hurtado method the process goes by splitting the outer edge of parents in generating new descendants; and thus, it can generate their hierarchy (about the topics see [7]). However, in our method, generating new descendants is based on the basis of movement through the polygon based on a given algorithm. All methods are implemented in Java programming language (NetBeans environment). Details of implementation and testing of Hurtado–Noy algorithm are presented in [8]. The testing results were analyzed from two aspects:

1) *Speed of execution (Table 2)*: Time which refers to the phase of generating triangulation without storage and total time for all three phases: input, generating triangulation, and output (their storage).

2) *Working memory (Table 3)*: Buffer occupancy during generating all triangulations for the given n.

Table 2 presents the results of testing for three methods. Number of triangulation of polygons with the number of vertices $v \in \{5, 6, ..., 15, 16\}$ is $t \in \{5, 14, ..., 742900, 2674440\}$. We considered the differences in times for generating triangulations for all methods. Implementation and experimental results of this kind of storage methods were given in [10] and a part of them is shown in Table 2.

Table 2. Experimental results: ballot lattice, ballot trivalent, and Hurtado tree (execution CPU times).

n-gon	T	Execution CPU times (in seconds)					
		lBT +	lBT –	lBL +	lBL –	lHT +	lHT –
5	5	**0.23**	0.21	**0.24**	0.22	**0.25**	0.21
6	14	**0.32**	0.28	**0.33**	0.31	**0.34**	0.29
7	42	**0.41**	0.34	**0.41**	0.38	**0.43**	0.35
8	132	**0.49**	0.40	**0.47**	0.44	**0.49**	0.40
9	429	**0.62**	0.54	**0.63**	0.60	**0.67**	0.55
10	1430	**1.15**	0.87	**1.03**	0.98	**1.18**	0.89
11	4862	**3.79**	2.14	**3.51**	3.09	**3.81**	2.19
12	16,796	**12.41**	5.69	**11.29**	10.11	**12.46**	5.81
13	58,786	**49.91**	15.07	**43.55**	38.81	**50.51**	15.24
14	208,012	**107.02**	42.22	**108.07**	97.13	**119.05**	46.34
15	742,900	**272.66**	124.18	**274.19**	244.02	**318.63**	124.18
16	2,674,440	**673.23**	570.01	**677.17**	572.87	/	/

Symbols in Table 2: BL-Ballot lattice method ([+] with or [–] without graphic generation of triangulations), BT-Ballot planted trivalent binary tree method ([+] with or [–] without graphic generation of triangulations), HT-Hurtado method ([+] with or [–] without graphic generation of triangulations).

The graph in Figure 7 shows the CPU in the process of generating triangulation, where it can be noted that the greatest load is in the Hurtado method, then the ballot lattice method, and the least load in the ballot trivalent method.

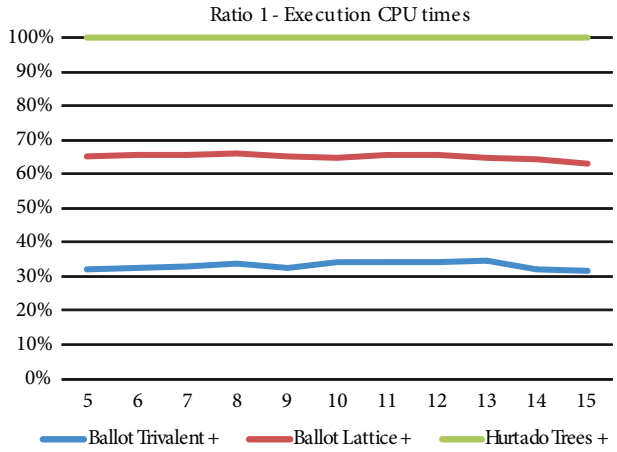

Figure 7. Load of CPU

Table 3 presents a comparison of storage requirements for the tested methods (load memory in KB). We give differences in the size of required memory as well as the percentage share of the time needed for storage in the program execution.

Table 3. Experimental results: ballot Lattice, ballot trivalent, and Hurtado Tree (load memory).

n-gon	No. of triang.	Load memory (in Kb)		
		BT	BL	HT
5	5	0.02	**0.02**	0.12
6	14	0.08	**0.08**	0.46
7	42	0.29	**0.29**	1.76
8	132	1.05	**1.05**	6.46
9	429	3.86	**3.86**	24.49
10	1430	14.31	**14.30**	93.13
11	4862	53.42	**53.40**	355.67
12	16,796	196.12	**196.00**	1363.43
13	58,786	502.71	**502.00**	4121.13
14	208,012	1442.00	**1441.00**	11,523.62
15	742,900	4299.90	**4295.00**	29,874.29
16	2,674,440	12,773.89	**12,752.00**	/

The graph in Figure 8 shows the workload ratio of the triangulation storage process, where it can be noted that the greatest load is in the Hurtado method, then the ballot lattice method, and the least load is on the ballot trivalent method (the graph is generated based on the values from Table 3).

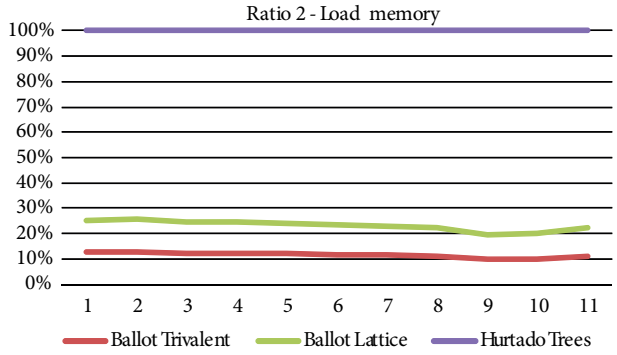

Figure 8. Load of memory.

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

Based on the testing results, it can be concluded that ballot lattice and ballot trivalent method give better results in the case where both generation and storage are included at the same time. Significant savings are achieved by applying the ballot output.

6.1. Complexity of the algorithms

Now, we will analyze the ratio of the data (notation for triangulations) amount for the Hurtado algorithm and ballot trivalent algorithm. The Hurtado method uses all internal and external diagonals, while the ballot trivalent method uses only internal diagonals. The Hurtado method requires recording of coordinates of n edges, plus two coordinates in storage $(n-3)$ of internal diagonals: $2n+2(n-3)$. From the aspect of the storage complexity the Hurtado algorithm method requires:

$$HT_s = 2(2n-3).$$

The ballot trivalent method requires a ballot notation that is $2(n-2)$ length. Because the first character (bit 1 or character A) and the last character (bit 0 or character B) in ballot notation are always known, the number of storage data is reduced by two. From the aspect of necessary data storage amount, the ballot trivalent method requires:

$$BT_s = 2(n-2)-2 = 2(n-3).$$

Let us estimate the processing complexity of both methods as we generate C_{n-2} triangulations of an n-gon by completing $2(n-2)$ permutation the number of movements in *lattice path* or in the *trivalent binary tree*. Total number of operations for our ballot trivalent method for all triangulations of n-gon is

$$BT_p = 2C_{n-2}(n-2).$$

On the other hand, in the case of the Hurtado algorithm, we have the following. For every triangulation at level $n-1$, we need to perform $2n-5$ checks to find the diagonals incident to the vertex $n-1$. Total number of these checks is $(2n-5)C_{n-3}$.

Furthermore, we must go through diagonals and copy some without transforming, while some of them should be transformed and two new diagonals should be inserted for every incident where diagonal is found. In this way, we make $2n-3$ pairs describing one new triangulation. The total number of incident diagonals is equal to C_{n-2}. Total number of operations for the Hurtado method for all triangulations of n-gon is

$$HT_p = (2n-5)C_{n-3} + (2n-3)C_{n-2}.$$

It is not difficult to verify the inequality for all values of $n \geq 5$:

$$(2n-5)C_{n-3} + (2n-3)C_{n-2} > 2C_{n-2}(n-2).$$

Table 4 displays numerical data related to the complexity of algorithms in two aspects: storage and processing complexity. The number of operations required to generate all triangulations of n-gon is presented. In addition, the required data storage amount of a single triangulation of n-gon is shown.

7. Conclusion and further work

The proposed method in the paper provides an effective way of constructing the convex polygon triangulation with the planted trivalent binary tree and ballot record. The algorithms are related to the combination of the properties of Catalan numbers decomposition, convex polygon triangulation, planted trivalent binary trees, and ballot combinatorics. The significance of the presented algorithms is reflected in the effective construction of the convex polygon triangulation and their processing and storing in the form of ballot or binary notation.

Table 4. Ballot trivalent (BT) vs Hurtado trees (HT): storage and processing complexity.

n-gon	Storage complexity for one triangulation			Processing complexity for all triangulations		
	BT_storage	HT_storage	Savings	BT_processing	HT_processing	Speedup
5	4	14	10	30	45	15
6	6	18	12	112	161	49
7	8	22	14	420	588	168
8	10	26	16	1584	2178	594
9	12	30	18	6006	8151	2145
10	14	34	20	22,880	30,745	7865
11	16	38	22	87,516	116,688	29,172
12	18	42	24	335,920	445,094	109,174
13	20	46	26	1,293,292	1,704,794	411,502
14	22	50	28	4,992,288	6,552,378	1,560,090
15	24	54	30	19,315,400	25,258,600	5,943,200

We hope that this research serves as the first step toward further developing this important issue to the concave polygons and polyhedron. By decomposing the concave polygon in a set of convex polygons algorithms can be extended to the case of a concave polygon. The decomposition will be the task of identifying of the reflex vertex/vertices and creating a new edge/edges and vertex/vertices (if it is necessary) until there is no reflex vertex in the concave polygon. It is important to remark here that the best solution to the problem is the decomposition with the least produced diagonals.

Similar to the case of the concave polygon, by identifying the notch edges in the polyhedron, the algorithms can be extended to solving problems in 3D. The solution of the problem would be looked in the identification of notch edges and plane cuts of the polyhedron until all polyhedrons in decomposition become convex.

References

[1] Koshy T. Catalan Numbers with Applications. New York, NY, USA: Oxford University Press,2009.

[2] Even S. Graph Algorithms. New York, NY, USA: Cambridge University Press, 2011.

[3] Goodman JE, O'Rourke J. Handbook of Discrete and Computational Geometry. New York, NY, USA: Chapman and Hall and CRC Press, 2004.

[4] O'Rourke J. Computational Geometry in C. 2nd ed. Cambridge, UK: Cambridge University Press, 1997.

[5] Preparata F, Shamos MI. Computational Geometry: An introduction. Berlin, Germany: Springer-Verlag, 1985.

[6] Koshy T. Discrete Mathematics with Applications. Burlington, MA, USA: Elsevier Academic Press, 2004.

[7] Hurtado F, Noy M. Graph of triangulations of a convex polygon and tree of triangulations. Computational Geometry 1999; 13, 179-188.

[8] Saračević M, Stanimirović P, Mašović S, Biševac E. Implementation of the convex polygon triangulation algorithm. Facta Universitatis, Series Mathematics and Informatics 2012; 27, 213-228.

[9] Stanimirović P, Krtolica P, Saračević M, Mašović S. Decomposition of Catalan numbers and convex polygon triangulations. International Journal of Computer Mathematics 2014; 19, 1315-1328.

[10] Saračević M, Stanimirović P, Krtolica P, Mašovič S. Construction and notation of convex polygon triangulation based on ballot problem. Romanian Journal of Information Science and Technology 2014; 17, 237-251.

[11] Sen-Gupta S, Mukhopadhyaya K, Bhattacharya BB, Sinha BP. Geometric classification of triangulations and their enumeration in a convex polygon. Computers and Mathematics with Applications 1994; 27, 99-115.

[12] Saračević M, Mašović S, Milošević D. JAVA Implementation for triangulation of convex polygon based on Lukasiewicz algorithm and binary tree. Southeast Europe Journal of Soft Computing 2013; 2, 40-45.